Project Title :

Tic Tac Toe

(By SEMESTER - VII 4thYear M.Sc.(2023-24))

Submitted By -

Name	Roll No
Varsha Oza	4043
Tinkal Prajapati	4056
Maitri Sanghvi	4061

Submitted To -

K. S. School of Business management

M.Sc. – Computer Application and Information Technology

Index -

1. PROJECT INTRODUCTION

Tic-Tac-Toe is one of the paper-and-pencil games. This game requires two players in 3x3 grid with Player 1 acts as “O” and Player 2 acts as “X”, or vice versa. The objective of this game is to take place of three connecting grids in a horizontal, vertical, or diagonal way/fork.

This game was first introduced at ancient time, however there is no evidence who invented it and which year. Some people think this game was invented at Ancient Egypt, and then Roman Empire called this game “Terni Lapilli”. The grid drawing for the game had been found chalked all over the ancient city’s ruins.

Terni Lapilli was resurfaced in England with the name “Nought and Crosses” in 1864. This resurfaced version is the modern of Tic-Tac-Toe game that people know until this present day.

In 1952, Alexander S. Douglas for the EDSAC computer at University of Cambridge developed a computerized Tic-Tac-Toe game called “OXO”. This was the first video game of Tic-Tac-Toe and it has AI inside, therefore human could play against the computer opponent.

2. PROJECT OVERVIEW

Tic-Tac-Toe is one of the paper-and-pencil games. This game requires two players in 3x3 grid with Player 1 acts as “O” and Player 2 acts as “X”, or vice versa.

Tic-Tac-Toe is a two-player game played on a 3x3 grid. Player 1 uses "O," and Player 2 uses "X". The objective is to achieve three connecting marks in a horizontal, vertical, or diagonal line. Develop a Python-based Tic-Tac-Toe game incorporating the Min-Max algorithm for an intelligent opponent.

Design an intuitive and visually appealing user interface for the Tic-Tac-Toe game. Enhance user experience by incorporating interactive elements and clear visual feedback. Enable users to play against the computer, which strategically evaluates and selects optimal moves. Provide an immersive and challenging gaming experience.

By implementing the Min-Max algorithm, this project aims to elevate the traditional Tic-Tac-Toe game into a sophisticated and strategic AI-driven experience.

3. METHODOLOGY

Using Min- Max Algorithm

Artificial Intelligence (AI) can be used for this game in order to play only single player, in other words human against computer. There are many samples of this game with AI on the internet, and each sample has its own algorithm for AI and it depends on the developer, which kind of algorithm will be used. One of AI algorithms that can be used for Tic-Tac-Toe is Minimax algorithm. This algorithm is a reliable algorithm for Tic-Tac-Toe game. By using this strategy, computer will avoid the loser condition against human.

This program is written in python and it has simple AI algorithm that can make people understand about how to implement AI in the game.

This AI program has two strategies based on Newell and Simon’s 1972 Tic-Tac-Toe program:

a) Try to take the center pile if human player did not take it.
b) Block the opponent way, if there are two piles taken by opponent already either in horizontal, vertical, or diagonal fork.

4. RESULT

1. Logic of the tic tac toe

```
import math
import copy

X = "X"
O = "O"
EMPTY = None


def initial_state():
    """
    Returns starting state of the board.
    """
    return [[EMPTY, EMPTY, EMPTY],
            [EMPTY, EMPTY, EMPTY],
            [EMPTY, EMPTY, EMPTY]]


def player(board):
    """
    Returns player who has the next turn on a board.
    """
    xCounter = 0
    oCounter = 0

    for i in range(0, len(board)):
        for j in range(0, len(board[0])):
            if board[i][j] == X:
                xCounter += 1
            elif board[i][j] == O:
                oCounter += 1

    if xCounter > oCounter:
        return O
    else:
        return X


def actions(board):
    """
    Returns set of all possible actions (i, j) available on the board.
    """
    possibleActions = set()

    for i in range(0, len(board)):
        for j in range(0, len(board[0])):
            if board[i][j] == EMPTY:
                possibleActions.add((i, j))

    return possibleActions


def result(board, action):
    """
    Returns the board that results from making move (i, j) on the board.
    """
    # Create new board, without modifying the original board received as
input
    result = copy.deepcopy(board)
    result[action[0]][action[1]] = player(board)
    return result


def winner(board):
    """
    Returns the winner of the game, if there is one.
    """
    # Check rows
    if all(i == board[0][0] for i in board[0]):
        return board[0][0]
    elif all(i == board[1][0] for i in board[1]):
        return board[1][0]
    elif all(i == board[2][0] for i in board[2]):
        return board[2][0]
    # Check columns
    elif board[0][0] == board[1][0] and board[1][0] == board[2][0]:
        return board[0][0]
    elif board[0][1] == board[1][1] and board[1][1] == board[2][1]:
        return board[0][1]
    elif board[0][2] == board[1][2] and board[1][2] == board[2][2]:
        return board[0][2]
    # Check diagonals
    elif board[0][0] == board[1][1] and board[1][1] == board[2][2]:
        return board[0][0]
    elif board[0][2] == board[1][1] and board[1][1] == board[2][0]:
        return board[0][2]
    else:
        return None


def terminal(board):
    """
    Returns True if game is over, False otherwise.
    """

    if winner(board) is not None or (not any(EMPTY in sublist for sublist in
board) and winner(board) is None):
        return True
    else:
        return False
    #return True if winner(board) is not None or (not any(EMPTY in sublist
for sublist in board) and winner(board) is None) else False # noqa E501


def utility(board):
    """
    Returns 1 if X has won the game, -1 if O has won, 0 otherwise.
    """
    if terminal(board):
        if winner(board) == X:
            return 1
        elif winner(board) == O:
            return -1
        else:
            return 0
    # Check how to handle exception when a non terminal board is received.


def minimax(board):
    """
    Returns the optimal action for the current player on the board.
    """
    if terminal(board):
        return None
    else:
        if player(board) == X:
            value, move = max_value(board)
            return move
        else:
            value, move = min_value(board)
            return move


def max_value(board):
    if terminal(board):
        return utility(board), None

    v = float('-inf')
    move = None
    for action in actions(board):
        # v = max(v, min_value(result(board, action)))
        aux, act = min_value(result(board, action))
        if aux > v:
            v = aux
            move = action
            if v == 1:
                return v, move

    return v, move


def min_value(board):
    if terminal(board):
        return utility(board), None

    v = float('inf')
    move = None
    for action in actions(board):
        # v = max(v, min_value(result(board, action)))
        aux, act = max_value(result(board, action))
        if aux < v:
            v = aux
            move = action
            if v == -1:
                return v, move

    return v, move
```

2. Graphic File

```
import pygame

import sys

import time


import tictactoe as ttt


pygame.init()

size = width, height = 600, 400


# Colors

black = (0, 0, 0)

white = (255, 255, 255)
```

```
screen = pygame.display.set_mode(size)


mediumFont = pygame.font.Font("OpenSans-Regular.ttf", 28)

largeFont = pygame.font.Font("OpenSans-Regular.ttf", 40)

moveFont = pygame.font.Font("OpenSans-Regular.ttf", 60)


user = None

board = ttt.initial_state()

ai_turn = False


while True:


    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            sys.exit()


    screen.fill(black)


    # Let user choose a player.

    if user is None:


        # Draw title

        title = largeFont.render("Play Tic-Tac-Toe", True, white)
```

```
titleRect = title.get_rect()
titleRect.center = ((width / 2), 50)
screen.blit(title, titleRect)


# Draw buttons
playXButton = pygame.Rect((width / 8), (height / 2), width / 4, 50)
playX = mediumFont.render("Play as X", True, black)
playXRect = playX.get_rect()
playXRect.center = playXButton.center
pygame.draw.rect(screen, white, playXButton)
screen.blit(playX, playXRect)


playOButton = pygame.Rect(5 * (width / 8), (height / 2), width / 4, 50)
playO = mediumFont.render("Play as O", True, black)
playORect = playO.get_rect()
playORect.center = playOButton.center
pygame.draw.rect(screen, white, playOButton)
screen.blit(playO, playORect)


# Check if button is clicked
click, _, _ = pygame.mouse.get_pressed()
if click == 1:
    mouse = pygame.mouse.get_pos()
    if playXButton.collidepoint(mouse):
```

```
                time.sleep(0.2)
                user = ttt.X
            elif playOButton.collidepoint(mouse):
                time.sleep(0.2)
                user = ttt.O

    else:

        # Draw game board
        tile_size = 80
        tile_origin = (width / 2 - (1.5 * tile_size),
                       height / 2 - (1.5 * tile_size))
        tiles = []
        for i in range(3):
            row = []
            for j in range(3):
                rect = pygame.Rect(
                    tile_origin[0] + j * tile_size,
                    tile_origin[1] + i * tile_size,
                    tile_size, tile_size
                )
                pygame.draw.rect(screen, white, rect, 3)

                if board[i][j] != ttt.EMPTY:
```

```
            move = moveFont.render(board[i][j], True, white)
            moveRect = move.get_rect()
            moveRect.center = rect.center
            screen.blit(move, moveRect)
        row.append(rect)
    tiles.append(row)

game_over = ttt.terminal(board)
player = ttt.player(board)

# Show title
if game_over:
    winner = ttt.winner(board)
    if winner is None:
        title = f"Game Over: Tie."
    else:
        title = f"Game Over: {winner} wins."
elif user == player:
    title = f"Play as {user}"
else:
    title = f"Computer thinking..."
title = largeFont.render(title, True, white)
titleRect = title.get_rect()
titleRect.center = ((width / 2), 30)
```

```
screen.blit(title, titleRect)


# Check for AI move

if user != player and not game_over:

    if ai_turn:

        time.sleep(0.5)

        move = ttt.minimax(board)

        board = ttt.result(board, move)

        ai_turn = False

    else:

        ai_turn = True


# Check for a user move

click, _, _ = pygame.mouse.get_pressed()

if click == 1 and user == player and not game_over:

    mouse = pygame.mouse.get_pos()

    for i in range(3):

        for j in range(3):

            if (board[i][j] == ttt.EMPTY and tiles[i][j].collidepoint(mouse)):

                board = ttt.result(board, (i, j))


if game_over:

    againButton = pygame.Rect(width / 3, height - 65, width / 3, 50)

    again = mediumFont.render("Play Again", True, black)
```

```
        againRect = again.get_rect()

        againRect.center = againButton.center

        pygame.draw.rect(screen, white, againButton)

        screen.blit(again, againRect)

        click, _, _ = pygame.mouse.get_pressed()

        if click == 1:

            mouse = pygame.mouse.get_pos()

            if againButton.collidepoint(mouse):

                time.sleep(0.2)

                user = None

                board = ttt.initial_state()

                ai_turn = False


    pygame.display.flip()
```

5. CONCLUSION

In conclusion, the timeless appeal of the Tic-Tac-Toe game endures as it continues to captivate players across generations. Despite its apparent simplicity, the strategies employed in various algorithms by developers converge on a common objective: thwarting the opponent's path to victory. Each developer infuses their unique style into the algorithmic approach, adding to the diversity of gameplay experiences.

Drawing inspiration from Newell and Simon's seminal work in 1972 on Tic-Tac-Toe, it becomes evident that an effective algorithm encompasses a comprehensive understanding of all facets of the game. By embracing the foundational principles outlined by Newell and Simon, developers can create algorithms that not only block the opponent effectively but also contribute to a more strategic and engaging gameplay experience.

As the Tic-Tac-Toe legacy persists, developers play a pivotal role in refining and innovating algorithms, ensuring the game remains a compelling and intellectually stimulating pursuit. Through continuous exploration and adaptation of algorithmic strategies, the classic Tic-Tac-Toe game retains its relevance in the ever-evolving landscape of game development.

6 . REFERENCES

[1] P. Wensink, The History of Tic-Tac-Toe is Pretty Fascinating Stuff, [Online].

[2] M. J. P. Wolf, Encyclopaedia of Video Games: The Culture, Technology, and Art of Gaming, Greenwood Publishing Group, 2012.

[3] T. Bolon, How to never lose at Tic-Tac-Toe, BookCountry, 2013.

[4] A. Abdolsaheb, How to Make your Tic Tac Toe Game Unbeatable by using the Minimax Algorithm, Feb. 2017. [Online].

[5] https://www.javatpoint.com/tic-tac-toe-game