GOTIAS UNIVERSITY

Plot No.2, Sector -17 A, Yamuna Expressway,

Greater Noida, Gautam Buddh Nagar, U.P., India

SCHOOL OF COMPUTING SCIENCE & ENGINEERING

COURSE NAME: Advanced Data Structures And Algorithms	SEMESTER: V
COURSE CODE: E2UC503C	SECTION: 4
BRANCH: B.Tech CSE	SESSION: 2022-2023

SUBMITTED BY-		SUBMITTED TO-
NAME	ADMISSION NO.	Dr. Vipul Narayan
Aida Sharon Bruce	21SCSE1011272	
Nikhil Kumar	21SCSE1010832	

A Project Report On
TIC-TAC-TOE GAME

Objective:

Tic-Tac-Toe is a two-player game where the goal is to get three of your marks (either "X" or "O") in a row, either horizontally, vertically, or diagonally, on a 3x3 grid. Our aim to is to create this game using python programming language.

Pre-Requisites:

Python

Platform:

Jupyter Notebook from Anaconda Navigator

Libraries Used:

- Numpy: It is a set of python modules that are designed for writing video games.
- Pygame: It is the fundamental package used for scientific computing in python.
- Math: This module helps in performing mathematical operations on numbers.
- Sys: It provides information about functions, methods and constants of the python interpreter.

Game Setup:

- The game is played on a 3x3 grid.
- Two players take turns marking an empty cell with their respective symbols.
- The first player to get three of their symbols in a row wins the game.
- If the grid is filled, and no player has three in a row, the game is a draw.

Game Flow:

1. Start: The game starts with an empty grid.
2. Players' Turns: Players take turns making moves. A player chooses an empty cell and marks it with their symbol.
3. Checking for a Win: After each move, the game checks if the current player has achieved three in a row. This involves checking rows, columns, and diagonals.
4. Checking for a Draw: If the grid is full and no player has three in a row, the game is a draw.
5. End of Game: The game ends when a player wins and the other loses or when it's a draw.

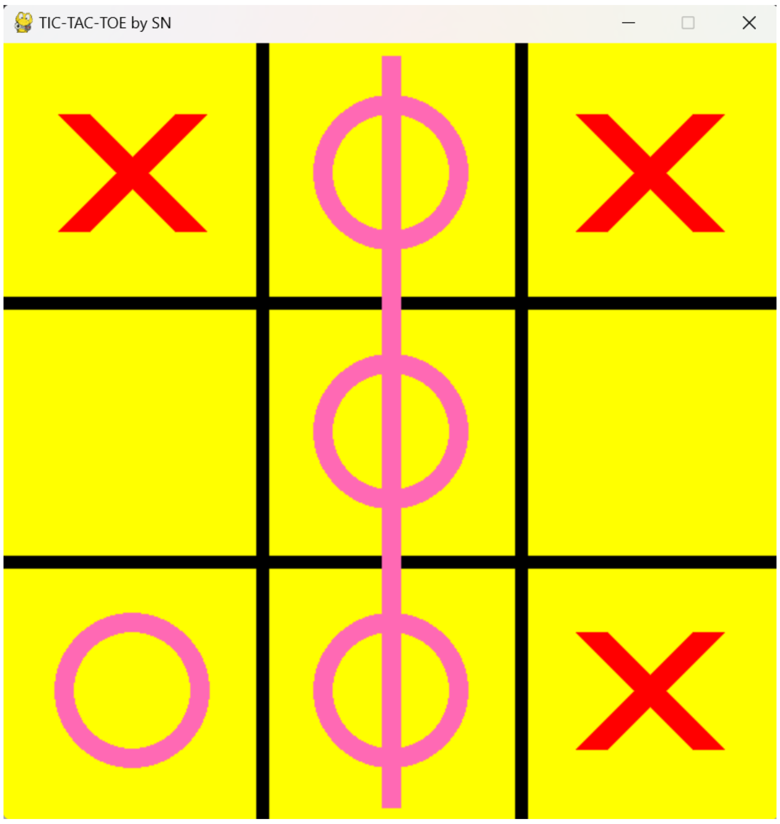
<u>Steps Involved:</u>

1. Install Pygame.
2. Import required modules.
3. Define variables and colors.
4. Initialize game window.
5. Define functions to create game board.
6. Marking the square and check the available square.
7. Check for win or tie.

<u>Source Code:</u>

```python
!pip install pygame
import numpy as np
import pygame
import math
import sys
Rows = 3
Columns = 3
sizeofsquare = 200
radiusofcircle = 60
offset = 55
circlelinewidth= 15
xlinewidth = 25
widthofscreen = Columns * sizeofsquare
heightofscreen = Rows * sizeofsquare
colorofline = (0,0,0)
backgroundcolour = (255,255,0)
colorofcircle= (255,105,180)
xcolor = (255,0,0)

def printingboard():
    flippedboard = np.flip(Board, 0)
    print(flippedboard)
    print("")

def drawboard():
    drawlines()
    drawfigures()

def drawlines():
    pygame.draw.line(Screen, colorofline, (0, 200), (600, 200), 10)
    pygame.draw.line(Screen, colorofline, (0, 400), (600, 400), 10)
    pygame.draw.line(Screen, colorofline, (200, 0), (200, 600), 10)
    pygame.draw.line(Screen, colorofline, (400, 0), (400, 600), 10)

def drawfigures():
    for col in range(Columns):
        for row in range(Rows):
            if Board[row][col] == 1:
                pygame.draw.circle(Screen, colorofcircle, (int(col *
sizeofsquare + sizeofsquare / 2), int(row * sizeofsquare +
sizeofsquare / 2)), radiusofcircle, circlelinewidth)
            elif Board[row][col] == 2:
                pygame.draw.line(Screen, xcolor, (col * sizeofsquare
+ offset, row * sizeofsquare + offset), (col * sizeofsquare +
sizeofsquare - offset, row *sizeofsquare + sizeofsquare - offset),
xlinewidth)
                pygame.draw.line(Screen, xcolor, (col * sizeofsquare
+ offset, row * sizeofsquare + sizeofsquare - offset), (col *
sizeofsquare + sizeofsquare - offset, row * sizeofsquare + offset),
xlinewidth)

def fullboard():
    for col in range(Columns):
        for row in range(Rows):
            if Board[row][col] == 0:
                return False
    return True

def availablesquare(row, col):
    is_available = Board[row][col] == 0
    return is_available

def marksquare(row, col, player):
    Board[row][col] = player

def win(player):
    verwin = verticalwin(player)
    horwin = horizontalwin(player)
    diagwin = diagonalwin(player)
    if verwin or horwin or diagwin:
        return True
    else:
        return False

def verticalwin(player):
    for col in range(Columns):
        if Board[0][col] == player and Board[1][col] == player and
Board[2][col] == player:
            drawverticalline(col, player)
            return True
    return False

def horizontalwin(player):
    for row in range(Rows):
        if Board[row][0] == player and Board[row][1] == player and 
Board[row][2] == player:
            drawhorizontalline(row, player)
            return True
    return False

def diagonalwin(player):
    if Board[0][0] == player and Board[1][1] == player and 
Board[2][2] == player:
        drawdiagonalline(player)
        return True
    elif Board[2][0] == player and Board[1][1] == player and 
Board[0][2] == player:
        drawdiagonalline(player, False)
        return True
    else:
        return False

def drawverticalline(col, player):
    posX = col * sizeofsquare + sizeofsquare / 2
    if player == 1:
        pygame.draw.line(Screen, colorofcircle, (posX, 10), (posX, 
heightofscreen - 10), circlelinewidth)
    else:
        pygame.draw.line(Screen,xcolor, (posX, 10), (posX, 
heightofscreen - 10), circlelinewidth)

def drawhorizontalline(row, player):
    posY = row * sizeofsquare + sizeofsquare/ 2
    if player == 1:
        pygame.draw.line(Screen, colorofcircle, (10, posY), 
(widthofscreen - 10, posY), circlelinewidth)
    else:
        pygame.draw.line(Screen, xcolor, (10, posY), (widthofscreen 
- 10, posY), circlelinewidth)

def drawdiagonalline(player, down_diag=True):
    if down_diag:
        if player == 1:
            pygame.draw.line(Screen, colorofcircle, (25, 25), 
(widthofscreen - 25, heightofscreen - 25), xlinewidth)
        else:
            pygame.draw.line(Screen, colorofcircle, (25, 25), 
(widthofscreen - 25, heightofscreen - 25), xlinewidth)
    else:
        if player == 1:
            pygame.draw.line(Screen, colorofcircle, (25,
heightofscreen - 25), (widthofscreen - 25, 25), xlinewidth)
        else:
            pygame.draw.line(Screen, xcolor, (25, heightofscreen -
25), (widthofscreen - 25, 25), xlinewidth)

Board = np.zeros((Rows,Columns))
pygame.init()
pygame.display.set_caption("TIC-TAC-TOE by SN")
Screen = pygame.display.set_mode((widthofscreen, heightofscreen))
Screen.fill(backgroundcolour)
drawlines()
pygame.display.update()
player = 0
gameover = False
inmenu = True

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.MOUSEBUTTONDOWN and not gameover:
            positiony = event.pos[1]
            row = int(math.floor(positiony / sizeofsquare))
            positionx = event.pos[0]
            col = int(math.floor(positionx / sizeofsquare))

            if player % 2 == 0:
                if availablesquare(row, col):
                    marksquare(row, col, 1)
                    if win(1):
                        gameover = True
                    player += 1

            else:
                if availablesquare(row, col):
                    marksquare(row, col, 2)
                    if win(2):
                        gameover = True
                   player += 1

           if fullboard():
                gameover = True

    drawfigures()
    pygame.display.update()
```

Sample Output:


TIC-TAC-TOE by SN