Tic Tac Toe – About the Python Project

The interesting Python project will be built using the pygame library. We will be explaining all the pygame object methods that are used in this project. Pygame is a great library that will allow us to create the window and draw images and shapes on the window. This way we will capture mouse coordinates and identify the block where we need to mark 'X' or 'O'. Then we will check if the user wins the game or not.

Prerequisites

To implement this game, we will use the basic concepts of Python and Pygame which is a Python library for building cross-platform games. It contains the modules needed for computer graphics and sound libraries. To install the library, you can use pip installer from the command line:

```
pip install pygame
```

Steps to Build a Python Project – Tic Tac Toe Game

First, let's check the steps to build Tic Tac Toe program in Python:

- Create the display window for our game.
- Draw the grid on the canvas where we will play Tic Tac Toe.
- Draw the status bar below the canvas to show which player's turn is it and who wins the game.
- When someone wins the game or the game is a draw then we reset the game.

We need to run our game inside an infinite loop. It will continuously look for events and when a user presses the mouse button on the grid we will first get the X and Y coordinates of the mouse. Then we will check which square the

user has clicked. Then we will draw the appropriate ‘X’ or ‘O’ image on the canvas. So that is basically what we will do in this Python project idea.

1. Initializing game components

So let’s start by importing the pygame library and the time library because we will use the time.sleep() method to pause the game at certain positions. Then we initialize all the global variables that we will use in our Tic Tac Toe game.

```
import pygame as pg,sys
from pygame.locals import *
import time
#initialize global variables
XO = 'x'
winner = None
draw = False
width = 400
height = 400
white = (255, 255, 255)
line_color = (10,10,10)
#TicTacToe 3x3 board
TTT = [[None]*3,[None]*3,[None]*3]
```

Here, the TTT is the main 3×3 Tic Tac Toe board and at first, it will have 9 None values. The height and width of the canvas where we will play the game is 400×400.

2. Initializing Pygame window

We use the pygame to create a new window where we’ll play our Tic Tac Toe game. So we initialize the pygame with pg.init() method and the window display is set with a width of 400 and a height of 500. We have reserved 100-pixel space for displaying the status of the game.

The pg.display.set_mode() initializes the display and we reference it with the screen variable. This screen variable will be used whenever we want to draw something on the display.

The pg.display.set_caption method is used to set a name that will appear at the top of the display window.

```
#initializing pygame window
pg.init()
fps = 30
CLOCK = pg.time.Clock()
screen = pg.display.set_mode((width, height+100),0,32)
pg.display.set_caption("Tic Tac Toe")
```

3. Load and transform images

The Python machine learning project uses many images like the opening image that will display when the game starts or resets. The X and O images that we will draw when the user clicks on the grid. We load all the images and resize them so that they will fit easily in our window.

```
#loading the images
opening = pg.image.load('tic tac opening.png')
x_img = pg.image.load('x.png')
o_img = pg.image.load('o.png')
#resizing images
x_img = pg.transform.scale(x_img, (80,80))
o_img = pg.transform.scale(o_img, (80,80))
opening = pg.transform.scale(opening, (width, height+100))
```

4. Define the functions

Now we create a function that will start the game. We will also use this function when we want to restart the game. In pygame, the blit() function is used on the surface to draw an image on top of another image.

So we draw the opening image and after drawing, we always need to update the display with pg.display.update(). When the opening image is drawn, we wait for 1 second using time.sleep(1) and fill the screen with white colour.

Next, we draw 2 vertical and horizontal lines on the white background to make the 3×3 grid. In the end, we call the draw_status() function

```
def game_opening():
    screen.blit(opening,(0,0))
    pg.display.update()
    time.sleep(1)
    screen.fill(white)
   # Drawing vertical lines
    pg.draw.line(screen,line_color,(width/3,0),(width/3, height),7)
    pg.draw.line(screen,line_color,(width/3*2,0),(width/3*2, height),7)
   # Drawing horizontal lines
    pg.draw.line(screen,line_color,(0,height/3),(width, height/3),7)
    pg.draw.line(screen,line_color,(0,height/3*2),(width, height/3*2),7)
    draw_status()
```

The draw_status() function draws a black rectangle where we update the status of the game showing which player's turn is it and whether the game ends or draws.

```
def draw_status():
    global draw
    if winner is None:
        message = XO.upper() + "'s Turn"
    else:
        message = winner.upper() + " won!"
    if draw:
        message = 'Game Draw!'
    font = pg.font.Font(None, 30)
    text = font.render(message, 1, (255, 255, 255))
   # copy the rendered message onto the board
    screen.fill ((0, 0, 0), (0, 400, 500, 100))
    text_rect = text.get_rect(center=(width/2, 500-50))
    screen.blit(text, text_rect)
    pg.display.update()
```

The check_win() function checks the Tic Tac Toe board to see all the marks of 'X' and 'O'. It calculates whether a player has won the game or not. They can either win when the player has marked 3 consecutive marks in a row, column or diagonally. This function is called every time when we draw a mark 'X' or 'O' on the board.

```
def check_win():
    global TTT, winner,draw
```

```
  # check for winning rows
   for row in range (0,3):
       if ((TTT [row][0] == TTT[row][1] == TTT[row][2]) and(TTT [row][0] is not
None)):
          # this row won
           winner = TTT[row][0]
           pg.draw.line(screen, (250,0,0), (0, (row + 1)*height/3 -height/6),\
                              (width, (row + 1)*height/3 - height/6 ), 4)
           break
  # check for winning columns
   for col in range (0, 3):
       if (TTT[0][col] == TTT[1][col] == TTT[2][col]) and (TTT[0][col] is not
None):
          # this column won
           winner = TTT[0][col]
          #draw winning line
           pg.draw.line (screen, (250,0,0),((col + 1)* width/3 - width/6, 0),\
                         ((col + 1)* width/3 - width/6, height), 4)
           break
  # check for diagonal winners
   if (TTT[0][0] == TTT[1][1] == TTT[2][2]) and (TTT[0][0] is not None):
      # game won diagonally left to right
       winner = TTT[0][0]
       pg.draw.line (screen, (250,70,70), (50, 50), (350, 350), 4)
   if (TTT[0][2] == TTT[1][1] == TTT[2][0]) and (TTT[0][2] is not None):
      # game won diagonally right to left
       winner = TTT[0][2]
       pg.draw.line (screen, (250,70,70), (350, 50), (50, 350), 4)
   if(all([all(row) for row in TTT]) and winner is None ):
       draw = True
   draw_status()
```

The drawXO(row, col) function takes the row and column where the mouse is clicked and then it draws the ‘X’ or ‘O’ mark. We calculate the x and y coordinates of the starting point from where we’ll draw the png image of the mark.

```
def drawXO(row,col):
   global TTT,XO
   if row==1:
       posx = 30
   if row==2:
```

```
        posx = width/3 + 30
    if row==3:
        posx = width/3*2 + 30
    if col==1:
        posy = 30
    if col==2:
        posy = height/3 + 30
    if col==3:
        posy = height/3*2 + 30
TTT[row-1][col-1] = XO
    if(XO == 'x'):
        screen.blit(x_img,(posy,posx))
XO= 'o'
else:
        screen.blit(o_img,(posy,posx))
        XO= 'x'
    pg.display.update()
  #print(posx,posy)
  #print(TTT)
```

The userClick() function is triggered every time the user presses the mouse button.

When the user clicks the mouse, we first take the x and y coordinates of where the mouse is clicked on the display window and then if that place is not occupied we draw the ‘XO’ on the canvas. We also check if the player wins or not after drawing ‘XO’ on the board.

```
def userClick():
  #get coordinates of mouse click
   x,y = pg.mouse.get_pos()
  #get column of mouse click (1-3)
   if(x<width/3):
        col = 1
   elif (x<width/3*2):
        col = 2
   elif(x<width):
        col = 3
   else:
        col = None
  #get row of mouse click (1-3)
```

```
    if(y<height/3):
        row = 1
    elif (y<height/3*2):
        row = 2
    elif(y<height):
        row = 3
    else:
        row = None
  #print(row,col)
    if(row and col and TTT[row-1][col-1] is None):
        global XO
       #draw the x or o on screen
        drawXO(row,col)
        check_win()
```

The last function is the reset_game(). This will restart the game and we also reset all the variables to the beginning of the game.

```
def reset_game():
    global TTT, winner,XO, draw
    time.sleep(3)
    XO = 'x'
    draw = False
    game_opening()
    winner=None
    TTT = [[None]*3,[None]*3,[None]*3]
```

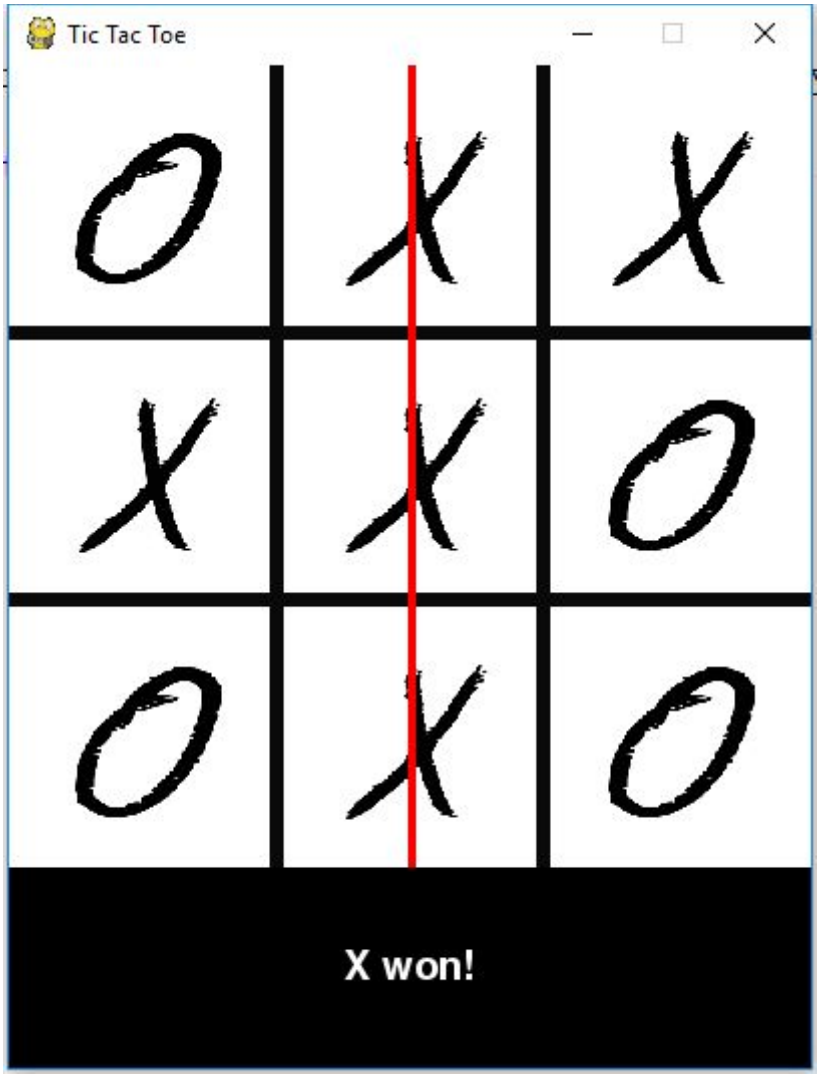
5. Run the game forever

To start the game, we will call the game_opening() function. Then, we run an infinite loop and continuously check for any event made by the user. If the user presses the mouse button, the MOUSEBUTTONDOWN event will be captured and then we will trigger the userClick() function. Then if the user wins or the game draws, we reset the game by calling reset_game() function. We update the display in each iteration and we have set the frames per second to 30.

```
game_opening()
# run the game loop forever
while(True):
    for event in pg.event.get():
        if event.type == QUIT:
```

```
            pg.quit()
            sys.exit()
        elif event.type is MOUSEBUTTONDOWN:
            # the user clicked; place an X or O
            userClick()
            if(winner or draw):
                reset_game()
    pg.display.update()
    CLOCK.tick(fps)
```

Output:



Summary

With this project in Python, we have successfully made the Tic Tac Toe game. We used the popular pygame library for rendering graphics on a display window. We learned how to capture events from the keyboard or mouse and trigger a function when the mouse button is pressed. This way we can calculate mouse position, draw X or O on the display and check if the player wins the game or not. I hope you enjoyed building the game.

Project Link : https://github.com/Pritam-007/Tic-Tac-Toe-Game/tree/master/Game-Tic-Tac-Toe