**TIC TAC TOE**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**Perumalla Dharan – AP21110010201**

**Tarun Teja Kudeti – AP21110010205**

**Vatala Phalgun – AP21110010223**

**Dinesh Grandhi – AP21110010240**

**A picture containing text

Description automatically generated**

Under the Guidance of

**Kavitha Rani Karnena**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**December,2022**

# Certificate

Date: 22-Dec-22

This is to certify that the work present in this Project entitled “**TIC TAC TOE**” has been carried out by **Perumalla Dharan (AP21110010201), Dinesh Grandhi (AP21110010240), Tarun Teja Kudeti (AP21110010205), Vatala Phalgun (AP21110010223)** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in **School of Engineering and Sciences**.

**Supervisor**

(Signature)

Prof. / Dr. **Kavitha Rani Karnena**

Designation,

Affiliation

# Abstract

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.

# Modules Used

1. Initializing Window

#initializing pygame window

pg.init()

fps = 30

#Creates a new Clock object that can be used to track an amount of time

CLOCK = pg.time.Clock()

#returns surface object for the window

screen = pg.display.set\_mode((width, height+100),0,32)

pg.display.set\_caption("Tic Tac Toe")

1. Loading Images

#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')

1. Game Opening

def game\_opening():

    #to copy the contents of one Surface onto another Surface

    screen.blit(opening,(0,0))

    #updates the entire Surface

    pg.display.update()

    #used to add delay in the execution of a program

    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()

1. Draw Status

def draw\_status():

    #allows us to modify the variable outside of the current scope

    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))

    # takes the coordinates to display the message

    text\_rect = text.get\_rect(center=(width/2, 500-50))

    screen.blit(text, text\_rect)

    pg.display.update()

1. Check Win

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()

1. User Click

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()

1. Game Opening

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 == MOUSEBUTTONDOWN:

            # the user clicked; place an X or O

            userClick()

            if(winner or draw):

                reset\_game()

    pg.display.update()

    CLOCK.tick(fps)