```
# Tic Tac Toe game
# Sources I used:
# 1) Pygame documentation: https://www.pygame.org/docs/
# 2) Pygame buttons: https://www.geeksforgeeks.org/how-to-create-buttons-in-a-
game-using-pygame/

import math
import pygame

# Sets the window size
WIDTH = 400
HEIGHT = 400
WINDOW_SIZE = (WIDTH, HEIGHT)

# Stores the centers of each box as 2d list/grid of (x, y) tuples
CENTERS = [[(WINDOW_SIZE[0]/6, WINDOW_SIZE[1]/6), (3 * WINDOW_SIZE[0]/6,
WINDOW_SIZE[1]/6), (5 * WINDOW_SIZE[0]/6, WINDOW_SIZE[1]/6)],
           [(WINDOW_SIZE[0]/6, 3 * WINDOW_SIZE[1]/6), (3 * WINDOW_SIZE[0]/6, 3 *
WINDOW_SIZE[1]/6), (5 * WINDOW_SIZE[0]/6, 3 * WINDOW_SIZE[1]/6)],
           [(WINDOW_SIZE[0]/6, 5 * WINDOW_SIZE[1]/6), (3 * WINDOW_SIZE[0]/6, 5 *
WINDOW_SIZE[1]/6), (5 * WINDOW_SIZE[0]/6, 5 * WINDOW_SIZE[1]/6)]]


# The function returns a list that stores '' values
def get_empty_grid():
    return [[' ', ' ', ' '],
            [' ', ' ', ' '],
            [' ', ' ', ' ']]



def display_grid(grid):
    # prints the row in each grid to dislay the entire grid
    print()
    for r in grid:
        print(r)
    print()

def check_rows(grid):
    is_winning = False
    # checks the conditions for each row in the grid
    for r in grid:
        if r in (['X', 'X', 'X'], ['O', 'O', 'O']):
            is_winning = True
            break
        else:
            is_winning = False
    return is_winning



def check_diaganols(grid):
    if grid[0][0] == grid[1][1] == grid[2][2] == "X" or grid[0][0] == grid[1][1]
```

```
== grid[2][2] == "O":
        return True
    if grid[0][2] == grid[1][1] == grid[2][0] == "X" or grid[0][2] == grid[1][1]
== grid[2][0] == "O":
        return True
    return False


def check_columns(grid):
    for i in range(0, 3):
        if grid[0][i] == grid[1][i] == grid[2][i] == "X" or grid[0][i] == grid[1]
[i] == grid[2][i] == "O":
            return True
    return False

# function takes in operator and grid; returns True if there's a winning
combination else false.
def  check_win(grid):
    if check_rows(grid) or check_diaganols(grid) or check_columns(grid):
        return True
    return False


def check_draw(grid):
    # checks if each row in the grid doesn't contain an operator (is empty)
    for row in grid:
        if ' ' in row:
            return False
    return True

def init_game():
    # Initialize Pygame
    pygame.init()

    # Creates the window
    screen = pygame.display.set_mode(WINDOW_SIZE)
    screen.fill((255, 255, 255))
    pygame.font.init()
    initial_font = pygame.font.SysFont('Comic Sans MS', 17)
    # Renders the text surface with the specific font, text, and color.
    initial_text_surface = initial_font.render('Click to Start!', True, (255, 63,
0))
    initialtwo_text_surface = initial_font.render('Player X is First', True, (35,
181, 211))
    initialText = initial_text_surface.get_rect()
    initialTwo = initialtwo_text_surface.get_rect()
    X = 400
    Y = 400
    initialText.center = (X // 2, Y // 2)
    initialTwo.center = (X // 2, Y // 1.7)
    # Blits both texts on to the screen
    screen.blit(initial_text_surface, initialText)
    screen.blit(initialtwo_text_surface, initialTwo )
    pygame.display.set_caption("Tic-Tac-Toe")
    game_loop(screen)

    return screen


def display_text(screen, text):
    screen.fill((255, 255, 255))
    pygame.font.init()
    result_font = pygame.font.SysFont('Comic Sans MS', 50)
    options_font = pygame.font.SysFont('Comic Sans MS', 30)

    # Renders the text surface with the specific font, text, and color.
    result_text_surface = result_font.render(text, True, (189, 99, 47))
    quit_text_surface = options_font.render("Quit", True, (235, 130, 176))
    reset_text_surface = options_font.render("Reset", True, (90, 193, 177))

    # Gets the rectangle for the result text surface
    resultText = result_text_surface.get_rect()
    X = 400
    Y = 400
    resultText.center = (X // 2, Y // 2)
    screen.blit(result_text_surface, resultText)
    screen.blit(quit_text_surface, (25, 295))
    screen.blit(reset_text_surface, (290, 295))
    pygame.display.update()

def display_new(screen):
    screen.fill((255,255,255))


def game_loop(screen):
    grid = get_empty_grid()
    running = True
    operator = "X"
    while running:
        text_msg = ''
        if check_win(grid):
            # at this point, the operator will already be alternated
            winner = "O" if operator == "X" else "X"
            text_msg = 'Player ' + winner + ' Won!'
        elif check_draw(grid):
            text_msg = 'Draw'

        # if there is a text message, it resets the grid to an empty one
        # and displays the specific text message
        if text_msg != '':
            display_text(screen, text_msg)
            for event in pygame.event.get():
                if event.type == pygame.MOUSEBUTTONDOWN:
                    mouse = pygame.mouse.get_pos()

                    # Checks if the mouse is clicked between these values (on the
Quit button)
                    if 16<=mouse[0]<=116 and 300<=mouse[1]<=350:
                        exit()
                    else:
                        # If mouse is not clicked on the Quit button
                        # then there's a new grid on the screen.
                        grid = get_empty_grid()
                        display_new(screen)
            continue


        # operator = "O" if operator == "X" else "X"

        i = 1
        while i < 3:

            # Displays the grid lines
            pygame.draw.line(screen, (0, 0, 0), (i * WINDOW_SIZE[0]/3, 0), (i *
WINDOW_SIZE[0]/3, WINDOW_SIZE[1]), 3)
            pygame.draw.line(screen, (0, 0, 0), (0, i * WINDOW_SIZE[1]/3),
(WINDOW_SIZE[0], i * WINDOW_SIZE[1]/3), 3)
            i += 1

        pygame.display.flip()


        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            elif event.type == pygame.MOUSEBUTTONDOWN:
                # Gets the position of the mouse
                mouse1 = pygame.mouse.get_pos()
                print(mouse1)
                # If the coordinates of the click are between the specified
values, it executes the if condition.
                if 160 <=mouse1[0] <=250 and 180 <=mouse1[1] <=200:
                    screen.fill((255, 255, 255))
                x, y = pygame.mouse.get_pos()
                user_input = (x, y)
                # Sets the minimum point and minimum distance to the first
                # center point in the list.
                min_point = CENTERS[0][0]
                min_dist = math.dist(user_input, min_point)

                # Loops through the center points and each specific point
                for c in CENTERS:
                    for j in c:
                        # Calculates the distance between the user input
coordinates
                        # and the current point
                        calc_dist = math.dist(user_input, j)
                        if calc_dist < min_dist:

                            # Updates the minimum point and distance
                            min_point = j
                            min_dist = calc_dist
                x1, y1 = min_point

                if operator == "O":
                    pygame.draw.circle(screen, (60, 219, 211),
                                            min_point, 60, width=7)
                else:

                    # stores the coordinates for drawing an two diagnol lines to
make an X
                    x2, y2 = x1 + WIDTH/8, y1 - HEIGHT/8
                    x3, y3 = x1 - WIDTH/8, y1 - HEIGHT/8
                    x4, y4 = x3 + 2*WIDTH/8, y3 + 2*HEIGHT/8
                    x5, y5 = x2 - 2*WIDTH/8, y3 + 2*HEIGHT/8
                    pygame.draw.line(screen, (232, 72, 85), (x3, y3), (x4, y4),
10)
                    pygame.draw.line(screen, (232, 72, 85), (x2, y2), (x5, y5),
10)

                for i, row in enumerate(CENTERS):
                    for j, element in enumerate(row):
                        if element == min_point:
                            index_x = i
                            index_y = j
                            break

                # puts the operator in the desired spot in the grid
                grid[index_x][index_y] = operator

                operator = "O" if operator == "X" else "X"

                display_grid(grid)

# Checks if the current file is being run as the main program
if __name__ == '__main__':
    screen = init_game()
    game_loop(screen)
    pygame.quit()
```