import pygame

import sys

import numpy as np

# Initialize Pygame

pygame.init()

# Screen dimensions

WIDTH = 600

HEIGHT = 600

LINE\_WIDTH = 15

BOARD\_ROWS = 3

BOARD\_COLS = 3

SQUARE\_SIZE = WIDTH // BOARD\_COLS

CIRCLE\_RADIUS = SQUARE\_SIZE // 3

CIRCLE\_WIDTH = 15

CROSS\_WIDTH = 25

SPACE = SQUARE\_SIZE // 4

# Colors

RED = (255, 0, 0)

BG\_COLOR = (28, 170, 156)

LINE\_COLOR = (23, 145, 135)

CIRCLE\_COLOR = (239, 231, 200)

CROSS\_COLOR = (66, 66, 66)

# Set up the screen

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption('TIC TAC TOE')

screen.fill(BG\_COLOR)

# Board

board = np.zeros((BOARD\_ROWS, BOARD\_COLS))

def draw\_lines():

    # Horizontal lines

    pygame.draw.line(screen, LINE\_COLOR, (0, SQUARE\_SIZE), (WIDTH, SQUARE\_SIZE), LINE\_WIDTH)

    pygame.draw.line(screen, LINE\_COLOR, (0, 2 \* SQUARE\_SIZE), (WIDTH, 2 \* SQUARE\_SIZE), LINE\_WIDTH)

    # Vertical lines

    pygame.draw.line(screen, LINE\_COLOR, (SQUARE\_SIZE, 0), (SQUARE\_SIZE, HEIGHT), LINE\_WIDTH)

    pygame.draw.line(screen, LINE\_COLOR, (2 \* SQUARE\_SIZE, 0), (2 \* SQUARE\_SIZE, HEIGHT), LINE\_WIDTH)

def draw\_figures():

    for row in range(BOARD\_ROWS):

        for col in range(BOARD\_COLS):

            if board[row][col] == 1:

                pygame.draw.circle(screen, CIRCLE\_COLOR, (int(col \* SQUARE\_SIZE + SQUARE\_SIZE // 2), int(row \* SQUARE\_SIZE + SQUARE\_SIZE // 2)), CIRCLE\_RADIUS, CIRCLE\_WIDTH)

            elif board[row][col] == 2:

                pygame.draw.line(screen, CROSS\_COLOR, (col \* SQUARE\_SIZE + SPACE, row \* SQUARE\_SIZE + SQUARE\_SIZE - SPACE), (col \* SQUARE\_SIZE + SQUARE\_SIZE - SPACE, row \* SQUARE\_SIZE + SPACE), CROSS\_WIDTH)

                pygame.draw.line(screen, CROSS\_COLOR, (col \* SQUARE\_SIZE + SPACE, row \* SQUARE\_SIZE + SPACE), (col \* SQUARE\_SIZE + SQUARE\_SIZE - SPACE, row \* SQUARE\_SIZE + SQUARE\_SIZE - SPACE), CROSS\_WIDTH)

def mark\_square(row, col, player):

    board[row][col] = player

def available\_square(row, col):

    return board[row][col] == 0

def is\_board\_full():

    for row in range(BOARD\_ROWS):

        for col in range(BOARD\_COLS):

            if board[row][col] == 0:

                return False

    return True

def check\_win(player):

    # Vertical win check

    for col in range(BOARD\_COLS):

        if board[0][col] == player and board[1][col] == player and board[2][col] == player:

            draw\_vertical\_winning\_line(col, player)

            return True

    # Horizontal win check

    for row in range(BOARD\_ROWS):

        if board[row][0] == player and board[row][1] == player and board[row][2] == player:

            draw\_horizontal\_winning\_line(row, player)

            return True

    # Ascending diagonal win check

    if board[2][0] == player and board[1][1] == player and board[0][2] == player:

        draw\_asc\_diagonal(player)

        return True

    # Descending diagonal win check

    if board[0][0] == player and board[1][1] == player and board[2][2] == player:

        draw\_desc\_diagonal(player)

        return True

    return False

def draw\_vertical\_winning\_line(col, player):

    posX = col \* SQUARE\_SIZE + SQUARE\_SIZE // 2

    if player == 1:

        color = CIRCLE\_COLOR

    elif player == 2:

        color = CROSS\_COLOR

    pygame.draw.line(screen, color, (posX, 15), (posX, HEIGHT - 15), LINE\_WIDTH)

def draw\_horizontal\_winning\_line(row, player):

    posY = row \* SQUARE\_SIZE + SQUARE\_SIZE // 2

    if player == 1:

        color = CIRCLE\_COLOR

    elif player == 2:

        color = CROSS\_COLOR

    pygame.draw.line(screen, color, (15, posY), (WIDTH - 15, posY), LINE\_WIDTH)

def draw\_asc\_diagonal(player):

    if player == 1:

        color = CIRCLE\_COLOR

    elif player == 2:

        color = CROSS\_COLOR

    pygame.draw.line(screen, color, (15, HEIGHT - 15), (WIDTH - 15, 15), LINE\_WIDTH)

def draw\_desc\_diagonal(player):

    if player == 1:

        color = CIRCLE\_COLOR

    elif player == 2:

        color = CROSS\_COLOR

    pygame.draw.line(screen, color, (15, 15), (WIDTH - 15, HEIGHT - 15), LINE\_WIDTH)

def restart():

    screen.fill(BG\_COLOR)

    draw\_lines()

    for row in range(BOARD\_ROWS):

        for col in range(BOARD\_COLS):

            board[row][col] = 0

def ai\_move():

    for row in range(BOARD\_ROWS):

        for col in range(BOARD\_COLS):

            if available\_square(row, col):

                board[row][col] = 2

                if check\_win(2):

                    return

                board[row][col] = 0

    for row in range(BOARD\_ROWS):

        for col in range(BOARD\_COLS):

            if available\_square(row, col):

                board[row][col] = 1

                if check\_win(1):

                    board[row][col] = 2

                    return

                board[row][col] = 0

    for row in range(BOARD\_ROWS):

        for col in range(BOARD\_COLS):

            if available\_square(row, col):

                board[row][col] = 2

                return

draw\_lines()

player = 1

game\_over = False

# Main loop

while True:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            pygame.quit()

            sys.exit()

        if event.type == pygame.MOUSEBUTTONDOWN and not game\_over:

            mouseX = event.pos[0]

            mouseY = event.pos[1]

            clicked\_row = mouseY // SQUARE\_SIZE

            clicked\_col = mouseX // SQUARE\_SIZE

            if available\_square(clicked\_row, clicked\_col):

                if player == 1:

                    mark\_square(clicked\_row, clicked\_col, 1)

                    if check\_win(player):

                        game\_over = True

                    player = 2

                    ai\_move()

                    if check\_win(player):

                        game\_over = True

                    player = 1

                draw\_figures()

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_r:

                restart()

                player = 1

                game\_over = False

    pygame.display.update()