Guide to a Simple Pygame Tic-Tac-Toe

Jack Massey

Overview

This guide explains a minimalistic Tic-Tac-Toe game built using `pygame`, a Python library for creating simple games. It uses a 3x3 grid, alternating turns for X and O, and detects winning and draw conditions.

Requirements

- Python 3 installed on your system
- Pygame: install using `pip install pygame`
- Compatible with Windows, macOS, and Linux

Code Breakdown

Setup and Constants

We import libraries, initialize pygame, and define constants like screen size, colors, and fonts.

```
import pygame
import sys

pygame.init()

WIDTH, HEIGHT = 300, 300
LINE_WIDTH = 5
CELL_SIZE = WIDTH // 3
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
```

Display and State

We create the game window and set up the initial board and current player.

```
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Tic Tac Toe")

board = [["" for _ in range(3)] for _ in range(3)]
current_player = "X"
game_over = False
```

Drawing the Board

The `draw_board` function draws the grid and displays each mark (X or O) on the screen.

```
def draw_board():
    screen.fill(WHITE)
    for i in range(1, 3):
        pygame.draw.line(screen, BLACK, (CELL_SIZE * i, 0), (CELL_SIZE * i, HEIGHT), LINE_WIDTH
    )
        pygame.draw.line(screen, BLACK, (0, CELL_SIZE * i), (WIDTH, CELL_SIZE * i), LINE_WIDTH)

    for row in range(3):
        for col in range(3):
            mark = board[row][col]
            if mark:
                text = font.render(mark, True, BLACK)
                rect = text.get_rect(center=(col * CELL_SIZE + CELL_SIZE // 2, row * CELL_SIZE
    + CELL_SIZE // 2))
                screen.blit(text, rect)
```

Checking for a Winner or Draw

Two functions check for victory or a full board (draw):

```
def check_winner():
    for i in range(3):
        if board[i][0] == board[i][1] == board[i][2] != "":
            return board[i][0]
        if board[0][i] == board[1][i] == board[2][i] != "":
            return board[0][i]
    if board[0][0] == board[1][1] == board[2][2] != "":
        return board[0][0]
    if board[0][2] == board[1][1] == board[2][0] != "":
        return board[0][2]
    return None

def is_draw():
    for row in board:
        if "" in row:
            return False
    return True
```

Displaying a Message

```
def show_message(message):
    text = font.render(message, True, BLACK)
    rect = text.get_rect(center=(WIDTH // 2, HEIGHT // 2))
    screen.blit(text, rect)
    pygame.display.update()
    pygame.time.wait(2000)
```

Main Loop

This is where the game runs: rendering the screen, processing input, and handling logic.

```
while True:
    draw_board()
    pygame.display.update()

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

        if not game_over and event.type == pygame.MOUSEBUTTONDOWN:
            x, y = pygame.mouse.get_pos()
            row = y // CELL_SIZE
            col = x // CELL_SIZE

            if board[row][col] == "":
                board[row][col] = current_player
                winner = check_winner()
                if winner:
                    draw_board()
                    show_message(f"{winner} wins!")
                    game_over = True
                elif is_draw():
                    draw_board()
                    show_message("Draw!")
                    game_over = True
                else:
                    current_player = "O" if current_player == "X" else "X"
```

Suggestions for Extensions

- Add a "Play Again" button or restart key
- Use a class to manage game state
- Add a basic AI for single-player mode
- Display a score counter