Click the Targets - Pygame Game

This document contains the complete source code and explanation for a simple Pygame-based game called 'Click the Targets'. In this game, targets appear on the screen, and the player must click them before they disappear. If the player misses too many targets, the game ends.

import pygame  
import random  
import sys

Import required modules: Pygame for the game engine, random for random target positions, and sys to exit cleanly.

# Initialize Pygame  
pygame.init()

Initialize all Pygame modules.

# Screen setup  
WIDTH, HEIGHT = 600, 400  
screen = pygame.display.set\_mode((WIDTH, HEIGHT))  
pygame.display.set\_caption("Click the Targets")

Set up the screen size and window title.

# Colors  
WHITE = (255, 255, 255)  
RED = (255, 50, 50)  
BLACK = (0, 0, 0)

Define color constants for drawing.

# Clock and font  
clock = pygame.time.Clock()  
font = pygame.font.SysFont("comicsans", 24)  
FPS = 60

Set up the game clock to control the frame rate and a font for displaying text.

# Game variables  
score = 0  
missed = 0  
max\_missed = 10  
target\_radius = 25  
target\_duration = 1500 # milliseconds

Initialize gameplay variables such as score, number of misses, maximum allowed misses, target radius, and how long each target stays.

# Generate target as (x, y) center  
def new\_target():  
 x = random.randint(target\_radius, WIDTH - target\_radius)  
 y = random.randint(target\_radius, HEIGHT - target\_radius)  
 return (x, y)

Function that generates a new target at a random position on the screen, ensuring it doesn't go out of bounds.

target\_pos = new\_target()  
target\_spawn\_time = pygame.time.get\_ticks()

Create the first target and record the time it was created.

# Game loop  
running = True  
while running:  
 screen.fill(WHITE)  
 current\_time = pygame.time.get\_ticks()

Start the main game loop, fill the screen with white each frame, and get the current time to track target durations.

for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 running = False

Check for quit events (window close).

# Mouse click  
 if event.type == pygame.MOUSEBUTTONDOWN:  
 mx, my = pygame.mouse.get\_pos()  
 dx = mx - target\_pos[0]  
 dy = my - target\_pos[1]  
 if dx\*\*2 + dy\*\*2 <= target\_radius\*\*2:  
 score += 1  
 target\_pos = new\_target()  
 target\_spawn\_time = current\_time  
 else:  
 missed += 1

Handle mouse click events. If the click is inside the circle, count as a hit. Otherwise, count as a miss.

# Timeout check  
 if current\_time - target\_spawn\_time > target\_duration:  
 missed += 1  
 target\_pos = new\_target()  
 target\_spawn\_time = current\_time

If the player doesn't click in time, it's counted as a miss, and a new target is generated.

# Draw target  
 pygame.draw.circle(screen, RED, target\_pos, target\_radius)

Draw the red circular target on the screen.

# Draw score  
 score\_text = font.render(f"Score: {score}", True, BLACK)  
 missed\_text = font.render(f"Missed: {missed}/{max\_missed}", True, BLACK)  
 screen.blit(score\_text, (10, 10))  
 screen.blit(missed\_text, (10, 40))

Render and display the score and missed count on the screen.

# Game over  
 if missed >= max\_missed:  
 game\_over\_text = font.render(f"Game Over! Final Score: {score}", True, BLACK)  
 screen.blit(game\_over\_text, (WIDTH // 2 - 120, HEIGHT // 2))  
 pygame.display.flip()  
 pygame.time.delay(3000)  
 break

If the player has missed too many times, display a game over message and exit the loop.

pygame.display.flip()  
 clock.tick(FPS)

Update the display and cap the frame rate.

pygame.quit()  
sys.exit()

Cleanly exit the game.