Catch the Falling Blocks - Pygame Game

import pygame  
import random  
import sys

Import required modules: pygame for game development, random for randomness, and sys for system exit.

# Initialize Pygame  
pygame.init()

Initialize all imported Pygame modules.

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

Set up the game window with a width of 600 and height of 400, and set its title.

# Colors  
WHITE = (255, 255, 255)  
BLUE = (50, 100, 255)  
RED = (255, 50, 50)  
BLACK = (0, 0, 0)

Define RGB color values for later use in the game.

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

Create a clock to manage frame rate and a font for rendering text.

# Paddle  
paddle\_width = 100  
paddle\_height = 15  
paddle\_y = HEIGHT - 30

Set dimensions and position of the player-controlled paddle.

# Block  
block\_width = 30  
block\_height = 30  
block\_speed = 5

Define the dimensions and falling speed of the red blocks.

# Game variables  
score = 0  
total\_blocks = 0  
max\_blocks = 100

Initialize the game state variables including score and block counters.

# Generate first block  
def new\_block():  
 x = random.randint(0, WIDTH - block\_width)  
 return pygame.Rect(x, -block\_height, block\_width, block\_height)  
  
block = new\_block()

Create a function to spawn a new block at a random horizontal position, and generate the first block.

# Main loop  
running = True  
while running:  
 screen.fill(WHITE)  
   
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 running = False

Start the game loop. Handle the quit event and fill the screen with white color.

# Paddle movement  
 mouse\_x = pygame.mouse.get\_pos()[0]  
 paddle\_x = mouse\_x - paddle\_width // 2  
 paddle\_x = max(0, min(WIDTH - paddle\_width, paddle\_x))  
 paddle = pygame.Rect(paddle\_x, paddle\_y, paddle\_width, paddle\_height)

Move the paddle based on the mouse's X position, ensuring it stays within screen bounds.

# Move block  
 block.y += block\_speed

Move the falling block downward at a constant speed.

# Check for collision  
 if block.colliderect(paddle):  
 score += 1  
 total\_blocks += 1  
 block = new\_block()

If the block touches the paddle, increase score and total block count, then spawn a new block.

# Block missed  
 elif block.y > HEIGHT:  
 total\_blocks += 1  
 block = new\_block()

If the block falls past the screen, count it as missed and generate a new one.

# Draw paddle and block  
 pygame.draw.rect(screen, BLUE, paddle)  
 pygame.draw.rect(screen, RED, block)

Render the paddle and block on the screen.

# Draw score and progress  
 score\_text = font.render(f"Score: {score}", True, BLACK)  
 blocks\_left\_text = font.render(f"Blocks: {total\_blocks}/{max\_blocks}", True, BLACK)  
 screen.blit(score\_text, (10, 10))  
 screen.blit(blocks\_left\_text, (10, 40))

Display the current score and number of blocks processed.

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

If the total number of blocks reaches 100, show the game over message and delay before exiting.

pygame.display.flip()  
 clock.tick(60)

Update the display and maintain a frame rate of 60 FPS.

pygame.quit()  
sys.exit()

Quit Pygame and exit the program.