import pygame

import random

# Initialize Pygame

pygame.init()

# Constants

SCREEN\_WIDTH, SCREEN\_HEIGHT = 800, 600

BRICK\_WIDTH = SCREEN\_WIDTH // 16

BRICK\_HEIGHT = 20

NUM\_BRICKS\_PER\_ROW = SCREEN\_WIDTH // BRICK\_WIDTH

NUM\_ROWS\_OF\_BRICKS = 8

PADDLE\_WIDTH, PADDLE\_HEIGHT = 100, 20

BALL\_RADIUS = 10

GRAY = (200, 200, 200)

GREEN = (0, 255, 0)

WHITE = (255, 255, 255)

RED = (255, 0, 0)

YELLOW = (255, 255, 0)

BRICK\_POINTS = 10

POWER\_UP\_CHANCE = 0.1  # 10% chance for a power-up when a brick is destroyed

# Set up display

screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT))

pygame.display.set\_caption("Breakout Game")

font = pygame.font.Font(None, 36)

# Functions and Classes

class Brick:

    def \_\_init\_\_(self, x, y):

        self.rect = pygame.Rect(x, y, BRICK\_WIDTH - 2, BRICK\_HEIGHT - 2)  # Small gaps between bricks

        self.color = GRAY

    def draw(self):

        pygame.draw.rect(screen, self.color, self.rect)

def create\_bricks():

    return [Brick(x \* BRICK\_WIDTH, y \* BRICK\_HEIGHT) for y in range(NUM\_ROWS\_OF\_BRICKS) for x in range(NUM\_BRICKS\_PER\_ROW)]

class Paddle:

    def \_\_init\_\_(self):

        self.rect = pygame.Rect((SCREEN\_WIDTH - PADDLE\_WIDTH) / 2, SCREEN\_HEIGHT - PADDLE\_HEIGHT - 10, PADDLE\_WIDTH, PADDLE\_HEIGHT)

    def draw(self):

        pygame.draw.rect(screen, GREEN, self.rect)

    def move(self, direction):

        if direction == "left":

            self.rect.x -= 5

        elif direction == "right":

            self.rect.x += 5

        self.rect.x = max(min(self.rect.x, SCREEN\_WIDTH - self.rect.width), 0)

class Ball:

    def \_\_init\_\_(self, x, y):

        self.rect = pygame.Rect(x, y, BALL\_RADIUS \* 2, BALL\_RADIUS \* 2)

        self.speed\_x = 4 \* random.choice([-1, 1])

        self.speed\_y = -4

    def draw(self):

        pygame.draw.ellipse(screen, WHITE, self.rect)

    def move(self):

        self.rect.x += self.speed\_x

        self.rect.y += self.speed\_y

        # Bounce off walls

        if self.rect.left <= 0 or self.rect.right >= SCREEN\_WIDTH:

            self.speed\_x \*= -1

        if self.rect.top <= 0:

            self.speed\_y \*= -1

        # Bounce off paddle

        if self.rect.colliderect(paddle.rect):

            self.rect.bottom = paddle.rect.top  # Prevent sticking

            self.speed\_y \*= -1

            offset = (self.rect.centerx - paddle.rect.centerx) / (paddle.rect.width / 2)

            self.speed\_x = 5 \* offset

class PowerUp:

    def \_\_init\_\_(self, x, y):

        self.rect = pygame.Rect(x, y, 20, 20)

        self.speed\_y = 2

    def draw(self):

        pygame.draw.rect(screen, YELLOW, self.rect)

    def move(self):

        self.rect.y += self.speed\_y

# Create objects

paddle = Paddle()

balls = [Ball(SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2)]

bricks = create\_bricks()

score = 0

power\_ups = []

# Main game loop

running = True

while running:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            running = False

    # Move paddle

    keys = pygame.key.get\_pressed()

    if keys[pygame.K\_LEFT]:

        paddle.move("left")

    if keys[pygame.K\_RIGHT]:

        paddle.move("right")

    # Move balls

    for ball in balls:

        ball.move()

    # Check collision with bricks

    for brick in bricks[:]:

        for ball in balls:

            if ball.rect.colliderect(brick.rect):

                ball.speed\_y \*= -1

                bricks.remove(brick)

                score += BRICK\_POINTS

                if random.random() < POWER\_UP\_CHANCE:

                    power\_ups.append(PowerUp(brick.rect.x + BRICK\_WIDTH / 2, brick.rect.y))

                break

    # Handle power-ups

    for power\_up in power\_ups[:]:

        power\_up.move()

        if power\_up.rect.colliderect(paddle.rect):

            power\_ups.remove(power\_up)

            new\_ball = Ball(paddle.rect.x + paddle.rect.width / 2, paddle.rect.y - 20)

            new\_ball.speed\_x = random.choice([-4, 4])

            new\_ball.speed\_y = -4

            balls.append(new\_ball)

        elif power\_up.rect.top > SCREEN\_HEIGHT:

            power\_ups.remove(power\_up)

    # Drawing

    screen.fill((0, 0, 0))  # Clear the screen

    paddle.draw()

    for ball in balls:

        ball.draw()

    for brick in bricks:

        brick.draw()

    for power\_up in power\_ups:

        power\_up.draw()

    score\_text = font.render(f"Score: {score}", True, WHITE)

    screen.blit(score\_text, (10, 10))

    # Check game over

    all\_balls\_lost = all(ball.rect.bottom >= SCREEN\_HEIGHT for ball in balls)

    if all\_balls\_lost and balls:

        game\_over\_text = font.render("GAME OVER", True, RED)

        screen.blit(game\_over\_text, (SCREEN\_WIDTH // 2 - game\_over\_text.get\_width() // 2, SCREEN\_HEIGHT // 2 - game\_over\_text.get\_height() // 2))

        pygame.display.flip()

        pygame.time.wait(2000)

        running = False

    # Reset bricks if all are destroyed

    if not bricks:

        bricks = create\_bricks()  # Recreate bricks

    pygame.display.flip()

    pygame.time.delay(10)

pygame.quit()



