import pgzrun  
import pygame  
import random  
  
# Screen dimensions  
WIDTH = 800  
HEIGHT = 400  
  
# Game states  
game\_started = False  
game\_over = False  
score = 0  
lives = 3 # Player starts with 3 lives  
is\_running\_sound\_playing = False # Track running sound state  
  
# Player attributes  
player = Actor("player", (100, HEIGHT - 60))  
player.dy = 0 # Vertical velocity  
  
grav = 0.5 # Gravity  
jump\_power = -15 # Jump strength  
  
# Obstacles & Hearts  
obstacles = []  
hearts = []  
obstacle\_speed = 5  
obstacle\_spawn\_time = 90 # Frames before spawning a new obstacle  
heart\_spawn\_time = 400 # Frames before a heart appears  
spawn\_counter = 0  
heart\_counter = 0  
obstacle\_type = 0 # Toggle between 0 (obstacle1) and 1 (obstacle2)  
  
# Load images  
background = pygame.image.load("images/background.png")  
background = pygame.transform.scale(background, (WIDTH, HEIGHT))  
  
# Resize obstacles  
original\_obstacle1 = pygame.image.load("images/obstacle.png")  
small\_obstacle1 = pygame.transform.scale(original\_obstacle1, (40, 40))  
pygame.image.save(small\_obstacle1, "images/obstacle\_small.png")  
  
original\_obstacle2 = pygame.image.load("images/obstacle2.png")  
small\_obstacle2 = pygame.transform.scale(original\_obstacle2, (40, 40))  
pygame.image.save(small\_obstacle2, "images/obstacle2\_small.png")  
  
# Resize heart power-up  
original\_heart = pygame.image.load("images/heart.png")  
small\_heart = pygame.transform.scale(original\_heart, (30, 30))  
pygame.image.save(small\_heart, "images/heart\_small.png")  
  
  
def reset\_game():  
 *"""Resets the game to the initial state."""* global player, obstacles, hearts, score, game\_over, spawn\_counter, heart\_counter, obstacle\_type, is\_running\_sound\_playing, lives  
  
 player.pos = (100, HEIGHT - 60)  
 player.dy = 0  
 obstacles.clear()  
 hearts.clear()  
 score = 0  
 spawn\_counter = 0  
 heart\_counter = 0  
 game\_over = False  
 obstacle\_type = 0 # Reset obstacle alternation  
 lives = 3 # Reset lives  
  
 sounds.start.play() # Play game start sound  
 sounds.running.play(-1) # Start running sound again  
 is\_running\_sound\_playing = True  
  
  
def update():  
 *"""Updates game state every frame."""* global spawn\_counter, heart\_counter, score, game\_over, obstacle\_type, is\_running\_sound\_playing, lives  
  
 if not game\_started or game\_over:  
 return # Don't update game if it's not started or game over  
  
 # Apply gravity  
 player.dy += grav  
 player.y += player.dy  
  
 # Keep player on screen  
 if player.y > HEIGHT - 60:  
 player.y = HEIGHT - 60  
 player.dy = 0  
  
 # Play running sound if not already playing  
 if not is\_running\_sound\_playing:  
 sounds.running.play(-1)  
 is\_running\_sound\_playing = True  
  
 # Move obstacles & hearts  
 for obstacle in obstacles:  
 obstacle.x -= obstacle\_speed  
  
 for heart in hearts:  
 heart.x -= obstacle\_speed  
  
 # Remove off-screen objects  
 obstacles[:] = [ob for ob in obstacles if ob.x > -50]  
 hearts[:] = [heart for heart in hearts if heart.x > -50]  
  
 # Spawn obstacles alternately  
 spawn\_counter += 1  
 if spawn\_counter >= obstacle\_spawn\_time:  
 spawn\_counter = 0  
 if obstacle\_type == 0:  
 obstacles.append(Actor("obstacle\_small", (WIDTH, HEIGHT - 60))) # First obstacle  
 obstacle\_type = 1 # Next time, spawn second obstacle  
 else:  
 obstacles.append(Actor("obstacle2\_small", (WIDTH, HEIGHT - 60))) # Second obstacle  
 obstacle\_type = 0 # Next time, spawn first obstacle  
  
 # Spawn hearts randomly  
 heart\_counter += 1  
 if heart\_counter >= heart\_spawn\_time:  
 heart\_counter = 0  
 if random.random() < 0.5: # 50% chance to spawn a heart  
 hearts.append(Actor("heart\_small", (WIDTH, HEIGHT - 100))) # Heart appears higher  
  
 # Collision detection - Obstacles  
 for obstacle in obstacles:  
 if player.colliderect(obstacle):  
 obstacles.remove(obstacle) # Remove obstacle on collision  
 lives -= 1 # Lose a life  
 sounds.crash.play() # Play crash sound  
  
 if lives <= 0: # If lives run out, game over  
 game\_over = True  
 sounds.running.stop() # Stop running sound  
 sounds.gameover.play() # Play game over sound  
  
 # Collision detection - Hearts (Power-up)  
 for heart in hearts:  
 if player.colliderect(heart):  
 hearts.remove(heart) # Remove heart when collected  
 if lives < 3: # Only increase life if it's less than 3  
 lives += 1  
 sounds.powerup.play() # Play heart collection sound  
  
 # Score keeping  
 score += 1  
  
  
def draw():  
 *"""Renders the game objects."""* screen.surface.blit(background, (0, 0)) # Draw background  
  
 if not game\_started:  
 screen.draw.text("Press ENTER to Start", center=(WIDTH // 2, HEIGHT // 2), color="white", fontsize=40)  
 return  
  
 screen.draw.text(f"Score: {score}", (10, 10), color="white", fontsize=30)  
  
 # Draw Lives (Hearts on screen)  
 for i in range(lives):  
 screen.blit("heart\_small", (WIDTH - 100 + (i \* 30), 10))  
  
 player.draw()  
 for obstacle in obstacles:  
 obstacle.draw()  
 for heart in hearts:  
 heart.draw()  
  
 if game\_over:  
 screen.draw.text(  
 "Game Over! Press ENTER to Restart",  
 center=(WIDTH // 2, HEIGHT // 2),  
 color="red",  
 fontsize=40,  
 )  
  
  
def on\_key\_down(key):  
 *"""Handles player input."""* global game\_started, game\_over, is\_running\_sound\_playing  
  
 if key == keys.RETURN and not game\_started: # Start game when ENTER is pressed  
 game\_started = True  
 reset\_game() # Restart the game variables  
  
 if key == keys.SPACE and player.y == HEIGHT - 60 and game\_started:  
 player.dy = jump\_power  
 sounds.jump.play() # Play jump sound  
 sounds.running.stop() # Stop running sound when jumping  
 is\_running\_sound\_playing = False # Running stops when jumping  
  
 if key == keys.RETURN and game\_over: # Restart game when ENTER is pressed  
 game\_over = False # Reset game state  
 game\_started = True  
 reset\_game() # Restart after game over  
  
  
pgzrun.go()