Creating SNAKE GAME using PYGAME

Pygame is a Python library designed for creating 2D games and multimedia applications, built on top of the Simple DirectMedia Layer (SDL) library, which handles low-level tasks like graphics, audio, and input.

Game Setup and Initialization

The snake game begins with importing necessary libraries and initializing Pygame:

```
import pygame
import sys
import random

pygame.init()
```

The game uses several constants to define its behavior and appearance:

```
# Constants
WIDTH, HEIGHT = 700, 700
BLOCK_SIZE = 25
FPS = 7
WHITE = (255, 255, 255)
RED = (200, 0, 0)
BLACK = (0, 0, 0)
```

Loading Game Assets

The game loads and scales images for the background, snake head, and food:

```
# Load images
background_img = pygame.transform.scale(pygame.image.load("background.jpg"), (WIDTH, HEIGHT))
snake_head_img = pygame.transform.scale(pygame.image.load("snake_head.png"), (BLOCK_SIZE))
food_img = pygame.transform.scale(pygame.image.load("apple.png"), (BLOCK_SIZE))
```

The display window is then set up with a title:

```
# Set up display
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Snake Game")

# Font
font = pygame.font.SysFont("consolas", 35)
```

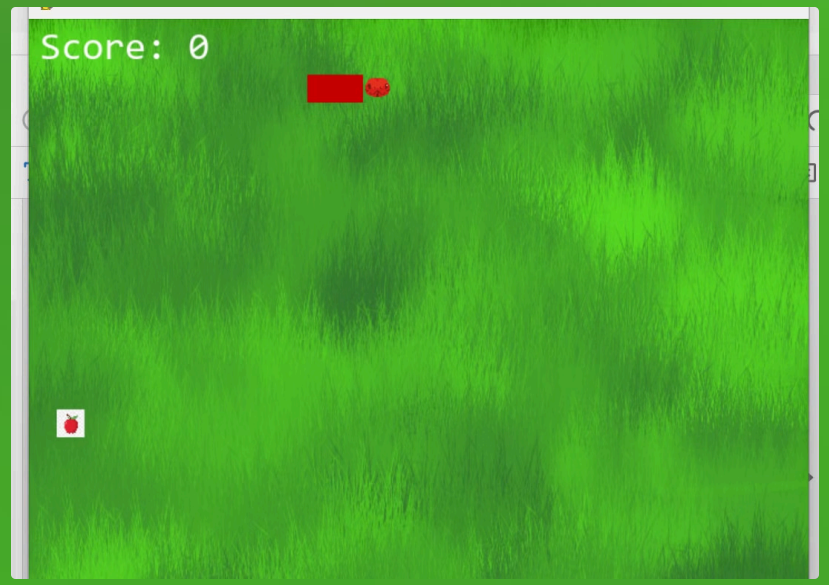
Game Over Function

The game over screen displays the final score and provides options to restart or quit:

```
def show_game_over(score):
    while True:
        screen.fill(BLACK)
        draw_text("Game Over!", 60, RED, -50)
        draw_text(f"Score: {score}", 40, WHITE, 20)
        draw_text("Press R to Restart or Q to Quit", 30,
WHITE, 80)
        pygame.display.flip()
```

The function handles user input to either restart the game or exit:

```
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit(); sys.exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_q:
                    pygame.quit(); sys.exit()
                if event.key == pygame.K_r:
                    main()
```



Main Game Function

The main function initializes the game state and contains the main game loop:

```
def main():
    clock = pygame.time.Clock()
    snake = [[100, 50], [75, 50], [50, 50]]
    direction = "RIGHT"
    score = 0

    food = [random.randrange(0, WIDTH, BLOCK_SIZE),
random.randrange(0, HEIGHT, BLOCK_SIZE)]

    running = True
    while running:
        clock.tick(FPS)
```

The clock.tick(FPS) function regulates the game speed by controlling the frame rate.

Handling User Input

The game processes keyboard input to control the snake's direction:

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
       pygame.quit(); sys.exit()
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_UP and direction != "DOWN":
            direction = "UP"
        elif event.key == pygame.K_DOWN and direction != "UP":
            direction = "DOWN"
        elif event.key == pygame.K_LEFT and direction != "RIGHT":
            direction = "LEFT"
        elif event.key == pygame.K_RIGHT and direction != "LEFT":
            direction = "RIGHT"
```

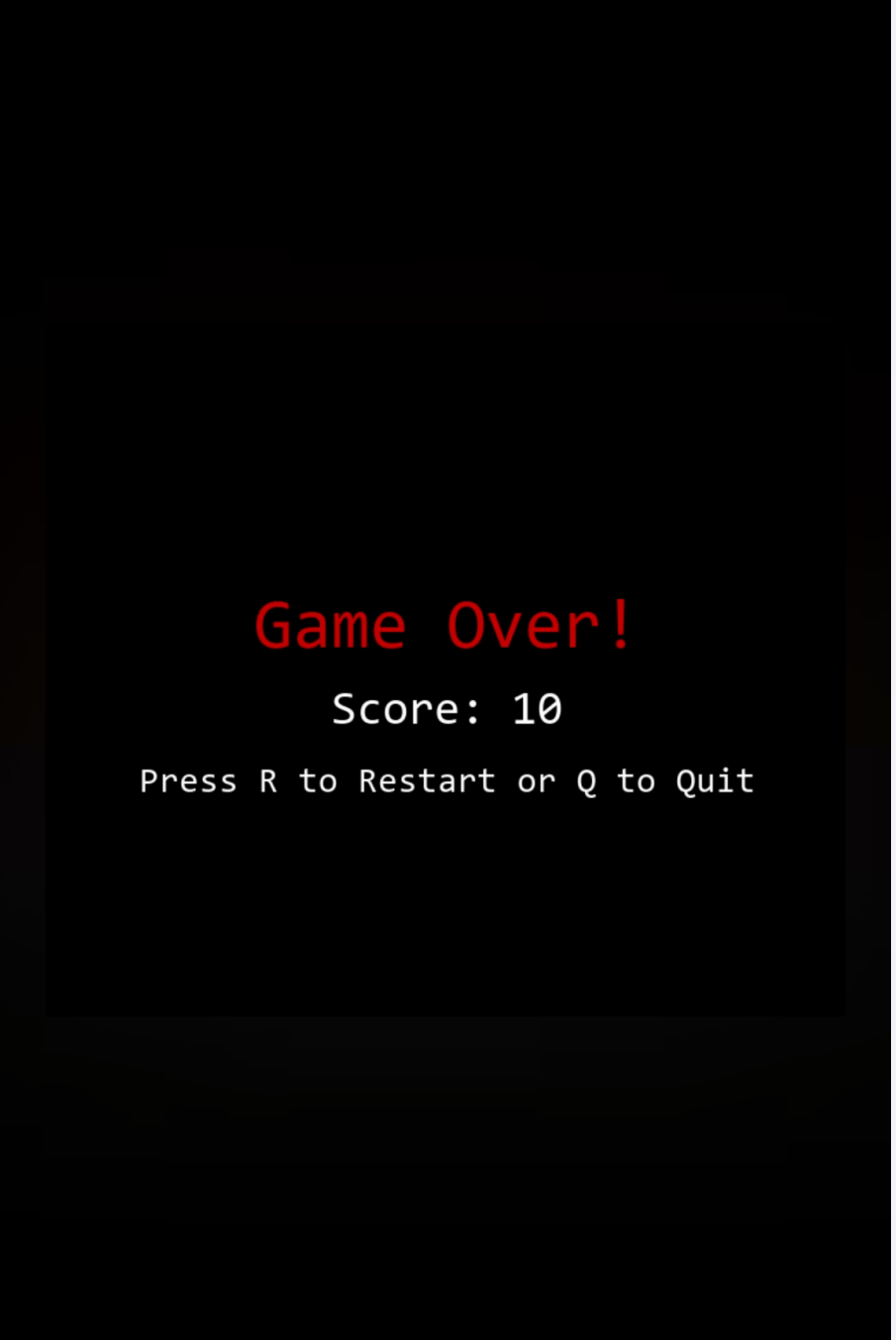
Snake Movement and Collision Detection

The snake moves by adding a new head in the direction of movement:

```
head_x, head_y = snake[0]
if direction == "UP": head_y -= BLOCK_SIZE
if direction == "DOWN": head_y += BLOCK_SIZE
if direction == "LEFT": head_x -= BLOCK_SIZE
if direction == "RIGHT": head_x += BLOCK_SIZE
new_head = [head_x, head_y]
snake.insert(0, new_head)
```

When the snake eats food, it grows and the score increases:

```
if new_head == food:
    score += 1
    food = [random.randrange(0, WIDTH,
BLOCK_SIZE), random.randrange(0, HEIGHT,
BLOCK_SIZE)]
else:
    snake.pop()
```



Game Output

When the game runs, it displays the snake moving around the screen, eating apples, and growing longer. The score increases with each apple eaten.

When the snake collides with itself or the boundaries, the game ends and displays:

> Game Over! Score: 10 Press R to Restart or Q to Quit