Snake Game Project

Design and Analysis of Alogrithms
Final Project

Submitted By :

Arkum Muhammad
{45303}
Muhammad Basheer Shah
{47141}

Department of Computer Science

Riphah International University, ISLAMABAD

[11 - 11 – 2024]

Report & Analysis

1. Introduction :

The Snake game is a classic arcade game where the player controls a snake, aiming to eat apples to grow in length while avoiding collisions with walls and the snake's own body. This project implements the Snake game using Python, specifically utilizing the Pygame library for graphics and game functionality.

2. Game Overview :

The game consists of a snake and an apple on a grid. The snake moves in four directions: left, right, up, and down. The player controls the snake, attempting to eat apples that appear at random locations on the screen. Each time the snake eats an apple, its length increases, and a new apple appears. If the snake collides with the boundaries of the screen or itself, the game ends.
Key features of the game include :

- ***Snake Movement :*** The snake is controlled using arrow keys.
- ***Apple consumption:*** Eating an apple increases the snake's length.
- ***Collision detection:*** The game ends if the snake collides with itself or the boundaries.
- ***Background music and sound effects:*** Background music and sound effects play during the game.
- ***Game-over screen:*** Displays the score and gives the player the option to restart the game or exit.

3. Code Structure and Explanation :

The project is structured as follows:

- Apple Class: Handles the apple object, including its initial position and drawing.
- Snake Class: Handles the snake's behavior, movement, collision detection, and growth when an apple is eaten.
- Game Class: Manages the game flow, including collision detection, background rendering, score display, and game-over logic.

Key Functions :

- ***move_left(), move_right(), move_up(), move_down():*** Control the snake's direction.
- ***walk():*** Updates the snake's position, moving the body and the head.
- ***is_collision():*** Detects collisions between objects (snake and apple, snake and walls, snake and its own body).
- ***play():*** The main game loop that updates the game state each frame.
- ***display_score():*** Displays the current score in the top right corner of the screen.
- ***show_game_over():*** Displays a game-over message when the game ends.

4. Time Complexity Analysis :

The main operations of the game occur in the play() function, which runs the game loop. The time complexity is determined by how often the game state is updated.

1. Movement & Collision Detection :

- The snake's movement involves updating the position of each part of the snake, which is a linear operation. For a snake of length n n, updating the positions of the snake body parts takes $O(n)$.
- Collision detection between the snake's head and its body is also $O(n)$, where n n is the snake's length.

2. Apple Consumption :
   - Checking if the snake eats the apple involves checking the position of the snake's head and the apple, which is an $O(1)$ operation.

3. Overall Time Complexity :
   - The overall time complexity of each game loop iteration is $O(n)$, where n n is the length of the snake.

5. Space Complexity Analysis :

The space complexity is determined by the number of objects and their sizes in memory.

1. ***Snake:*** The snake's body is represented as two lists, x and y, each holding the coordinates of each segment of the snake. The space complexity for the snake is $O(n)$, where n is the length of the snake.
2. ***Apple:*** The apple is represented by its position in space and the image. The space complexity is $O(1)$.
3. ***Overall Space Complexity:*** The overall space complexity is $O(n)$, where n is the length of the snake, due to the space required to store the snake's coordinates.

6. Enhancements and Features :

The game could be enhanced in several ways :

- ***Difficulty Levels:*** Increase the speed of the snake as the player progresses or add more obstacles to the game.
- ***High Score Tracking:*** Implement a system to track and display the player's highest score.
- ***Graphical Improvements:*** Improve the appearance of the snake and the apple, perhaps using custom images or animations.
- ***Multiplayer Mode:*** Allow two players to control two snakes and compete for apples.

7. Conclusion :

This Snake game project effectively demonstrates the use of Python and Pygame to create a fully functional, interactive game. The game incorporates fundamental game development concepts such as event handling, collision detection, and animation. The time and space complexity analysis reveals that the game's performance is manageable even with longer snakes, thanks to the linear growth of the snake's body. Future enhancements could improve the game's replay value and user experience.