Final Project Report — Programming Fundamentals

University Name: NUCES FAST KARACHI.
Department: Department of Software Engineering.
Course: Programming Fundamentals
Project Title: Snake Game (Console + Raylib)
Submitted By: Muhammad Sheharyar Zafar (25K-3041) , Muhammad Zeeshan (25K-3068).
Submitted To: Ms. Izzah Salam
Semester: Fall 2025
Date: 24/Nov/2025

Abstract

The Snake Game project includes two versions: one made using the console in C and the other built using the Raylib library. Both versions follow the same basic idea, moving a snake, eating food, and avoiding collisions but differ in controls, visuals, and user experience. The console version uses simple characters to draw the snake and boundaries, while the Raylib version uses graphics to draw a smoother and more modern-looking game. This project helped us understand loops, arrays, conditions, game logic, and how to manage continuous movement in a program.

1. Introduction

This project focuses on creating the classic Snake Game using two different approaches. The first version is made in a text-based console using basic C functions, while the second uses Raylib to create a graphical version. Both versions helped us learn how to control movement, detect collisions, update the screen, and handle player input. The project also strengthened our understanding of arrays, coordinate systems, and simple game development logic.

2. Objectives

- To build a console-based version of the Snake Game.
- To understand how movement, collisions, and tail updates work.
- To practice using arrays for storing snake body positions.
- To learn how to take input continuously while updating the game.

3. System Design

System Overview:
The game starts by placing the snake in the center and generating food at a random location. The player controls the snake using W, A, S, D keys. Each time the snake eats food, the score increases and the tail grows. The game ends when the snake hits a wall or collides with itself.

Algorithm:
1. Start the game

2. Place snake and food
3. Take input and update direction
4. Move snake and shift tail
5. Check collisions
6. Update score if food is eaten
7. Redraw game
8. Repeat until game ends

Input & Output:
Input: Movement keys (W, A, S, D)
Output: Snake movement, food, score, and game-over message.

4. Implementation

Language: C
Compiler: MinGW64/UCRT64.

Key Features:
- Snake movement controlled by keyboard
- Tail grows when food is eaten
- Collision detection with walls and tail
- Score increases over time
- Raylib version includes graphics, colored blocks, and smooth rendering

Console Version Snippet:

```
switch (direction) {
   case 1: headY--; break;
   case 2: headY++; break;
   case 3: headX--; break;
   case 4: headX++; break;
}
```

Raylib Version Snippet:

```
DrawRectangle(snakeX[i] * CELL_SIZE, snakeY[i] * CELL_SIZE, CELL_SIZE, CELL_SIZE, GREEN);
```

5. Testing & Results

Test 1:
Input: Snake moves normally and eats food
Expected Output: Score increases, tail grows
Actual Output: Works correctly

Test 2:
Input: Snake hits wall

Expected Output: Game ends (console) / game resets (Raylib)
Actual Output: Works correctly

Test 3:
Input: Snake collides with its own tail
Expected Output: Game ends or resets
Actual Output: Correct behavior

Overall, both versions worked as expected and handled input, movement, and collisions properly.

6. Conclusion, Limitations & References

Conclusion:
This project helped us understand how classic games are built. The console version taught us basic logic and how to draw using characters. The Raylib version showed us how graphics libraries work and how to refresh frames smoothly.

Limitations:
- Console version has no graphics
- Raylib version restarts instantly instead of showing a game-over screen
- No menus or difficulty levels
-Flickering/ Low frame-time on Console version.

Future Enhancements:
- Add sound effects and better animation
- Add a game-over screen and menus
- Add levels or increasing speed

References:
- Raylib documentation.
- YouTube tutorials.
-C Documentation.