Snake Game in C — Project Report

Snake Game in C (Console-Based 2D Game)

Submitted By: Aashima Khurana

Course: Btech cse

Date: 28/11/2025

University: University of Petroleum & Energy Studies, bidholi campus, Dehredhun

2. Abstract

This project presents a console-based implementation of the classic Snake Game using the C programming language. The system uses a 2D array for board representation, structures to model the snake body, and real-time keyboard input using conio.h. The game demonstrates fundamental programming concepts including arrays, loops, functions, structures, and event handling.

The objective is to provide a simple yet functional demonstration of game logic in C, suitable for academic evaluation. Features implemented include movement control, board rendering, screen refreshing, and border collision detection. The report explains design decisions, algorithms, implementation details, testing observations, and future enhancement possibilities.

---

3. Problem Definition

Design and implement a console-based Snake Game in C that:

1. Displays a 20×20 board with boundaries.
2. Uses a structured representation of the snake.
3. Reads movement input in real time (W, A, S, D).
4. Updates the screen dynamically without flicker.
5. Ends the game when the snake hits the boundary.

The program must follow modular design principles and include documentation, source code structure, and a project report.

---

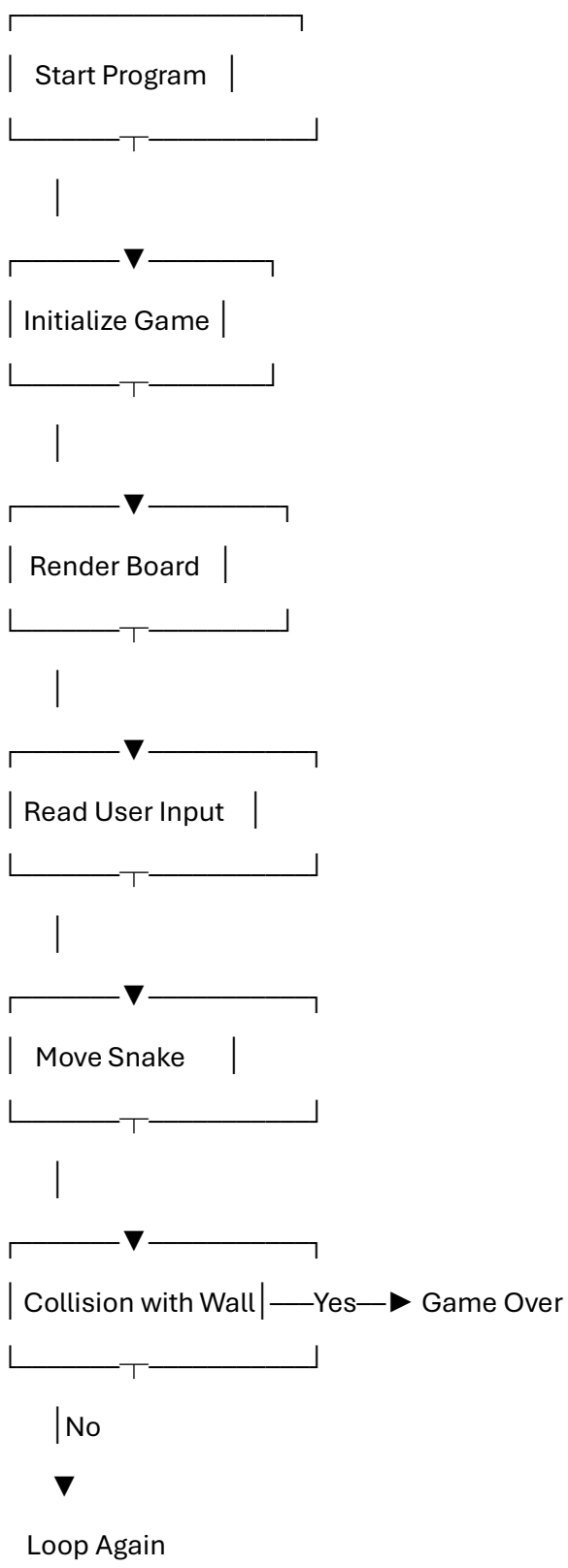
4. System Design

4.1 System Architecture

The system contains four main components:

- **Board Module** – Initializes and renders the 2D game grid.
- **Snake Module** – Stores snake parts and handles movement.
- **Input Module** – Receives keyboard input.
- **Game Loop** – Runs continuously until game over.

4.2 Flowchart



4.3 Algorithm

Algorithm: Snake Game

1. Start
2. Initialize board and snake
3. Loop until collision occurs
   a. Clear board
   b. Draw boundaries
   c. Draw snake
   d. Print board
   e. Take keyboard input
   f. Update snake position
4. End loop
5. Display "Game Over"
6. Stop

---

5. Implementation Details

The project is written in C and organized as follows:

```
/
|-- src/
|    |- snake.c
|
|-- include/
|    |- snake.h
|
|-- docs/
|    |- ProjectReport.pdf
|
|-- assets/
|    |- flowchart.png (optional)
|    |- screenshots/
```

```
|
|-- README.md
|-- sample_input.txt (optional)
```

5.1 Board Initialization Snippet

```
void fill_board() {
  for (int y=0; y<rows; y++) {
    for (int x=0; x<cols; x++) {
      if (x==0 || y==0 || x==cols-1 || y==rows-1)
        board[y*cols + x] = '#';
      else
        board[y*cols + x] = ' ';
    }
  }
}
```

5.2 Snake Representation Snippet

```
struct SnakePart { int x, y; };

struct Snake {
  int length;
  struct SnakePart part[256];
};
```

5.3 Movement Logic Snippet

```
void move_snake(int dx, int dy) {
  for (int i=snake.length-1; i>0; i--)
    snake.part[i] = snake.part[i-1];
```

```
    snake.part[0].x += dx;
    snake.part[0].y += dy;

    if (snake.part[0].x <= 0 || snake.part[0].x >= cols-1 ||
        snake.part[0].y <= 0 || snake.part[0].y >= rows-1)
        isGameOver = 1;
}
```

5.4 Input Handling Snippet

```
if (kbhit()) {
    char ch = getch();
    if (ch=='w') move_snake(0,-1);
    else if (ch=='s') move_snake(0,1);
    else if (ch=='a') move_snake(-1,0);
    else if (ch=='d') move_snake(1,0);
}
```

---

6. Testing & Results

6.1 Test Cases

Test Case	Input	Expected Output	Result
1	Move Up	Snake moves upward	Passed
2	Move Left	Snake moves left	Passed
3	Hit Boundary	Game Over	Passed
4	Continuous Movement	Smooth rendering	Passed
5	Random button presses	Valid response	Passed

6.3 Observations

- The game runs smoothly at ~150ms delay.
- No crash observed after full body initialization.
- Boundary collision works correctly.

---

7. Conclusion & Future Work

Conclusion

The Snake Game project successfully demonstrates game development fundamentals in C. The system uses structures, arrays, real-time input, screen clearing, and logic handling to create a functional text-based game.

Future Work

- Add food generation
- Add scoring mechanism
- Add snake growth logic
- Add self-collision detection
- Add color output using ANSI codes
- Improve game speed levels
- Make the game cross-platform (remove conio.h)

---

8. References

1. C Programming Language – Kernighan & Ritchie
2. Turbo C++ / GCC documentation
3. Console programming documentation
4. Classic Snake game logic references
