Semester Project Report

Course	Data Structures

Submitted By

Sami Ur Rehman	SP24-BSE-086
Maham Iftikhar	SP24-BSE-056
Komal Fawad	SP24-BSE-053

Submitted To

Instructor	Dr. Tahir Maqsood

Date: June 21, 2025

Introduction & Features of the Project

➢ Introduction

- **Project Title: *Desert Slither***
- **Technology Used:** Java (JavaFX), Data Structures, Object-Oriented Programming.
- **Theme:** A modern version of the classic snake game with enhanced visuals and gameplay elements.
- **Project Type:** Desktop Game Application
- *Desert Slither* is a Data Structures and Algorithms (DSA) based implementation of the classic Snake game using JavaFX. While the game offers an engaging visual experience and smooth gameplay, its core strength lies in the underlying **DSA concepts**. This project serves as a practical demonstration of applying **linked lists, game loops, event handling, and algorithmic state transitions** in a real-time environment.
- At the heart of the game is a **custom-built singly linked list** that represents the snake's body. Each node in the list corresponds to a segment of the snake (head, body, tail), dynamically growing as the snake consumes food. This structure allows efficient manipulation of the snake's size and movement, making the game an excellent example of how core data structures are applied beyond theory.
- The project also integrates additional algorithms to manage movement direction, collision detection, random food generation, and scene transitions, showcasing how logical thinking and structured algorithm design are essential in game development.

➢ Key Features

1. Splash Screen with Logo Animation

A professionally animated splash screen with the game's logo that fades in and out before transitioning to the login screen.

2. User Login & Sign-Up System

Users are required to either log in or sign up before accessing the game. This creates a personalized experience and is a potential foundation for storing individual scores.

3. Multiple Difficulty Modes

- **Easy:** Slower snake speed, game ends only if the snake bites itself.
- **Medium:** Faster speed, self-collision ends the game.
- **Hard:** Fast speed, game ends on both self-collision and border collision, making it highly challenging.

4. Snake Representation using Singly Linked List

The snake is modeled as a SnakeLinkedList, where each node holds coordinates of the segment. Efficient node insertion (when eating) and deletion (when moving) replicate snake growth/shrink behavior.

5. Game Loop Algorithm

The game runs on a timed loop using `Timeline` to simulate real-time motion. Each cycle updates the snake's position, checks for collisions, and renders graphics accordingly.

6. Collision Detection Algorithms

Self-collision detection is performed by traversing the linked list and comparing head coordinates with body segments.

In **Hard Mode**, additional boundary collision logic is used.

7. Randomized Food Generation and Bonus Items

- Randomized coordinates are generated for placing food and bonus items. Ensures that food does not spawn inside the snake's body (collision avoidance).
- Regular food increases score by 1.
- Special **Diamond Bonus Food** increases score by 5, adding an exciting gameplay twist.

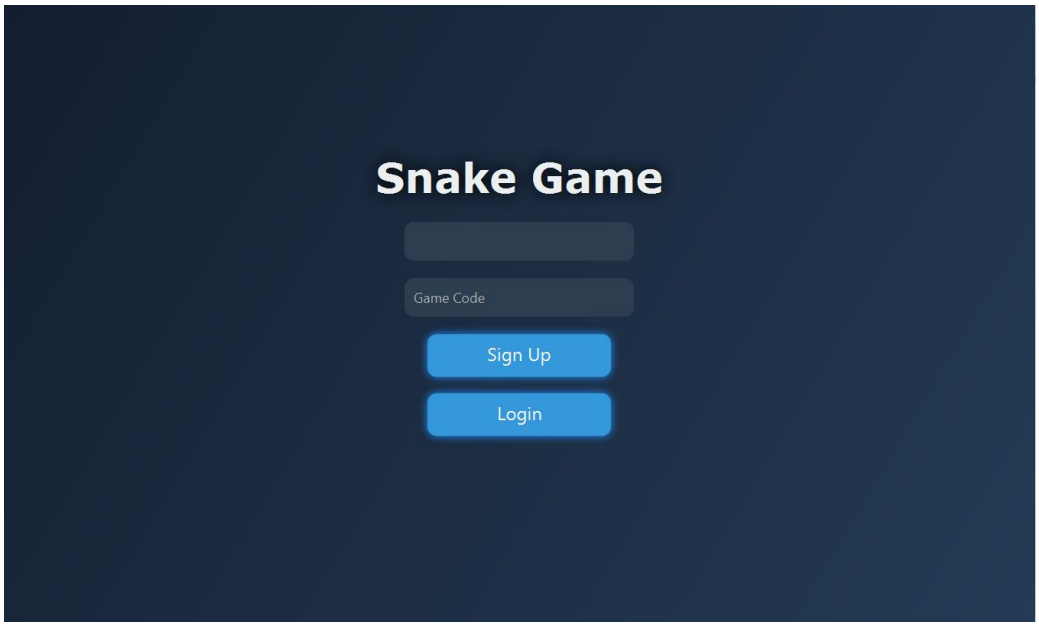
8. Score Tracking & File Handling

The score is calculated algorithmically based on food type. A file-based score saving system is in place, with plans to add high-score leaderboard using file I/O.

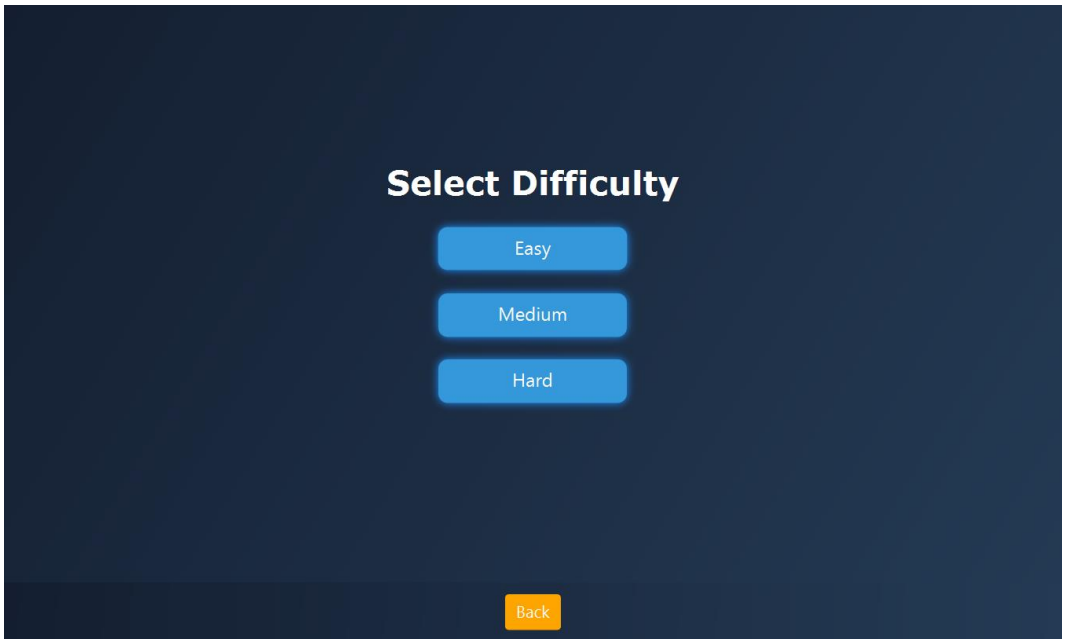
Output/Screenshots

1. Logo Animation

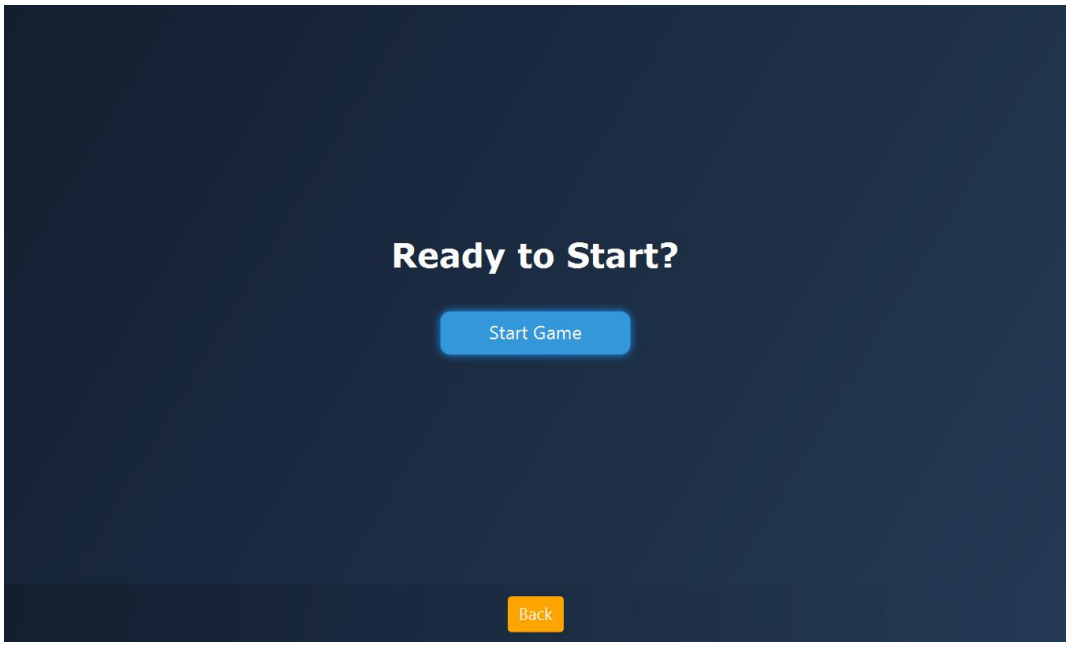
2. Login Screen



3. Difficulty Selection



4. Ready To Start Screen



5. Game Started

6. Snake Grows By Eating Food

7. Bonus Food

8. Game Overs in Medium Mode (Snake Biting Itself)

9. Game Over in Hard Mode (Snake Colliding with Border)

10. Pressed Space to Restart the Game

Conclusion, Findings and Challenges

➢ Challenges

1. Snake Structure Logic

One of the major hurdles was designing the snake as a realistic body, consisting of a head, body, and tail using linked segments. Initially, simple ovals were used, but later a custom `LinkedList` structure (`SnakeLinkedList.java`) was implemented to make the snake grow and move properly, simulating natural movement.

➢ Future Enhancements

1. Scorecard System:

- Save scores using file handling (already implemented).
- Add UI element in-game to view the top scores directly.
- Highlight highest score achieved across sessions.

2. Missions/Challenges Mode:

- **Introduce timed tasks like:**
  - "Eat 5 red apples in 30 seconds"
  - "Survive without food for 1 minute"

3. Themes & Skins:

- **Add background themes:**
  - Day/Night mode
  - Jungle or Desert Theme
  - Retro Pixel Grid Style

4. Variety of Food Items:

- Introduce different types of food (with different effects or score boosts).
- Add animations or glowing effects for rare foods like bonus diamonds.

➢ Conclusion

Desert Slither effectively applies key Data Structures and Algorithms (DSA) concepts in a real-time game environment. The snake is implemented using a custom singly linked list, allowing dynamic growth and movement through efficient node operations. Collision detection, food generation, and difficulty-based logic were designed using structured algorithms and control flow techniques. This project strengthened our understanding of how foundational DSA principles like linked lists and algorithm design can power interactive, user-facing applications.

Group Members and Contributions

Members	Contributions
Sami Ur Rehman	Designed splash screen, difficulty levels, and scene transitions. Helped with UI/UX consistency.
Maham Iftikhar	Added sound effects, login/signup system. Handled SnakeLinkedList, game loop and worked on future logic.
Komal Fawad	Implemented game logic, snake movement, food & scoring system, and worked on bonus food.