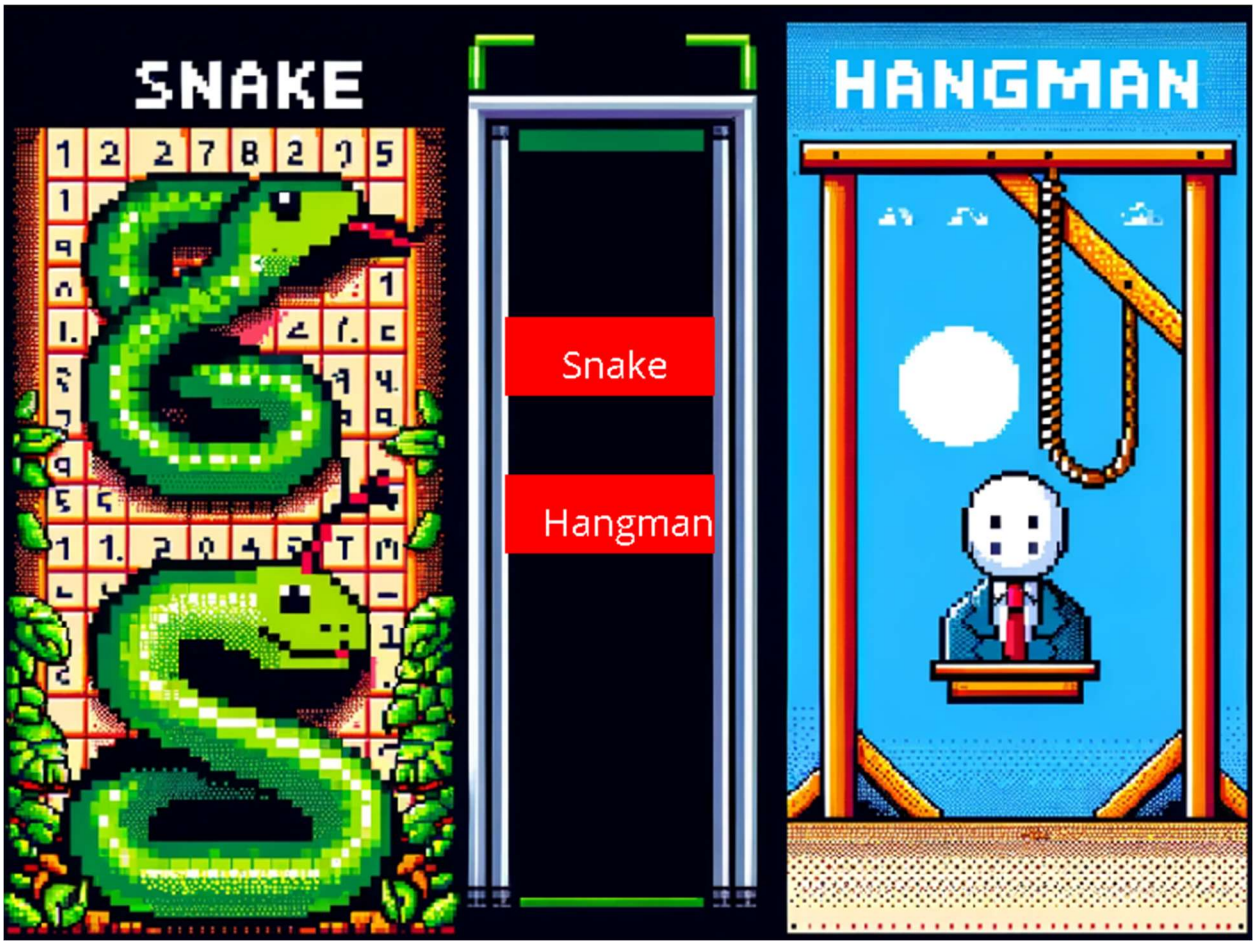
Final Project Report: Retro Gaming Reinvented through Data Structures



Group Members

Bilal Ahmed

Qazi Muhammad Mustata

Ayesha Aimen

Simra Shamim

Project Specification:

This project explores the implementation of two distinct data structures, Trie, and Doubly Linked List, within two popular games, Hangman and Snake. The development was done using SDL libraries in C++, with a focus on creating interactive and engaging gameplay through the efficient use of data structures. It uses an integrated main starting screen for both games features background music to enhance user engagement.

Data Structures Used

- Trie: Implemented in the Hangman game to manage a dictionary of words, facilitating quick retrieval and insertion.
- Doubly Linked List: Applied in the Snake game to dynamically handle the movement and growth of the snake during gameplay.

Application Features

Integrated Starting Screen:

- Features an initial menu that allows players to choose between Hangman and Snake.
- Background music sets a playful and engaging tone, enhancing the overall gaming experience.
- The screen is developed using SDL, showcasing smooth transitions and high-quality graphics.

Hangman Game:

- Word Selection: Utilizes a Trie to store a large dictionary for rapid word access, crucial for initializing and resetting game rounds.
- Gameplay Mechanics: Interactive gameplay where each guess is checked against the Trie, ensuring efficient progression without delays.
- User Interface: Shows the current word progress, hangman's status, and remaining guesses, updating in real time to reflect player interactions.
- Interactive Elements: Keyboard inputs are managed via SDL, with each letter guess instantly updating the game display and hangman graphic.

Snake Game:

- Game Mechanics: A Doubly Linked List represents the snake's body, facilitating efficient constant-time operations as the snake navigates the board.
- Movement and Growth: Instant response to keyboard commands, enabling dynamic movement and growth as the snake consumes apples.
- Collision Detection: Continuous monitoring for collisions with boundaries or self, managed through direct list manipulations.
- Rendering and Speed Adjustment: Uses SDL for fluid visual updates and adjusts the snake's speed based on its length, progressively increasing the game's difficulty.

Complexity Analysis

Trie in Hangman:

Time Complexity: Efficient O(L) operations for search, insert, and delete, where L is the length of the word.

Space Complexity: Though Tries can be space-intensive, this implementation uses shared prefixes to minimize space usage.

Doubly Linked List in Snake:

Time Complexity: O(1) for each insertion and deletion, supporting the high-speed requirements of real-time gaming.

Space Complexity: Grows linearly with the snake's length but is managed to maintain optimal performance.

Challenges Faced

- **Integrating SDL for Dynamic Graphics:** Ensuring smooth, real-time animations for the Snake game required precise graphical updates and synchronization.
- **Optimizing Trie for Efficient Word Management:** Enhancing the Trie was crucial for rapid word management and instant feedback in the Hangman game.
- **Designing a Robust User Interface with SDL:** Developing an intuitive user interface capable of handling real-time updates and maintaining responsiveness was a complex task.

Work Division

- Snake Game Development (20% - Bilal and Qazi):
  - Implementation of game mechanics and dynamics.
  - Integration of SDL for graphics and event handling.
  - Optimization of game performance and responsiveness.

- Doubly Linked List Implementation (15% - Bilal and Qazi):
  - Design and implementation of the Doubly Linked List data structure.
  - Integration of the list for managing snake movement and growth in the Snake game.
  - Testing and debugging to ensure proper functionality.

- Trie Implementation (15% - Ayesha and Aimen):
  - Development of the Trie data structure for word management in Hangman.
  - Optimization for efficient word insertion and retrieval.
  - Testing and refinement to ensure seamless integration with the game mechanics.

- Hangman Game Development (20% - Simra and Ayesha Aimen):
  - Implementation of Hangman game logic and functionality.
  - Integration of Trie for word selection and management.
  - User interface design and development, including graphics and interaction feedback.

- Main Screen Integration (15% - Simra and Qazi):
  - Design and implementation of the main screen interface.
  - Integration of both games into the main screen for seamless navigation.
  - Testing and refinement of the main screen functionality for optimal user experience.

- Graphics (15% - Bilal and Qazi):
  - Making the images for each hangman frame
  - Including winning and lose screen.