**Snake Game Using Raylib: Project Report**

Samson Dawit Bekele

Department of Computer Science, Al-Farabi Kazakh National University

Languages and Programming Technologies

Mr. Makhmut Ye.

September 27, 2023

**Snake Game Using Raylib: Project Report**

In the developed Snake game, the Raylib library has been employed, which is renowned for its simplicity and efficiency in game development. The game showcases an advanced version of the classic Snake game with additional features and gameplay elements. In the course of developing this project, valuable insights and code components were obtained from the GitHub repository by (educ8s, 2023) and YouTube video titled “Creating Simple Snake Game in C++” (ProgrammingKnowledge, 2021).

**Technologies**

***C++:*** As the primary programming language, C++ will serve as the foundation for developing the game's logic and mechanics. I will be using C++ 17 with the MinGW g++ compiler. (DevDocs, 2023)

***Raylib:*** A C library used for graphics and input handling, making it suitable for game development. It is an open-source and cross-platform library, making it accessible to developers on various operating systems, including Windows, macOS, Linux, Android, and more. Raylib's primary goal is to provide a straightforward and efficient framework for creating 2D and 3D games, simulations, and interactive applications. (Raylib, n.d)

***Deque:*** A C++ data structure used to manage the snake's body as a queue. This data structure enables efficient control and manipulation of the snake's segments, ensuring that the game mechanics run smoothly. (Double-ended queue, 2023)

***Raymath:*** A library for mathematical operations in Raylib, used for vector calculations. It provides functions for vector operations like addition and subtraction, which are essential for calculating the snake's new position as it moves in different directions. It supports distance calculations between points, which are crucial for detecting collisions between the snake's head and the food items or boundaries of the game grid. (Raylib, n.d)

**Functionality**

The Snake Game project is designed with an object-oriented approach, where distinct objects encapsulate the game's entities and functionalities. OOP characteristics such as inheritance, overloading and encapsulation were used.

***Snake Class.*** The Snake class is at the heart of the game. This class is responsible for the fundamental operations of the snake such as drawing itself on the display, updating its position, and resetting when necessary. The snake’s body is managed as a deque of ‘Vector2’ objects. This allows the addition and removal of segments to simulate movement and growth. The ‘Update’ method in the snake class is important as it advances the snake in its current direction, and handling the addition of new segments when the snake consumes food. Moreover, the ‘Reset’ method resets the snake’s position and direction if the player decides to restart.

***RandomSnake Class.*** This class is inherited from the snake class. This snake moves randomly (using a random number generator which dictates its movements). It changes directions at random, creating unexpected obstacles and dynamic interactions within the game space. The RandomSnake is drawn onto the board when the score of the player is 5 or above.

***Food Class.*** The Food class will be tasked with generating and rendering the `regularfood` on the game board. It will ensure that `regularfood` items appear randomly on the grid without overlapping the snake's body. The `regularfood`, represented by an apple, is loaded from an image file. When the snake eats this food, its `score` is increased by one and its `segment` is also increased by one.

***SpecialFood Class.*** This class is inherited from the `food class`. It represents a `specialfood` item (represented by a heart) that offers 2 more lives if the snake eats it. However, this heart does not appear all the time. There is a 30% chance of it appearing if the snake’s life is less than 4 and the score is 10 or above. This is done to make the game more interesting.

***Game Class.*** The Game class serves as the central hub for managing the overall game state. This class ties together all elements of the game, managing the state, updating entities, and handling collisions. The snake's interactions with food items are meticulously checked; consuming regular food increases the score and lengthens the snake, while special food provides additional lives. It also monitors collisions with the game boundaries, the snake's own body and the RandomSnake. Additionally, the class incorporates a scoring mechanism and audio feedback.

An essential aspect of the Game class is its handling of game states, such as running, pausing, and restarting the game. It also controls the difficulty level, dynamically adjusting the game's speed based on the player's score. It increases the game speed by 5FPS (base is 30) each time the player gets 5 scores.

The game's primary loop, located in the Game Class, will continually update the game state, respond to player input (arrow key presses), and render all in-game elements.

**Features**

***Pausing, restarting, exiting:*** The game includes a pause functionality when the key “P” is pressed. It can be resumed the same way. The game can be quit at any time once “ESC” is pressed. When the game is over (lives <= 0), the player will be prompted to either press “R” to restart or press any other key to quit.

***Increasing Difficulty:*** The game has an increasing degree of difficulty. As players progress, the snake moves faster and there will be a random snake that appears once the score is 5 or more. If the play snake collides with the random snake or if the random snake collides with any part of the player snake, the player snake’s life is reduced by one, and the segment is reduced by one.

***Adding Lives.*** The snake game dynamically draws a `specialfood` that is represented by a heart. This heart appears in a special condition as mentioned above, and if the player snake eats this heart, lives will be increased by 2.

**Planned Future Imporvements**

***Intelligent AI Opponent.*** It is planned to integrate a Q-learning or deep reinforcement learning algorithm to simulate a smart AI opponent that learns from the player's moves and adapts its strategy over time. The AI opponent will learn the player's behavior and develop tactics to challenge the player.

***More Random Snakes.*** It is planned to integrate a menu where users can choose difficulty level and game types (classic, with AI, etc). This proved challenging to realize given the amount of time available and the basic libraries used.

***Customization.*** It is planned to make the game more interactive by allowing the user to determine parameters like speed, size, colors.

***Advanced score and high score tracking.*** A more advanced score system could be implemented that includes levels, and high score management by saving previous high score to file.

***More power-ups and special items.*** It is possible to add more power-ups and special items beyond the special food.

***Parallel Processing***. Although it is possible to use multi-threading by uncommenting some lines in the code, the synchronization of the game is not perfect and needs more thought. It is planned to achieve this through time.

**References**

*DevDocs*. (2023). Retrieved from C++17 documentation: https://devdocs.io/cpp/

*Double-ended queue*. (2023). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Double-ended\_queue

educ8s. (2023). *Cpp-Retro-Snake-Game-with-raylib*. Retrieved from Github: https://github.com/educ8s/Cpp-Retro-Snake-Game-with-raylib

(n.d). Retrieved from Raylib: https://www.raylib.com/

ProgrammingKnowledge. (2021). *Creating Simple Snake Game in C++*. Retrieved from Youtube: https://www.youtube.com/watch?v=AxrQje7V65o