**INTRODUCTION**

Purpose

Our main aim through this project was to develop a Snake Game comprising two players, to learn some skills of programming and eventually to grasp the idea of object oriented programming via C++. The major motive of the project was that it should provide entertainment to the users, furthermore it would help us with enhancing our programming skills.

Concept of Game

The game “SARPA” is simply a modified form of the traditional snake game that we are very familiar with. We discussed in group as how to make our project more effective that it would attract the users and even would not let their expectations down. Irrespective to the traditional snake game it has two snakes controlled by two players with the same keyboard. The game is a survival between two snakes where each compete against the other to eat the greater number of foods and beat the opponent. To make the game a little more exciting there are three levels with certain obstructions on each. The winner is evaluated on the basis of his/her three level game records.

Objectives

The project was carried out with the following objectives

* To introduce an entertaining dual player interface
* To modify the traditional snake game with additional interesting features
* To understand the concept of graphics designing and modular programming
* To enhance programming skills through teamwork

Features

We believe the game “SARPA” is superior to the traditional snake game due to its following features:

* It includes all the basic features of the traditional snake game.
* The basic rules and concept are easy to understand so the game is compatible with people of all ages.
* It makes an involvement of two players at an instant thus helps in socializing.
* Since the game is a survival between two which makes it exciting to play.
* The background music triggers the mood for the players.
* The additional obstacle feature makes the game more challenging.

Tools used

Following tools were used for the project building

* Code::Blocks 16.01.0.0
* SFML graphics library

# Software Development:

The project uses the following classes:

1. Program Flowchart:

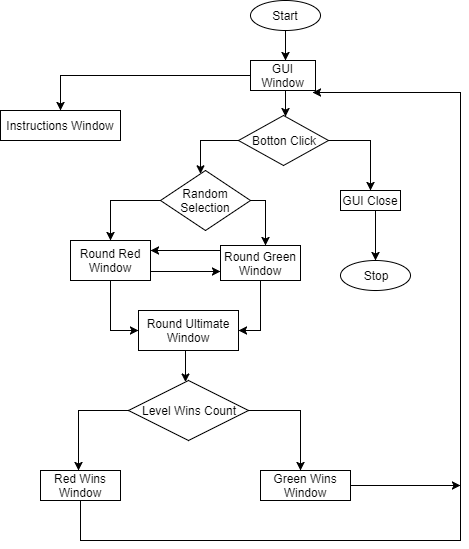


Figure: Program Flowchart

1. Screen shots:



Figure: GUI Window

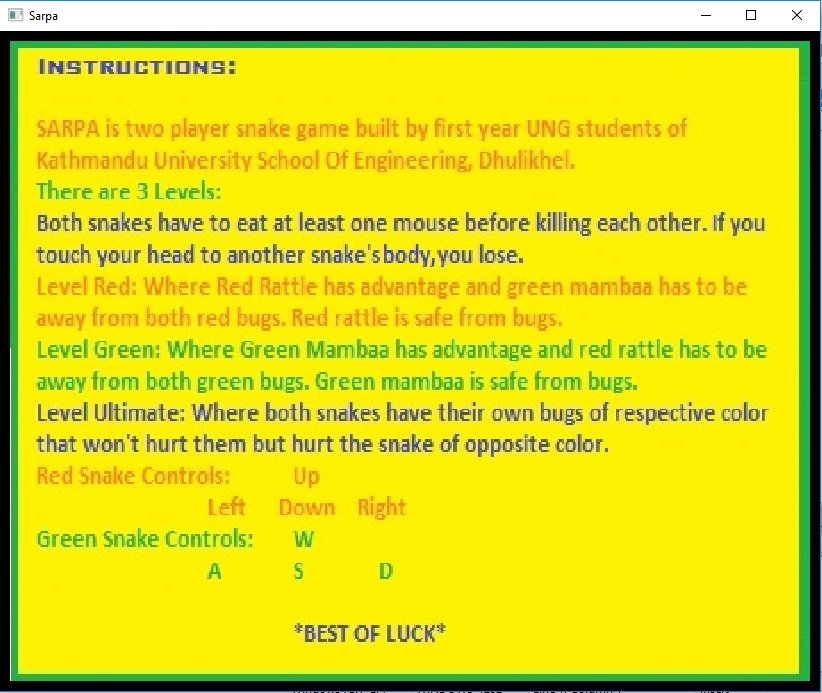


Figure: Instructions Window

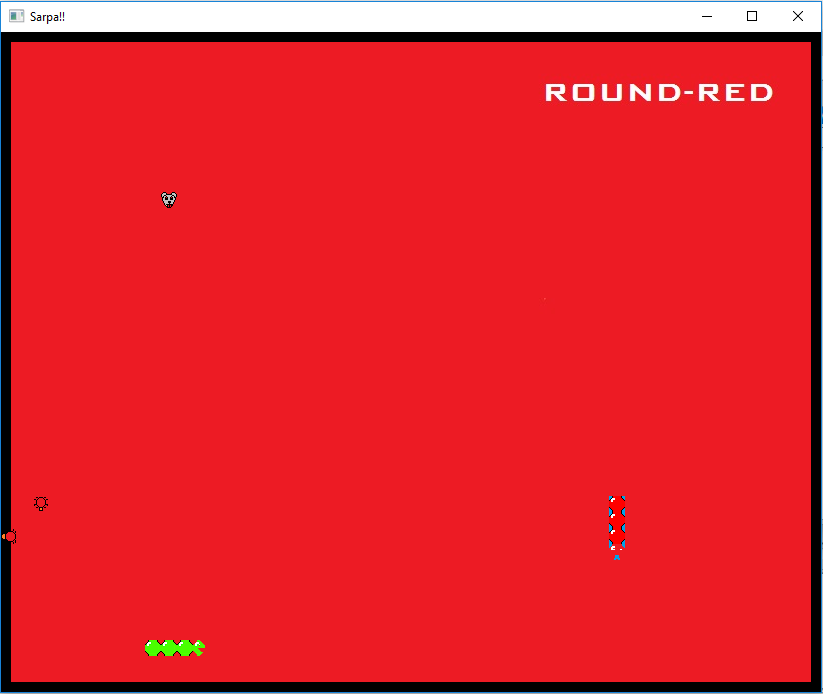


Figure: Round Red Window (After GUI red)

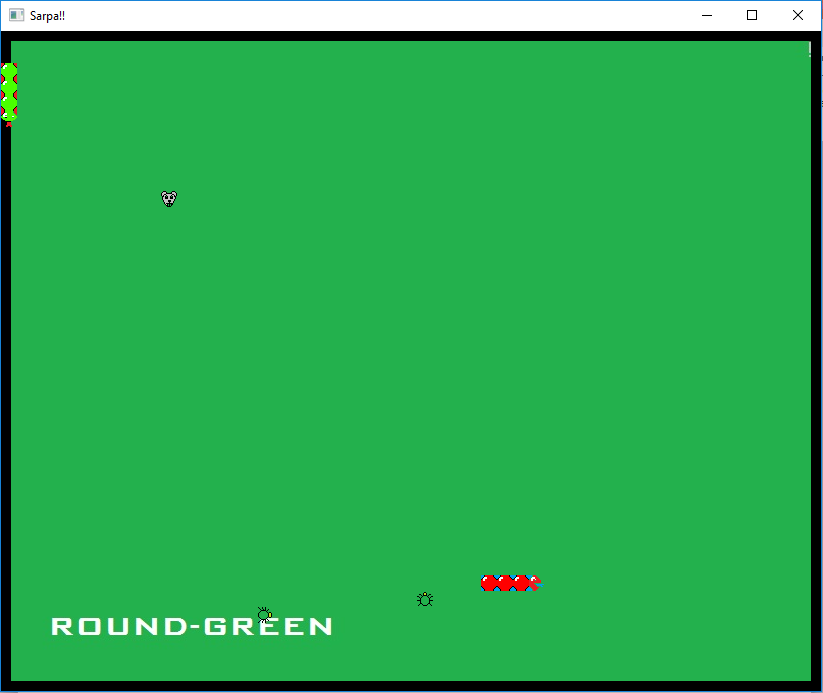


Figure: Round Green Window (After GUI green)

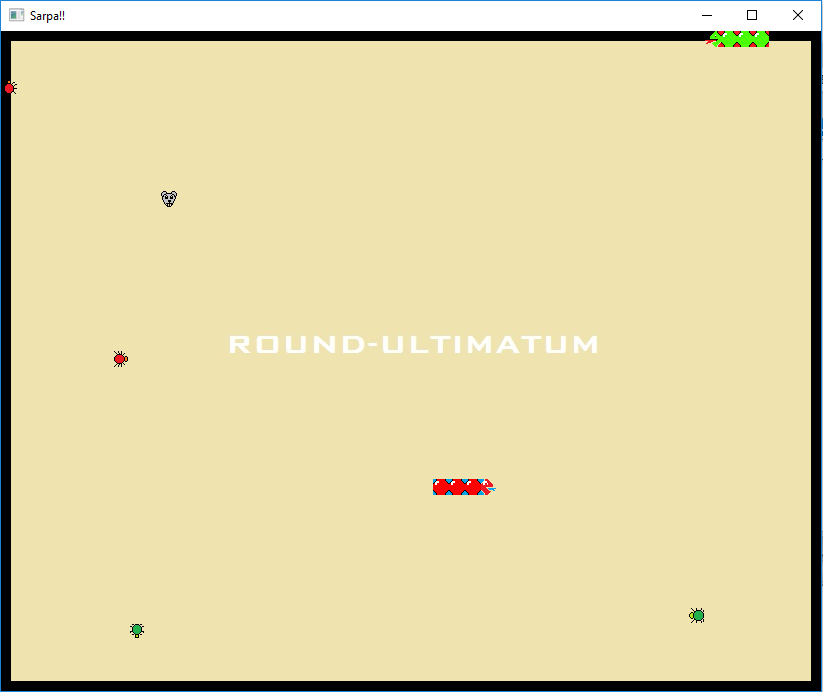


Figure: Round Ultimate Window (After GUI)

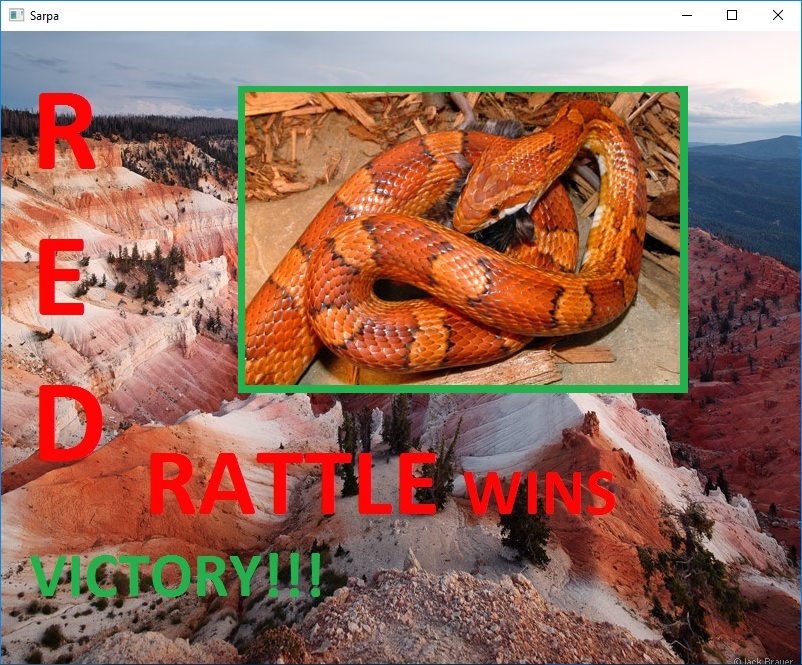


Figure: Red Rattle Wins



Figure: Green Mambaa Wins

Structures and Functions:

OpenFromFile():

It opens the music from respective file.

* setLoop(boolean):

It takes true or false as an argument and loops the music file or not.

* setVolume(int):

It provides the volume to the music.

* play():

It plays the music audio.

* Sprite name(&Texture):

It initializes the texture background to the sprite.

* isOpen():

It opens the user-defined window.

* pollEvent(&Eventname):

It processes the user-defined event in the window.

* close():

It closes the window after all the events are over.

* setPosition(int,int):

It initializes the position of the sprite taking position numbers in pixels as arguments.

* draw():

It draws the sprite in the window.

* getGlobalBounds():

It takes the position of the movement.

* contains(event.x-coordinate,event.y-coordinate):

It calculates if the cursor position is under getGlobalBounds.

* isButtonPressed():

It detects the button pressed on mouse.

* isKeyPressed():

It detects the key pressed on keyboard.

* display():

It displays the window.

User-Defined:

* Clear():

It is a void function that initializes all the variables required during different stages of program back to zero.

* Snake{}:

It creates x and y coordinate for each block for snake for initializing position.

* Food{}:

It creates x and y coordinate for each mouse aka food.

* SnakeCollision():

It detects collision of snake among each other.

* Move():

It provides a movement for each cube of snake and detects self collision.

* red():

It opens a new window that displays “Red Rattle Wins”.

* green():

It opens a new window that displays “Green Mambaa Wins”.

* bugs():

It creates the bug with x and y coordinates for initializing position.

* preys():

It provides the prey movement.

* preycollision(), preycollisionred() and preycollisiongreen():

They detect when snakes collide with bugs and return integer to count which player wins.

* newwindow(),newwindowred() and newwindowgreen():

They open different windows for different game levels of ultimate, red and green rounds respectively.

* Instructions():

It opens new window for providing help and instructions before game.

**Discussion:**

This project game successfully provides a two player experience never found in previous ‘Snake’ games. It also provides three levels with advantages on either sides. However, the project can be improved upon with multiplayer mode, high-score management, controls and choosing different avatars for snakes to add on the user experience. Any recommendations from the users to make the game more entertaining are highly encouraged.

Contact Information:

Names: Prajwol Lamichhane, Arpan Mahatra and Abhay Raut Chhettri

Address: 28 Kilo, Kavre

Phone Numbers: 9841544244, 9843436289, and 9845792748

Emails: [prajwollamichhane11@gmail.com](mailto:prajwollamichhane11@gmail.com)

[arpanmahatra@gmail.com](mailto:arpanmahatra@gmai.com)

[Abhayraut712@gmail.com](mailto:Abhayraut712@gmail.com)

**RELATED WORK**

* Traditional Snake game

***Snake*** is the common name for a videogame concept where the player maneuvers a line which grows in length, with the line itself being a primary obstacle. The concept originated in the 1976 [arcade game](https://en.wikipedia.org/wiki/Arcade_game) [*Blockade*](https://en.wikipedia.org/wiki/Blockade_(video_game)), and the ease of implementing *Snake* has led to hundreds of versions (some of which have the word *snake* or *worm* in the title) for many platforms. There are over 300 for [iOS](https://en.wikipedia.org/wiki/IOS) alone.

After a variant was preloaded on [Nokia](https://en.wikipedia.org/wiki/Nokia) [mobile phones](https://en.wikipedia.org/wiki/Mobile_phone) in 1998, there was a resurgence of interest in the snake concept as it found a larger audience.

The gameplay consists of a player who controls a dot, square, or object on a bordered plane. As it moves forward, it leaves a trail behind, resembling a moving snake. In some games, the end of the trail is in a fixed position, so the snake continually gets longer as it moves. In another common scheme, the snake has a specific length, so there is a moving tail a fixed number of units away from the head. The player loses when the snake runs into the screen border, a trail or other obstacle, or itself.

* Slither.io

***Slither.io*** is a [massively multiplayer](https://en.wikipedia.org/wiki/Massively_multiplayer_online_game) [browser game](https://en.wikipedia.org/wiki/Browser_game) developed by Steve Howse. Players control an avatar resembling a snake, which consumes multicolored pellets from other players, and ones that naturally spawn on the map in the game to grow in size. The objective of the game is to grow the longest snake in the server. *Slither.io* is similar in concept to the popular 2015 web game [*Agar.io*](https://en.wikipedia.org/wiki/Agar.io) and is reminiscent of the classic arcade game [*Snake*](https://en.wikipedia.org/wiki/Snake_(video_game)).

The objective of the game is to control and move a snake around a colored area, eat pellets to gain mass, defeat and consume other players to grow the biggest and longest in the game. If the player's snake's head collides into a part of another snake, the player loses the game and must start over. The defeated avatar's body turns into bright, shining pellets for other players to consume. These pellets that remain from "death" of an avatar will correspond to the color of the avatar itself, and are both brighter and bigger than normal pellets.

Pellets also spawn from other snake avatars. By either pressing the space bar or clicking on the mouse or trackpad, the player can activate "boost mode", which causes the avatar to speed up. When a player uses "boost mode", the snake loses some mass, causing the snake's size to shrink slightly, with the mass that is lost from the boost appearing as a line of dots where the boost was used. This feature is useful to outmaneuver and defeat opponents. Another strategy that players use to defeat opponents is coiling around them in a loop until the opponent, trapped in the loop, crashes into the player.

There is a border that confines avatars within the circular game board. If a snake hits the border, the player automatically dies without turning into the aforementioned pellets. As of 2016, the player with the biggest snake at the end of the day gets to share a "victory message" with the world.

# Bibliography:

* [www.sfml-dev.org](http://www.sfml-dev.org/)
* [www.github.com](http://www.github.com/)
* [www.gamecodeschool.com](http://www.gamecodeschool.com/)
* [www.wikipedia.org/wiki/Snake\_(video\_game)](http://www.wikipedia.org/wiki/Snake_(video_game))
* [www.stackoverflow.com](http://www.stackoverflow.com/)
* [www.cplusplus.com](http://www.cplusplus.com/)