

SNAKEZZZ...

(A CS101 Course Project)

By:

Utkarsh Gautam

Arushi Bansal

Surbhi Sahu

Anay Tripathi

CONTENTS

1. ACKNOWLEDGEMENTS

2. ABSTRACT

3. INTRODUCTION

4. BACKGROUND & ANALYTICAL CONCLUSIONS

5. ALGORITHM & FUNCTIONS

6. EXPECTATIONS MET & SCOPE FOR IMPROVEMENT

6. CONCLUSION

7. BIBLIOGRAPHY

ACKNOWLEDGEMENTS

We would like to thank Prof. Deepak Phatak and Prof. Supratik Chakraborty for their guidance and encouragement. Thanks and appreciation is also due to our TA Mr. Prasanth Presannan for his constructive suggestions, help in rectifying the errors and his valuable critics during the planning of the project. We would also like to thank all other TA's whom we came across informally at various points who also gave their ideas, suggestions and inputs.

The Team

ABSTRACT

In this project we have created a game called "Snakezzz...". It is just like one of the common gaming applications found in cell phones. The user moves the snake in a maze in order to eat the food appearing on the screen to gain points while avoiding collision with walls and itself which leads to death. The game has been created using C++.

The maze has been created using simplecpp. The snake is a series of circles whose coordinates are stored in a circular array. To move and turn the snake coordinates of tail character are modified and placed in front of head. The score increases with time and every time the food is consumed. Also, length of snake increases every time food is consumed, thereby increasing difficulty with time. Hence, a challenging yet refreshing game is created for the user.

INTRODUCTION

This project is an attempt to make the popular game “SNAKE” using C++ language. With the use of simplecpp library the graphical notion of the game has been accomplished.

The game has multiple mazes and difficulty levels which user can choose and play accordingly.

The movement of the snake is decided by the user using controls as mentioned in user manual. It can move up, down, left or right as per the user commands.

The snake goes around an arena bounded by the walls and having some obstructions in between which will block its way. Red colored food items appear periodically in the game . On consuming them the score increases but the length of the snake will also increase which will increase difficulty level of the game.

Points will be on the basis of how long the snake lives and how much food it eats.

BACKGROUND & INITIAL ANALYTICAL PROCESS

Background

We had to make a programming project for the course CS101. Various options were provided to us by our course instructor. After a lot of discussion, the team decided to make game snake as it seemed fun while providing sufficient challenge and scope for learning, which got the approval of our TA. The work was started as soon as possible.

Initial Analytical Process

Till first few meetings the core discussions was on the movement of snake and its representation.

Initial Ideas -Based on what we had learnt in the course so far, it was thought to first represent the snake as an array of x and y

coordinate (nx2)array, which had to be changed according to its moving directions.

**n is the length of snake.*

For moving, the idea was to change each coordinate as per the user commands e.g. to move left: decrease x coordinate of each element of array by one and continue doing it unless some new instruction from user is not commanded.

{ALGORITHM:

Move the head of the chain as instructed by the user and run a loop so that each i'th element of snake will occupy the coordinates of (i-1)th element . Run a forever loop which will follow a command unless some new instruction is not passed and as soon as some new instruction is passed change accordingly.

}

Simultaneously check that the head of the user (coordinates of head of the snake) doesn't coincide with the wall or any other obstruction.

We decided to make functions for movement of snake (turn left or right). Another parameter viewer direction will give idea how the left or right instruction would be executed on the snake. e.g. if initially snake is moving in right direction(variable viewer direction will be initialized R, if further user instructs to move say 'Left', the y coordinate of head of snake would be incremented and viewer direction would turn to U(up)).

Function calls would be made to make a movement successful .

With not much knowledge of graphics all these ideas were formulated so the team didn't had much idea how practically it will work.

In our further meetings we felt that the program was going to be extremely slow because very large number of checks are being performed, and then the next movement is decided.

Also the algorithm for movement of snake failed as it created errors and complications in test cases of multiple turns.

A temporary improved algorithm on movement of snake which was thought

{ALGORITHM

As soon as user makes a change in movement that coordinate will be recorded each part of snake passing through that point will follow that particular direction.

Change of coordinates will be according to the previous algorithm.

}

This facilitated turns of snakes but was soon disregarded as number of movements was not fixed so those specific points couldn't be stored as such. Also number of steps in every movement of snake was anyhow increasing which slowed down the programmed.

The forever loop idea also failed since we didn't knew how to write a code such that user can anytime enter a command and movement can change accordingly.

On further discussion with some TA's including our own, we reached a conclusion that for simultaneous movement of snake with death checks and food appearance and checking

for user command prompt, we may have to use multithreading. However, on consulting our professor about the same,he told us that simplecpp has features of non-blocking input.

Also, once we had learnt OOPs, we managed to simplify our program and make it much neater.

FINAL ALGORITHMS

AND FUNCTIONS

Movement of Snake :[void move();]

The function “void move()” has been created which will move the snake according to the command given by the user. The user gives command through keyboard buttons: ‘w’, ‘s’, ‘a’ and ‘d’ to move the snake up, down, left or right respectively.

The algorithm is based on the principle of circular array. The last element of snake’s body will be redefined w.r.t the first one and then the last element would become the new first element.

e.g. for a snake moving right, the tail element:

$((\text{head} + \text{length} - 1) \% \text{length})$ ’th element

would be redefined as $(\text{headX} + 1, \text{headY})$. And now, this would be the new head. Now, the user gives a command to move up, tail element would be redefined as $(\text{headX}, \text{headY} + 1)$.

Snake (circular array)

An array of circles was created dynamically, named “snake”. These circles form the body elements of the snake and were placed adjacent to each other.

Random Appearance of Food [double foodx() & double foody()]

The random function already included in simplecpp library was used to specify the random coordinates of the food. Its limits were set to keep it from appearing outside the boundary of the maze or on the obstructions. The food will disappear and reappear at different locations if not eaten in a limited time.

Class Buttons

It was created to facilitate the user in choosing options between play/help/choosing a maze/level etc.

Maze Designing[class maze_1{ }; class maze_2{ }; class maze_3{ };]

All the mazes will be predesigned and the user will have an option at the start to choose one of them. They will have obstructions typical shapes to make the game tough for the user.

LEVELS

The user has the option to choose from three levels: Easy, Normal & Tough. The levels vary in speed of snake and frequency of disappearance of food.

Distance Function [double dist()]

This function was created using the `sqrt()` function from `<cmath>` library to detect click on button.

CHECK FUNCTIONS [bool checkmoves() & void checkfoodeaten()]

bool checkmoves() has been made to be passed in the while loop of the main program which ends the game if the coordinates of the head of the snake coincides with that of walls, obstructions or any other part of its own body.

void checkfoodeaten() increases the score and the length of the snake if it eats the available food.

EXPECTATIONS MET

& SCOPE FOR

IMPROVEMENT

- The program created meets performs most of the defined functions. Only the display of mazes couldn't be achieved properly as the compiler refused to recognize the Rectangle and Line classes pre-included in simplecpp library for a reason neither we nor any ta could explain. Eventually it worked when we inherited Rectangle in our class maze, but Line gave trouble on inheritance. Finally the maze wasn't displayed. It just blinked.
- The new circle element on incrementing the length is white in color and is not getting recolored.
- We thought of adding a pause/resume feature at the last moment but could not debug it entirely due to shortage of time.
- Files could be used to store the Highscores from previous games.

- User could also be able to continue a paused game from the last time he/she had played.
- We could add some extra bonus items to be consumed by snake which temporarily give it some special powers like collision immunity , speed etc.

CONCLUSION

So, we have created an interesting and dynamic game: "Snakezzz..." using what we learnt in the course CS101. Making this project has given us some real-life software development experience and we found this project to be one of the best parts of the course. We learnt much more than what we had learnt in the class, it has given us an opportunity to further explore the world of computer programming and realize that we have a long and interesting journey ahead of us in the world of programming.

BIBLIOGRAPHY

- **An introduction to programming through C++ *by* Abhiram G. Ranade**
- **Lectures on simplecpp graphics *by* Prof. Abhiram G. Ranade**
- **www.cplusplus.com**
- **www.google.com**

