**SNAKY ADVENTURES**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfilment of the requirements to award the degree of

**Bachelor of Technology/Master of Technology**

In

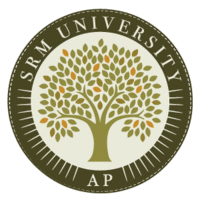
**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**V Harsha Vardhan**

**(AP22110010677)**



Under the Guidance of

**(MS Poonam Yadav)**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[December , 2023]**

# Certificate

Date: 1-Dec-23

This is to certify that the work present in this Project entitled “**SNAKY ADVENTURES”** has been carried out by **V Harsha Vardhan** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in **School of Engineering and Sciences**.

## Supervisor

(Signature)

MS Poonam Yadav

Designation,

Affiliation.

## Co-supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

**Acknowledgements**

I would like to introduce my c++ project

SNAKY ADVENTURES.

Under the guidance of MS POONAM YADAV.

I am really grateful to her for providing me with such a great and interesting subject for our project.

I also want to express my gratitude to my professor

Without their advice and feedback, this project wouldn't have gotten off to such a great start.

My hope is that you will all enjoy and appreciate my efforts.

# Table of Contents

|  |  |
| --- | --- |
| Certificate |  |
| Acknowledgements |  |
| Table of Contents |  |
| Abstract |  |
| Abbreviations |  |
| List of Tables |  |
| List of Figures |  |
| 1. Introduction |  |
| 1.1 Background |  |
| 1.2 goals  1.3 project scope |  |
| 2. Methodology  2.1 |  |
| 2.1.1 design  2.1.2 implementation  2.2 tools and technology |  |
| 3. Discussion |  |
| 4. Concluding Remarks |  |
| 5. Future Work |  |

# Abstract

|  |
| --- |
| The code provided is a Snake game written in C++. It uses the turbo c++ Graphics Interface to draw the game on the screen. The game is played by moving a snake around the screen using the arrow keys. The goal is to eat as many eggs as possible without hitting the snake's own body. If the snake hits anything, the game is over. The code works by first initializing the graphics and creating a new Snake object. The Snake object contains all of the information about the snake, such as its position, length, and direction of travel. The main loop of the game then begins. In each iteration of the loop, the Snake object is moved and the game is checked for collisions. If there is a collision, the game is over. Otherwise, the game continues until the player quits. This a game where player controls a snake and tries to eat as many eggs as possible without running into snake's own body. |

# Abbreviations

# p1,p2 – Represents the X and Y - coordinate of the snakes HEAD

# v1,v2 - Represents the X and Y - coordinate of the snakes TAIL

# e1,e2 - Represents the X and Y - coordinate of the EGG POSITION

# prev - Represents the PREVIOUS DIRECTION of the movement for the SNAKE

# now - Represents the CURRENT DIRECTION of the movement for the SNAKE

# n - LENGTH of the snake

# colr - Represents the COLOR used to draw the snake on the screen

# dsp - Represents the DISTANCE between successive segments

# dly - Represents the DELAY OR SPPED of the snakes movement

# stp - Represents a flag/ control variable influences the games loop’s

# egGen – Represents the control variable related to generate a egg

xr - Represents the RADIUS of the circular segments(eggs)

yr - Represents the other RADIUS used in conjunction with xr(eggs, snakes body)

scr - Represents the score

strX , strY - Represents the starting X and y – coordinate or the LEFT AND TOP MOST BOUNDARIES of the game area

endX , endY - Represents the starting X and y – coordinate or the RIGHT and BOTTOM most BOUNDARIES of the game area

init - Function responsible for initializing game settings and graphical elements

gnrtCond - stands for "generate conditional." It is responsible for generating the next position of the snake's head based on the current direction of movement (now) and the previous direction (prev).

gnrtUncond - stands for "generate unconditional." It is responsible for unconditionally generating the next position of the snake's head based on the previous direction (prev).

chngDir - stands for "change direction." It is responsible for changing the direction of the snake based on user input

List of Tables

TABLE 1: EXAMPLE FOR CONTROL MAPPING TABLE

Key Action

Up Move Up Direction

Down Move Down Direction

Left Move Left Direction

Right Move Right Direction

e End and Exit Game

TABLE 2: EXAMPLE FOR GAME STATISTICS TABLE

Player Score Level Time Played

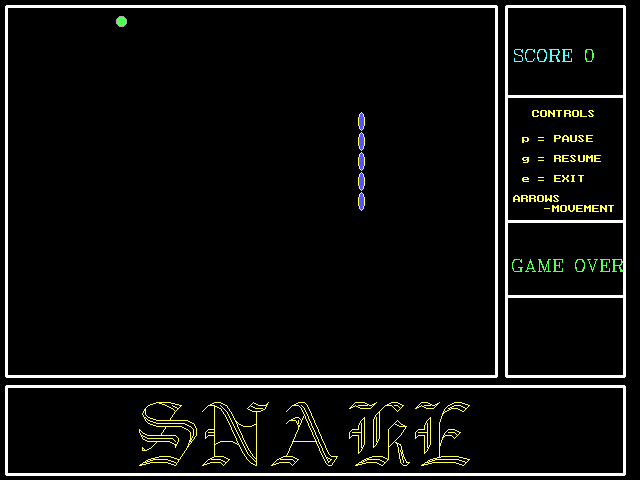
Player1 350 5 12:30

Player2 240 3 8:45

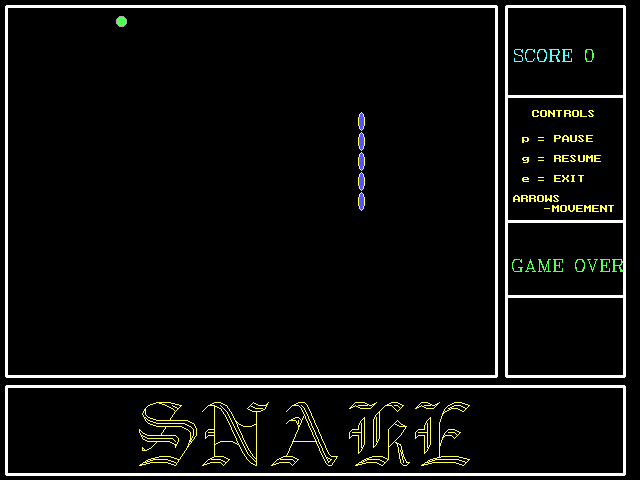
Player3 480 7 16:10

# List of Figures

SNAKE



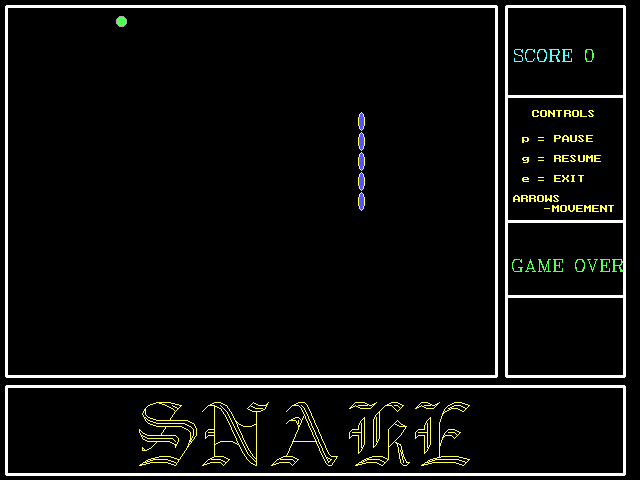
USER INTERFACE



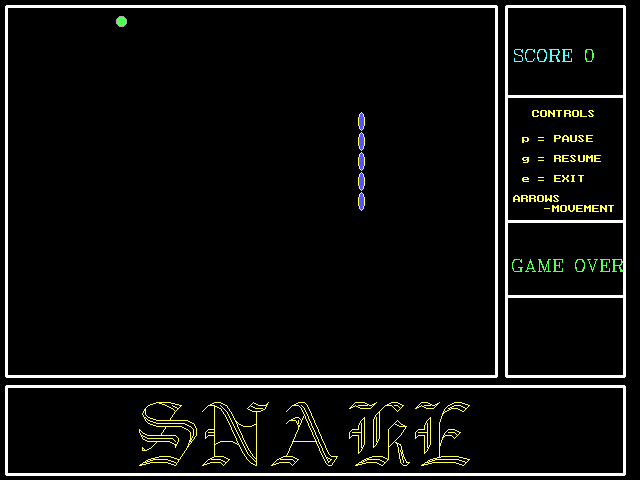


EGG

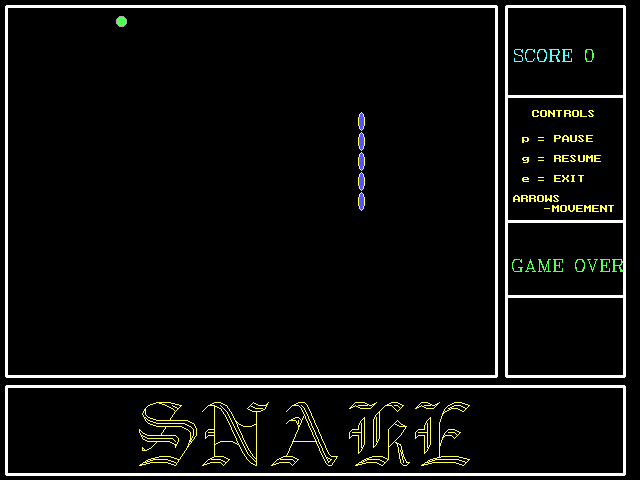




SCORE



CONTROLS



1.Introduction

The Snake Game project is a classic arcade style game. The game was developed with the aim of creating a nostalgic and fun gaming experience with programming techniques and graphical elements.

1.1 Background

The idea for Snake Game was born from the popularity of retro-video games and a desire to explore fundamental concepts of game development. Inspired by the classic Snake game, this project aimed to recreate and improve the gameplay while introducing modern features and graphical elements.

1.2 Goals

The main objectives of the project are:

Implement a functional and interactive snake game with simple controls. Enhance your gaming experience with graphical elements and animations. Incorporate scoring mechanics and dynamic difficulty. Demonstrate knowledge of C++ programming and graphics processing.

1.3 Project scope

The scope of the Snake Game project includes various aspects such as:

Core game mechanics such as snake movement, egg creation, and collision detection. Graphical elements that enhance the visual appeal of the game. User interface design for a seamless gaming experience. Implementation of scoring and difficulty progression.

METHODLOGY

The development of the Snake Game project followed an iterative and step-by-step approach based on agile methodology. The project lifecycle included the following major phases:

2.1.1 Design

The design phase includes creation of detailed design specifications. DFD diagrams are designed to show the relationships between various components, with an emphasis on modularity and extensibility.

2.1.2 Implementation

The actual implementation of the Snake Game was performed using the C++ programming language. Game logic, graphical elements, and user interface components were coded according to design specifications. The Graphics.h library was used to handle graphics operations.

2.2 Tools and technology

The following tools and technologies were used in the development of Snake Game:

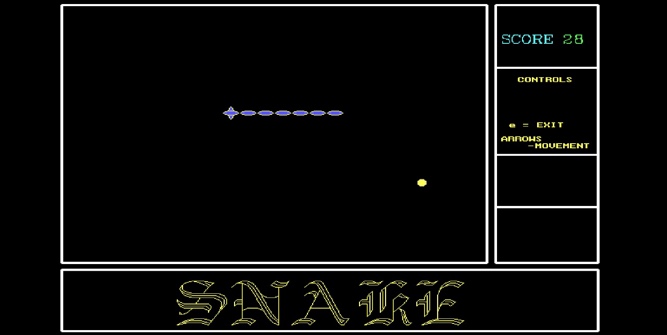
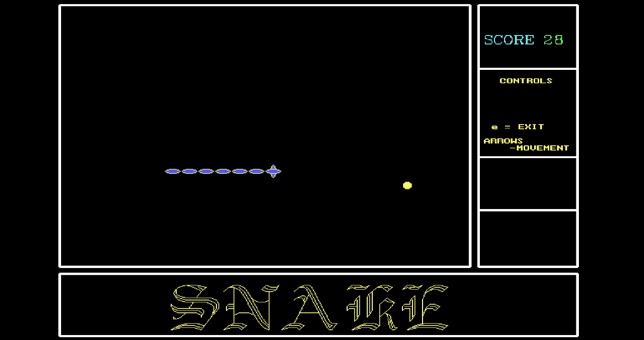
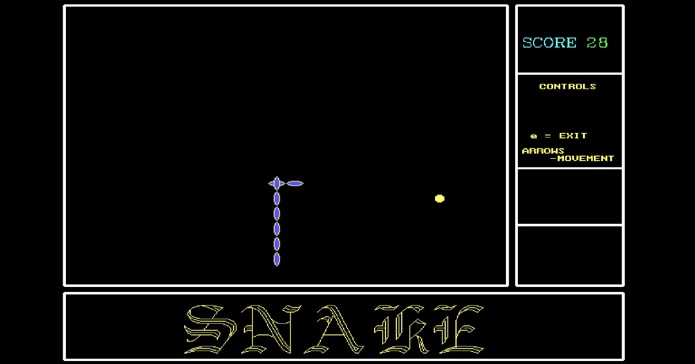
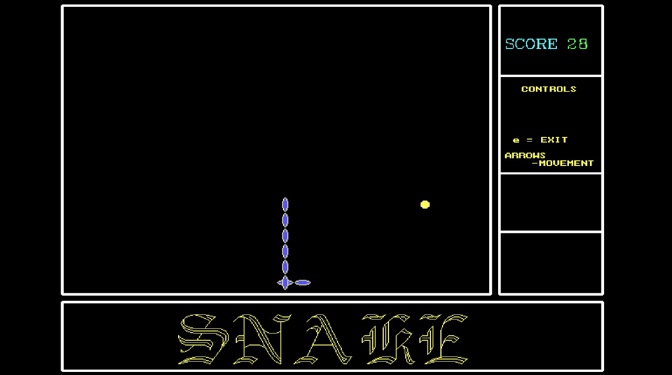
C++ programming language: chosen for its versatility and efficiency when processing game logic. Graphics.h library: Used for graphics manipulation and rendering. Turboc3 IDE: Selected as integrated development environment due to compatibility with Graphics.h.

DISCUSSION

3.1 Game Mechanics and User Experience

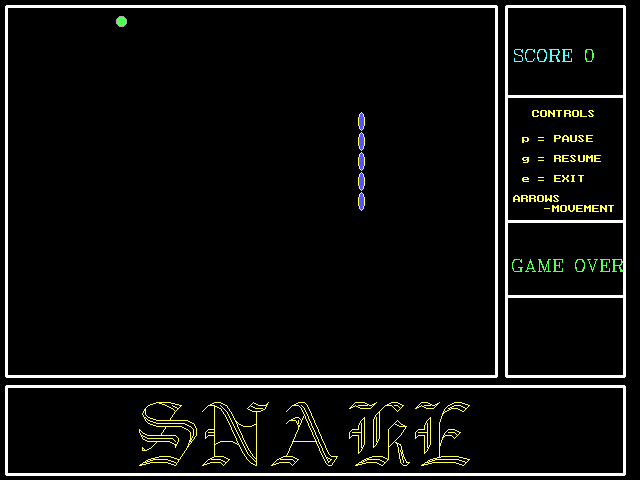
3.1.1 Snake Movement and Controls

The user can simple Ly navigate the snake using arrow keys, providing a classic and familiar gameplay experience.



3.1.2 User Interface Design

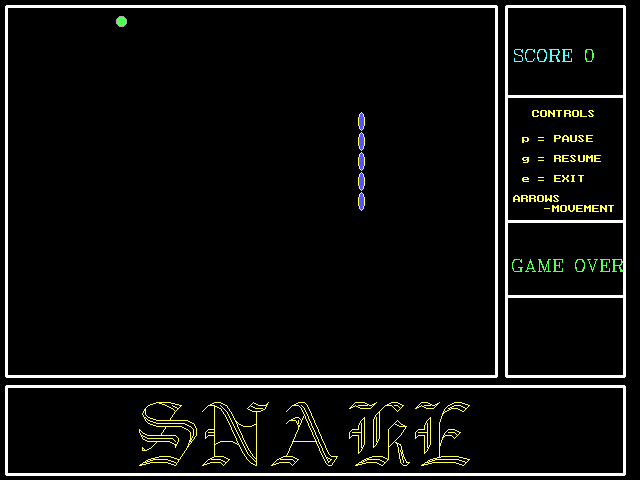
The user interface design successfully balances simplicity with functionality. Clear score display, game controls, and vibrant graphical elements contribute to an engaging user experience.

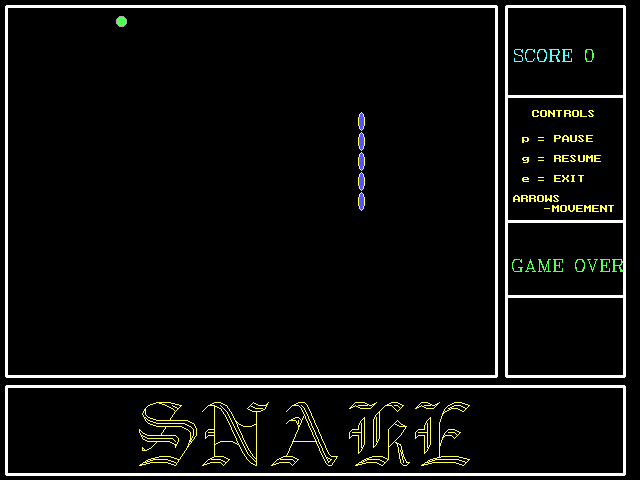
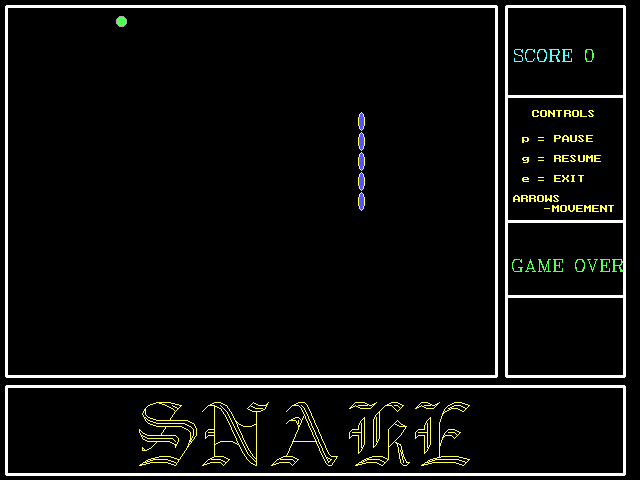
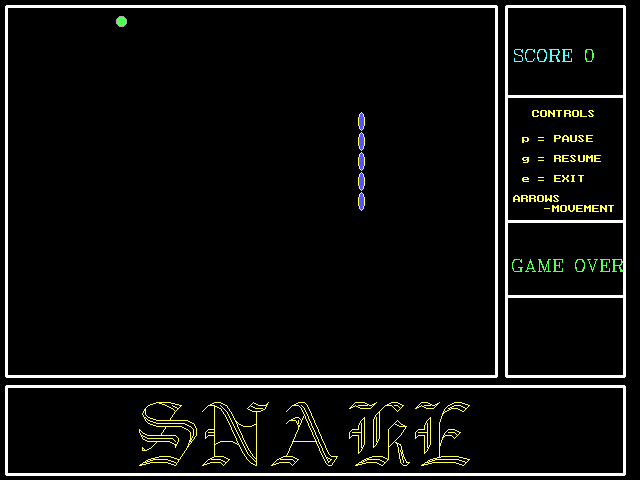
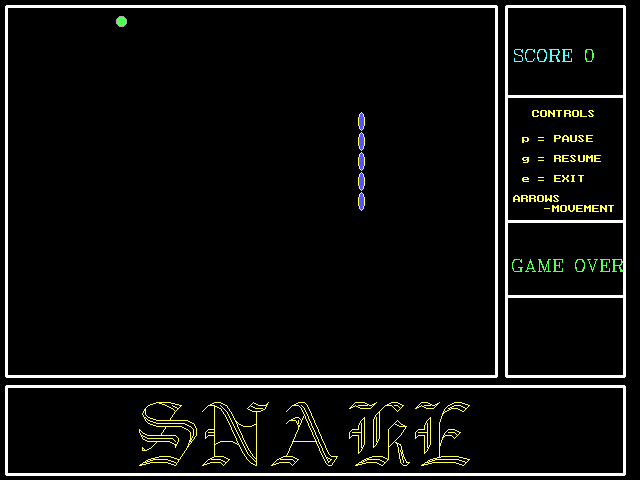


3.2 Graphical Elements

3.2.1 Integration of graphics.h Library

The graphics.h library facilitated the incorporation of graphical elements seamlessly. Graphics, such as the snake, eggs, and background, enhance the visual appeal of the game.



3.2.2 Scoring

The scoring mechanism successfully rewards players for sustained gameplay. The score increases with time, creating a sense of achievement and progression.

Initialization:

The initial score is set to 0.

The scoring system is displayed on the screen.

long scr; // Score variable

Score Display:

The score is displayed on the screen using a graphical library.

char \*p;

ltoa(scr, p, 10);

settextstyle(8, 0, 1);

setcolor(0);

outtextxy(585, 40, p);

Scoring Function:

The score function updates the score based on the game's conditions.

It increases the score by a certain amount when an egg is consumed.

The scoring rate is adjusted to increase the challenge over time.

void Snake::score()

{char \*p;

ltoa(scr, p, 10);

settextstyle(8, 0, 1);

setcolor(0);

outtextxy(585, 40, p);

if (egGen != 1)

{

scr = scr + dly / 10;

}

Dynamic Scoring:

The score is dynamically increased over time by a fraction of the delay (dly) value.

scr = scr + dly / 10;

3.3

**HEADER FILES USED:**

* + **iostream.h**: This header is related to input and output streams.
  + **dos.h**: This header provides various DOS-related functions and structures.
  + **stdlib.h**: Standard Library header for general-purpose functions.
  + **time.h**: Standard Library header for date and time functions.
  + **string.h**: Standard Library header for string manipulation functions.
  + Conio.h: set of functions that are used for console-based input and output operations
  + Graphics.h: provides a simple graphics programming interface for drawing basic shapes, text, and on a graphics window
  + Stdio.h: provides a set of functions for input and output operations.

CONCLUDING REMARKS

In conclusion, developing the Snake Game project has been a rewarding and enlightening journey. This project achieved its main goal of recreating a classic arcade game in a modern style, integrating graphical elements, and implementing dynamic gameplay features.

Intuitive gameplay:

Intuitive controls and implementation of classic snake movements contribute to an attractive and comfortable user experience.

Graphics improvements:

By integrating the Graphics.h library, we were able to create visually appealing game elements that liven up the entire game environment.

Rating Mechanism:

A dynamic rating mechanism motivates players to achieve higher scores over time, giving them a sense of accomplishment.

Advanced features:

Discover additional features such as different game modes, power-ups, and multiplayer features to further diversify your gaming experience.

Optimizations:

Perform further optimizations to improve code efficiency, improve graphics, and ensure compatibility with a wider range of platforms.

FUTUR WORK

Although the Snake Game project has achieved its main goals, there are several opportunities for future work and improvements to further improve the game experience.

5.1 Power-ups and special abilities

Gameplay can be expanded to include strategic elements by incorporating power-ups and special abilities.

Examples include:

Speed ​​boost:

Temporarily increase the snake's speed to pass through difficult sections.

Invisible:

Makes the snake temporarily invisible to overcome obstacles.

5.2 Multiplayer features

Implementing multiplayer functionality could improve the social aspects of the game. Competitive multiplayer mode:

Players can compete against each other in real time.

5.3 Mobile version

Developing a mobile version of the Snake game could appeal to a wider audience. Customize the interface, controls, and graphics for your mobile device.

\*\*\*