VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018

A Computer Graphics Mini Project Report

on

" 2D Snake Game "

Submitted in partial fulfillment of the requirements for the VI semester and award of the degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi

Submitted by:

Nitish K	*1RN19CS092*
Rajatha Bangera	*1RN20CS411*

Under the Guidance of:

Ms. Mamatha Jajur
Assistant Professor
Dept. of CSE

Department of Computer Science and Engineering
(NBA Accredited for academic years 2018-19, 2019-20, 2020-22)
RNS Institute of Technology
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098

2020-2021

RNS Institute of Technology

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(NBA Accredited for academic years 2018-19, 2019-20, 2020-22)

CERTIFICATE

Certified that the mini project work entitled **"2D Snake Game"**has been successfully carried out by **"Nitish k** bearing USN **"1RN19CS092"** and **"Rajatha Bangera"** bearing USN **"1RN20CS411"**, bonafide students of **"RNS Institute of Technology"** in partial fulfillment of the requirements for the 6th semester of **"Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University"**, Belagavi, during academic year 2021-2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the CG laboratory requirements of 6th semester BE, CSE.

Signature of the Guide	Signature of the HoD	Signature of the Principal
Ms. Mamatha Jajur	**Dr. Kiran P**	**Dr. M K Venkatesha**
Assistant Professor	Professor and Head	Principal
Dept. of CSE	Dept. of CSE	

External Viva:

Name of the Examiners — Signature with Date

1.

2.

Acknowledgement

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We are grateful to Management and **Dr. M K Venkatesha**, Principal, RNSIT, Bangalore, for his support towards completing this mini project.

We would like to thank **Dr. Kiran P**, Professor & Head, Department of Computer Science & Engineering, RNSIT, Bangalore, for his valuable suggestions and expert advice.

We deeply express our sincere gratitude to our guide **Ms. Mamatha Jajur** , Assistant Professor, Department of CSE, RNSIT, Bangalore, for her able guidance, regular source of encouragement and assistance throughout this project.

We would like to thank all the teaching and non-teaching staff of Department of Computer Science & Engineering, RNSIT, Bengaluru-98 for their constant support and encouragement.

Abstract

Snake game is a computer action game whose goal is to control a snake to move and collect food in a map. The player controls a long creature resembling a snake which roams around on a bordered plane picking up food. The player must try as much as possible to avoid hitting its own tail or the edges of the playing area. Each time the snake eats a piece of food, its tail grows longer. The food is randomly placed anywhere in the playing area.

We made use of Microsoft Visual Studio 2019 with opengl files and codes are written using C++ language

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Computer Graphics

The term computer graphics has been used in a broad sense to describe almost everything on computers that is not text or sound. Typically, the term computer graphics refers to several different things:

- The representation and manipulation of image data by a computer.
- The various technologies used to create and manipulate images.
- The sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content.

Today, computers and computer-generated images touch many aspects of daily life. Computer images is found on television, in newspapers, for example in weather reports, in all kinds of medical investigation and surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. In the media such graphs are used to illustrate papers, reports, thesis, and other presentation material.Many powerful tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: 2D, 3D, 4D, 7D, and animated graphics.

As technology has improved, 3D computer graphics have become more common. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with the visualization of three dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis

is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic component.

1.2 History of Computer Graphics

In 1959, the TX-2 computer was developed at MIT's Lincoln Laboratory. The TX-2 integrated a number of new man-machine interfaces. A light pen could be used to draw sketches on the computer using Ivan Sutherland's revolutionary Sketchpad software.Using a light pen, Sketchpad allowed one to draw simple shapes on the computer screen, save them and even recall them later. The light pen itself had a small photoelectric cell in its tip. This cell emitted an electronic pulse whenever it was placed in front of a computer screen and the screen's electron gun fired directly at it. By simply timing the electronic pulse with the current location of the electron gun, it was easy to pinpoint exactly where the pen was on the screen at any given moment. Once that was determined, the computer could then draw a cursor at that location.

In 1961 another student at MIT, Steve Russell, created the first video game, E. E. Zajac, a scientist at Bell Telephone Laboratory (BTL), created a film called "Simulation of a two-giro gravity attitude control system" in 1963. During 1970s, the first major advance in 3D computer graphics was created at University of Utah by these early pioneers, the hidden-surface algorithm. In order to draw a representation of a 3D object on the screen, the computer must determine which surfaces are "behind" the object from the viewer's perspective, and thus should be "hidden" when the computer creates (or renders) the image.

In the 1980s, artists and graphic designers began to see the personal computer, particularly the Commodore Amiga and Macintosh, as a serious design tool, one that could save time and draw more accurately than other methods. In the late 1980s, SGI computers were used to create some of the first fully computer-generated short films at Pixar. The Macintosh remains a highly popular tool for computer graphics among graphic design studios and businesses. Modern computers, dating from the 1980s often use graphical user interfaces (GUI) to present data and information with symbols, icons and pictures, rather than text. Graphics are one of the five key elements of multimedia technology.

3D graphics became more popular in the 1990s in gaming, multimedia and animation. In 1996, Quake, one of the first fully 3D games, was released. In 1995, Toy Story, the first full-length computer-generated animation film, was released in cinemas worldwide. Since then, computer graphics have only become more detailed and realistic, due to more powerful graphics hardware and 3D modeling software.

1.3 Applications of Computer Graphics

The applications of computer graphics can be divided into four major areas:

- Display of information
- Design
- Simulation and animation
- User interfaces

1.3.1 Display of information

Computer graphics has enabled architects, researchers and designers to pictorially interpret the vast quantity of data. Cartographers have developed maps to display the celestial and geographical information. Medical imaging technologies like Computerized Tomography (CT), Magnetic Resonance Imaging (MRI), Ultrasound, Positron Emission Tomography (PET) and many others make use of computer graphics.

1.3.2 Design

Professions such as engineering and architecture are concerned with design. They start with a set of specification; seek cost-effective solutions that satisfy the specification. Designing is an iterative process. Designer generates a possible design, tests it and then uses the results as the basis for exploring other solutions. The use of interactive graphical tools in Computer Aided Design (CAD) pervades the fields including architecture, mechanical engineering, and the design of very-large-scale integrated (VLSI) circuits and creation of characters for animation.

1.3.3 Simulation and Animation

Once the graphics system evolved to be capable of generating sophisticated images in real time, engineers and researchers began to use them as simulators. Graphical flight simulators have proved to increase the safety and to reduce the training expenses. The field of virtual reality (VR) has opened many new horizons. A human viewer can be equipped with a display headset that allow him/her to see the images with left eye and right eye which gives the effect of stereoscopic vision. This has further led to motion pictures and interactive video games.

1.3.4 User interfaces

Computer graphics has led to the creation of graphical user interfaces (GUI) using which even naive users are able to interact with a computer. Interaction with the computer has been dominated by a visual paradigm that includes windows, icons, menus and a pointing device such as mouse. Millions of people are internet users; they access the internet through the graphical network browsers such as Microsoft internet explorer and Mozilla Firefox.

Chapter 2

OpenGL

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. OpenGL provides a set of commands to render a three dimensional scene. That means you provide the data in an OpenGL-useable form and OpenGL will show this data on the screen (render it). It is developed by many companies and it is free to use. You can develop OpenGL-applications without licensing. OpenGL is a hardware- and system-independent interface. An OpenGL-application will work on every platform, as long as there is an installed implementation. Because it is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

2.1 OpenGL Libraries

Computer Graphics are created using OpenGL, which became a widely accepted standard software system for developing graphics applications. As a software interface for graphics hardware, OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images). OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer. OpenGL stands for 'open graphics library' graphics library is a collection of API's (Applications Programming Interface). Graphics library functions are:

- GL library (OpenGL in windows) – Main functions for windows.

- GLU (OpenGL utility library) - Creating and viewing objects.
- GLUT (OpenGL utility toolkit)- Functions that help in creating interface of windows

OpenGL draws primitives—points, line segments, or polygons—subject to several selectable modes. You can control modes independently of each other; that is, setting one mode doesn't affect whether other modes are set (although many modes may interact to determine what eventually ends up in the frame buffer). Primitives are specified, modes are set, and other OpenGL operations are described by issuing commands in the form of function calls. These libraries are included in the application program using preprocessor directives OpenGL User Interface Library (GLUI) is a C++ user interface library based on the OpenGL Utility Toolkit (GLUT) which provides controls such as buttons, checkboxes, radio buttons, and spinners to OpenGL applications. It is window and operating system independent, relying on GLUT to handle all system-dependent issues, such as window and mouse management. The OpenGL Utility Library (GLU) is a computer graphics library. It consists of a number of functions that use the base OpenGL library to provide higher-level drawing routines from the more primitive routines that OpenGL provides. It is usually distributed with the base OpenGL package.

2.2 OpenGL Contributions

It is very popular in the video games development industry where it competes with Direct3D (on Microsoft Windows).OpenGL is also used in CAD, virtual reality, and scientific visualization programs.OpenGL is very portable. It will run for nearly every platform in existence, and it will run well. It even runs on Windows NT 4.0 etc. The reason OpenGL runs for so many platforms is because of its Open Standard. OpenGL has a wide range of features, both in its core and through extensions. Its extension feature allows it to stay immediately current with new hardware features, despite the mess it can cause.

2.3 Limitations

- OpenGL is case sensitive.
- Line Color, Filled Faces and Fill Color not supported.
- Shadow plane is not supported.

Chapter 3

Resource Requirements

3.1 Hardware Requirements

The Hardware requirements are very minimal and the program can be run on most of the machines.Table 3.1 gives details of hardware requirements.

Table 3.1: Hardware Requirements

Processor	Intel Core i3 processor
Processor Speed	1.70 GHz
RAM	4 GB
Storage Space	40 GB
Monitor Resolution	1024*768 or 1336*768 or 1280*1024

3.2 Software Requirements

The software requirements are description of features and functionalities of the system.Table 3.2 gives details of software requirements.

Table 3.2: Software Requirements

Operating System	Windows 8.1
IDE	Microsoft Visual Studio with C++ 2022
OpenGL libraries	glut.h,glu32.lib,opengl32.lib,glut32.lib glu32.dll,glut32.dll,opengl32.dll.

Chapter 4

System Design

The description of all the functions used in the program is given below:

- **void glutInitDisplayMode(unsigned int mode):**
  This function requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE, GLUT_DOUBLE).

- **void glutInitWindowPosition(int x, int y):** This specifies the initial position of top-left corner of the windows in pixels.

- **void glutInitWindowSize(int width, int height):** This function specifies the initial height and width of the window in pixels.

- **void glutCreateWindow(char *title):** This function creates a window on the display the string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

- **void glutDisplayFunc(void (*func)(void)):** This function registers the display func that is executed when the window needs to be redrawn.

- **void glClearColor(GLclampfr,GLclampfg GLclampfb,GLclampf a):** This sets the present RGBA clear colour used when clearing the colour buffer. Variables of type GLclampf are floating point numbers between 0.0 and 1.0.

Chapter 5

Implementation

```
//save as main.cpp
#include <GL/glut.h>
#include <iostream>
#include <fstream>
#include "game.h"

#define ROWS 40.0               // no of row squares on the screen
#define COLUMNS 40.0            // no of column squares on the screen
#define MAX 50
#define UP 1
#define RIGHT 2
#define DOWN -1
#define LEFT -2

void initGrid(int, int);
void draw_grid();//used to draw grid
void draw_food();// a square shape indicated as food
void draw_snake();// as the snake passes the red square its length increases by one pi
std::ofstream ofile;
std::ifstream ifile;
bool game_over = false;//used to display message on the console
extern int sDirection;
int score = 0;
```

```
void init();
void display_callback();
void input_callback(int, int, int);
void reshape_callback(int, int);
void timer_callback(int);//time the moment of snake on screen

void init()
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    initGrid(COLUMNS, ROWS);
}

//Callbacks
void display_callback()     //to display game over status
{
    if (game_over)              //to display final score
    {
        ofile.open("score.dat", std::ios::trunc);
        //to open the src file and calculate the score
        ofile << score << std::endl;
        ofile.close();
        //closes the file
        ifile.open("score.dat", std::ios::in);
        char a[4];
        ifile >> a;
        std::cout << a;
        char text[50] = "Your score : ";
        strcat_s(text, a);
        //concatenates the the score value with string text
        std::cout << text;
        exit(0);
    }
    glClear(GL_COLOR_BUFFER_BIT);
```

```
    //used to clear a buffer
    glLoadIdentity();
    draw_grid();//the grid is drawn using the
    draw_food();
    draw_snake();
    glutSwapBuffers(); //used to swap the buffers
}
void reshape_callback(int w, int h)
{
    glViewport(0, 0, (GLfloat)w, GLfloat(h));
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();


    //leftmost point of origin 0.0
    //-ve coming towards us +ve going towards screen
    glOrtho(0.0, COLUMNS, 0.0, ROWS, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void timer_callback(int) //to set fps
{
    glutPostRedisplay();
    glutTimerFunc(100, timer_callback, 0);
}
void input_callback(int key, int x, int y)
{
    switch (key)
    {
    case GLUT_KEY_UP: //keyboard key up to move the snake up
        if (sDirection != DOWN)
            sDirection = UP;
        break;
    case GLUT_KEY_DOWN: //keyboard key up to move the snake down
```

```
        if (sDirection != UP)
            sDirection = DOWN;
        break;
    case GLUT_KEY_RIGHT://keyboard key up to move the snake right
        if (sDirection != LEFT)
            sDirection = RIGHT;
        break;
    case GLUT_KEY_LEFT://keyboard key up to move the snake left
        if (sDirection != RIGHT)
            sDirection = LEFT;
        break;
    }
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowPosition(10, 10);
    glutInitWindowSize(600, 600);
    glutCreateWindow("SNAKE v1.0");
    glutDisplayFunc(display_callback);
    glutReshapeFunc(reshape_callback);
    glutSpecialFunc(input_callback);
    glutTimerFunc(100, timer_callback, 0); //100 frames have been set
    init();
    glutMainLoop();
    return 0;
}

//save as game.cpp
#include <GL/glut.h>
#include <iostream>
#include <ctime>
#include "game.h"
```

```
#define MAX 50
#define UP 1
#define RIGHT 2
#define DOWN -1
#define LEFT -2

void initGrid(int, int);
void draw_grid();
void draw_food();
void draw_snake();
void unit(int, int);            //each pixel unit is taken as a single square
int random(int, int);

bool length_inc = false;
bool seedflag = false;
extern int score;
extern bool game_over;
bool food = false;
int rows = 0, columns = 0;
int sDirection = RIGHT;         //initially the snake will be seeing moving right
int foodx, foody;               // coordinate position of the food
int posx[MAX + 1] = { 4,3,2,1,0,-1,-1 };
int posy[MAX + 1] = { 10,10,10,10,10,10,10 };
int length = 7;


void initGrid(int x, int y)// specify the coordinate values so as to create the grid
{
    columns = x;
    rows = y;
}
//drawing of grid
void draw_grid()
{
```

```
    for (int i = 0; i < columns; i++)
    {
        for (int j = 0; j < rows; j++)
        {
            unit(i, j);
        }
    }
}

void draw_snake()
{
    for (int i = length - 1; i > 0; i--)
    {
        posx[i] = posx[i - 1];
        posy[i] = posy[i - 1];
    }
    for (int i = 0; i < length; i++)
    {
        glColor3f(0.0, 1.0, 0.0);
        if (i == 0)
        {
            glColor3f(0.0, 0.0, 1.0);
            switch (sDirection)
            {
            case UP:
            //keyboard key UP moves the snake one unit up
                posy[i]++;
                break;
            case DOWN:
            //keyboard key DOWN moves the snake one unit down
                posy[i]--;
                break;
            case RIGHT:
            //keyboard key right moves the snake one unit right
```

```
            posx[i]++;
            break;
        case LEFT:
        //keyboard key left moves the snake one unit left
            posx[i]--;
            break;
        }
    if (posx[i]==0||posx[i]==columns-1||posy[i]==0||posy[i] == rows - 1)

        //if it touches the red squares on the corener edges  game over



            game_over = true;
        else if (posx[i] == foodx && posy[i] == foody)
        {
            food = false;
            score++;
            if (length <= MAX)
                length_inc = true;
            if (length == MAX)
                std::cout << "The snake has reached max length";
        }
        for (int j = 1; j < length; j++)
        {
            if (posx[j] == posx[0] && posy[j] == posy[0])
                game_over = true;
        }
    }
    glBegin(GL_QUADS);
    glVertex2d(posx[i], posy[i]);
    glVertex2d(posx[i] + 1, posy[i]);
    glVertex2d(posx[i] + 1, posy[i] + 1); glVertex2d(posx[i], posy[i] + 1);
    glEnd();
}
```

```
    if (length_inc)
    {
        length++;
        length_inc = false; //to reset the length value
    }
}

void draw_food()
{
    if (!food)
    {
        foodx = random(2, columns - 2); //Since it should not cross the red line
        foody = random(2, rows - 2);
        std::cout << foodx << foody << std::endl;
        food = true;
    }
    glBegin(GL_QUADS);
    glVertex2d(foodx, foody);
    glVertex2d(foodx + 1, foody);
    glVertex2d(foodx + 1, foody + 1);
    glVertex2d(foodx, foody + 1);
    glEnd();
}

void unit(int x, int y)
{
    glLoadIdentity();
    //x==0 y==0 to check if the box is left and bottom of the screen
    //x==columns-1 and y==rows-1 if the box is towards the end
    if (x == 0 || x == columns - 1 || y == 0 || y == rows - 1)
    {
        //edges of the box are red hence snake is not allowed to move here
        glLineWidth(4.0);
        glColor3f(1.0, 0.0, 0.0);
```

```
    }
    else
    {
        //edges are coloured white
        glColor3f(0.0, 0.0, 0.0);
        glLineWidth(1.0);
    }
    glBegin(GL_LINES);
    //bottom left corner
    glVertex2d(x, y);
    glVertex2d(x + 1, y);
    glVertex2d(x + 1, y); glVertex2d(x + 1, y + 1);
    glVertex2d(x + 1, y + 1); glVertex2d(x, y + 1);
    glVertex2d(x, y + 1); glVertex2d(x, y);
    glEnd();
}
int random(int _min, int _max) //for random position of food
{
    if (!seedflag)
    {
        srand(time(NULL));
        seedflag = true;
    }
    else
        seedflag = false;
    return _min + rand() % (_max - _min);
}
```

Chapter 6

Testing

In unit testing, the program modules that make up the system are tested individually. Unit testing focuses to locate errors in the working modules that are independent to each other. This enables to detect errors in coding and the logic within the module alone. This testing is also used to ensure the integrity of the data stored. The various routines were checked by passing the inputs and the corresponding output is tested.Table 6.1 gives details of validation. Test cases used in the project as follows:

Table 6.1: Test Case Validation

Test Case No.	**Metric**	**Description**	**Observation**
1.	Keyboard Function	Key y/Y to continue ,key n/N to exit, key w/W for send window, key r/R for receive window, key s/S for safe sending, key p/P for packet crashing, key t/T for timeout.	Results obtained as expected.
2	Display Function	Computers, send window, receive window, packets, timer are displayed.	Results obtained as expected
3	Animation Effect	Movement of packets and timer movement.	Results obtained as expected

Chapter 7

Results & Snapshots

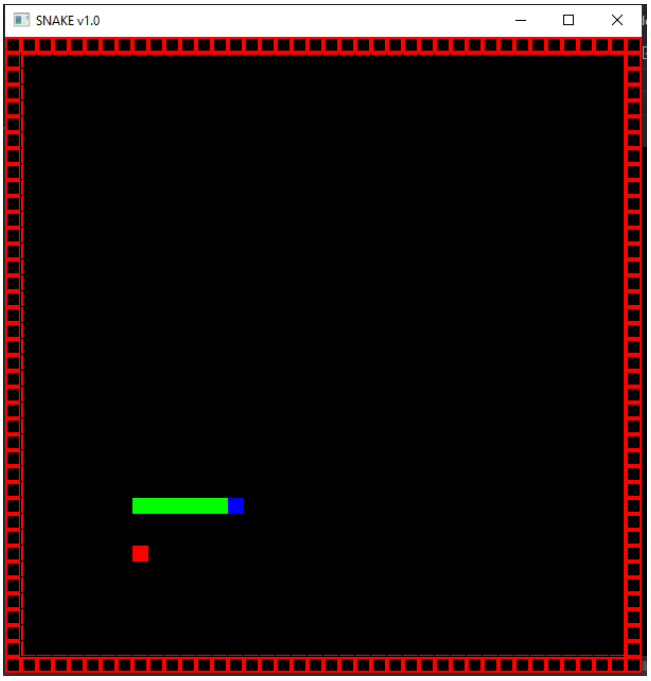

Figure 7.1: Initial Screen with Snake

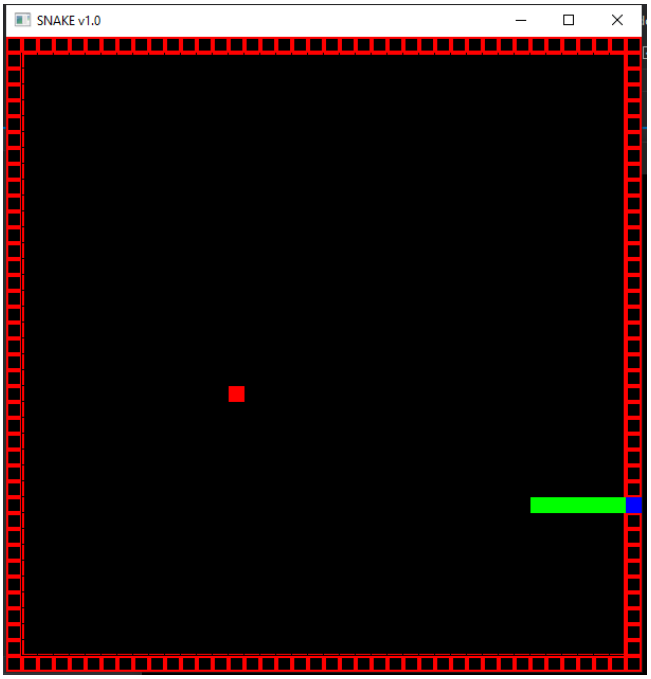

Figure 7.2: Game Over Status

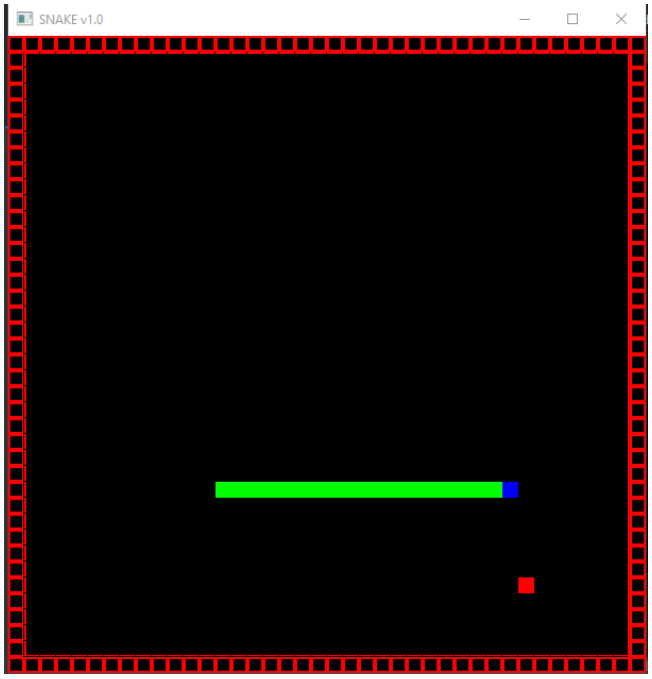

Figure 7.3: Increase in size of snake

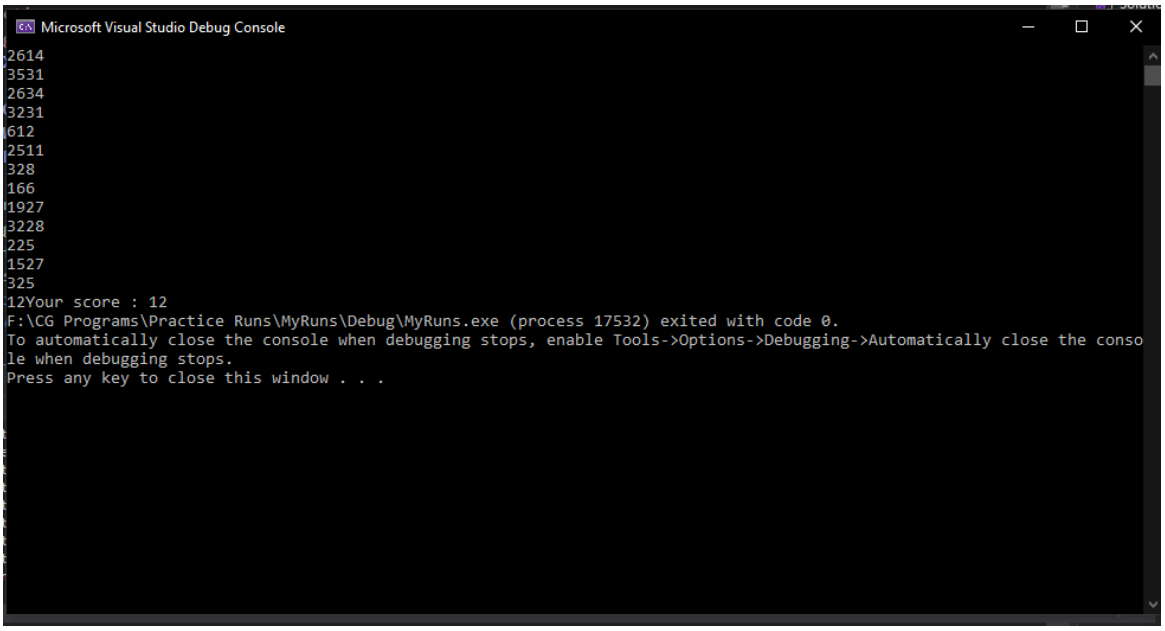

Figure 7.4: Final Score

Chapter 8

Conclusion & Future Enhancements

8.1 Conclusion

Insight – An application developed to implement one of the most played snake games which is available in most of the Nokia phones and various other keypad based cellphones. The various OpenGL functions available are really helpful in creating graphic animations and creating various shapes, adding colors.Two .cpp files were used in this main.cpp and game.cpp. The main.cpp handles keyboard functions, displays the game status. The game.cpp hadles the snake. Its size and movement. The score.dat file displays the final score of the player.

8.2 Future Enhancements

Some of the future works include as follows firstly we can make a initial menu which has options such as Play Game Leaderboard Settings Exit. Secondly we can make use of 3d animations to make the snake look more realistic, the game over status can be displayed as a popup widow rather than showing it on window command prompt. We can also create functions which will ask the user if he wants to play the game again

References

[1] Edward Angel, *"Interactive Computer Graphics A Top-Down Approach With OpenGL"* 5th Edition, Addison-Wesley, 2008.

[2] F.S. Hill,*"Computer Graphics Using OpenGL"*, 2nd Edition, Pearson Education, 2001.

[3] James D.Foley, Andries Van Dam, Steven K. Feiner, John F Hughes, *"Computer Graphics"*, Second Edition, Addison-Wesley Professional, August 14,1995.

[4] @online OpenGL Official ,`https://www.opengl.org/`

[5] @online OpenGL Overview , `https://https://www.khronos.org/opengl/`