VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590014

A Computer Graphics Mini Project Report On

"SNAKE GAME"

Submitted in Partial fulfillment of the Requirements for the VI Semester of the Degree of

Bachelor of Engineering
In
Computer Science & Engineering

By

NAVEED ALI (4MW16CS007)
KEERTHAN ACHARYA (4MW16CS028)

Under the Guidance of

B.N RAMACHANDRA
Asst. Prof, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING Shri Madhwa Vadiraja Institute of Technology & Management Vishwothama Nagar, Bantakal-574115

MAY, 2019

SHRI MADHWA VADIRAJA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(A Unit of Shri Sode Vadiraja Mutt Education Trust ®, Udupi)

Vishwothama Nagar, BANTAKAL – 574 115, Udupi District, Karnataka, INDIA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Computer Graphics Project work entitled **"SNAKE GAME"** has been carried out by **NAVEED ALI (4MW16CS007) and KEERTHAN (4MW16CS028)** bonafide students of Shri Madhwa Vadiraja Institute of Technology and Management in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2018-2019**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The Graphics Project Report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Mr. B.N.RAMACHANDRA.
Project Guide
Dept. of CSE

Dr. VASUDEVA
Professor and Head
Dept. of CSE

External Viva

Name of the examiners — Signature with date

1.

2.

Acknowledgements

It is our pleasure to express our heartfelt thanks to Prof. B.N.Ramachandra, Assistant Professor, Department of Computer Science and Engineering, for his supervision and guidance which enabled us to understand and develop this project.

We are indebted to Prof. Dr. Thirumaleshwara Bhat, Principal and Prof. Dr. Vasudeva, Head of the Department, for their advice and suggestions at various stages of the work.

We thank the Management of Shri Madhwa Vadiraja Institute of Technology and Management, Bantakal, Udupi for providing us with a good study environment and laboratories facilities. Besides, we appreciate the support and help rendered by the teaching and non-teaching staff of Computer Science and Engineering.

Lastly, we take this opportunity to offer our regards to all of those who have supported us directly or indirectly in the successful completion of this project work.

Naveed Ali (4MW16CS007)

Keerthan Acharya(4MW16CS028)

ABSTRACT

The term ‘computer graphics’ refers to computer-generated image data created with help from specialized graphical hardware and software. It is a vast and recent area in computer science.

The importance of computer graphics lies in its applications. In engineering applications (e.g. automotive and aerospace), the ability to quickly visualize newly designed shapes is indispensible. Before the advent of computer graphics, designers built expensive prototypes and time-consuming clay models. Now, designers can interactively view and modify models of their shapes using a computer. Medical imaging is another application where computer graphics has proven valuable. Interactive computer graphics allows the physician to interpret this large volume of data in new and useful ways.

Snake is two stage game in which food pops up randomly on the screen which is eaten by the snake. Task is to direct the snake using up, down, left and right direction keys. The snake moves and eats the food as soon as the head of the snake touches the food. With each food the snake eats, it grows bigger by one matrix. Now the snake is unable to pass through walls as a protective wall emerges at the boundary. OpenGL is the standard library used for implementing the game in C++ programming language.

Table of Contents

Chapter 1

INTRODUCTION

1.1 About Computer Graphics

Computer graphics is a sub-field of computer science and is concerned with digitally synthesizing and manipulating visual content. Although the term refers to three-dimensional computer graphics, it also encompasses two-dimensional graphics and image processing. Computer graphics is often differentiated from the field of visualization, although the two have similarities. Graphics are visual presentation on some surface like wall, canvas, computer screen. Graphics often combine text, illustration and color.

In this era of computing, computer graphics has become one of the most powerful and interesting fact of computing. It all started with display of data on hardcopy and CRT screen. Now computer graphics is about creation, retrieval, manipulation of models and images. Graphics today is used in many different areas such as industries, animation, gaming, simulation, designing etc.

Graphical interfaces have replaced textual interfaces as the standard means for user-computer interaction. Graphics has also become a key technology for communicating ideas, data and trends in most areas of commerce, science, engineering and education. Much of the task of creating effective graphic communication lies in modeling the objects whose images we want to produce.

Computer graphics is no more a rarity. Even people who do not use computers in their daily work encounter computer graphics in television commercials and as cinematic special effects. Computer graphics is an integral part of all user interfaces and is indispensable for visualizing objects.

1.2 About OpenGL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. It offers various functions to implement primitives, models and images. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

1.2.1 OpenGL User Interface Library:

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.

OpenGL functions can be accessed through the following three Libraries:

- The first library is the main **GL** library. The functions in this library start with the letters ‘gl’.
- The second is the **OpenGL Utility Library (GLU).** This library uses only GL functions but contains the code for creating common objects and simplifying viewing. All the functions in the GLU can be created from the core GL library. Functions in this library begin with the letters ‘glu’.
- The third is the **OpenGL Utility Toolkit (GLUT),** which provide the minimum functionality that should be expected in any modern windowing system. This is used to interface with the windows system & to get input from the external devices into our programs. Functions in this library start with the letters ‘glut’.

1.3 Simulation of SNAKE GAME:

Snake is two stage game in which food pops up randomly on the screen which is eaten by the snake. Task is to direct the snake using up, down, left and right direction keys. The snake moves and eats the food as soon as the head of the snake touches the food. With each food the snake eats, it grows bigger by one matrix. Now the snake is unable to pass through walls as a protective wall emerges at the boundary. OpenGL is the standard library used for implementing the game in C++ programming language.

Chapter 2

REQUIREMENT ANALYSIS

2.1 Software Requirements

- Operating System: Windows
- Compiler used: C/C++
- Microsoft Visual Studio 2012
- Header files:
    1. GL/glut.h
    2. Stdio.h

2.2 Hardware Requirements

- RAM Size: 4.0 GB
- Keyboard Standard qwerty serial
- Mouse: USB mouse
- :

Chapter 3

DESIGN

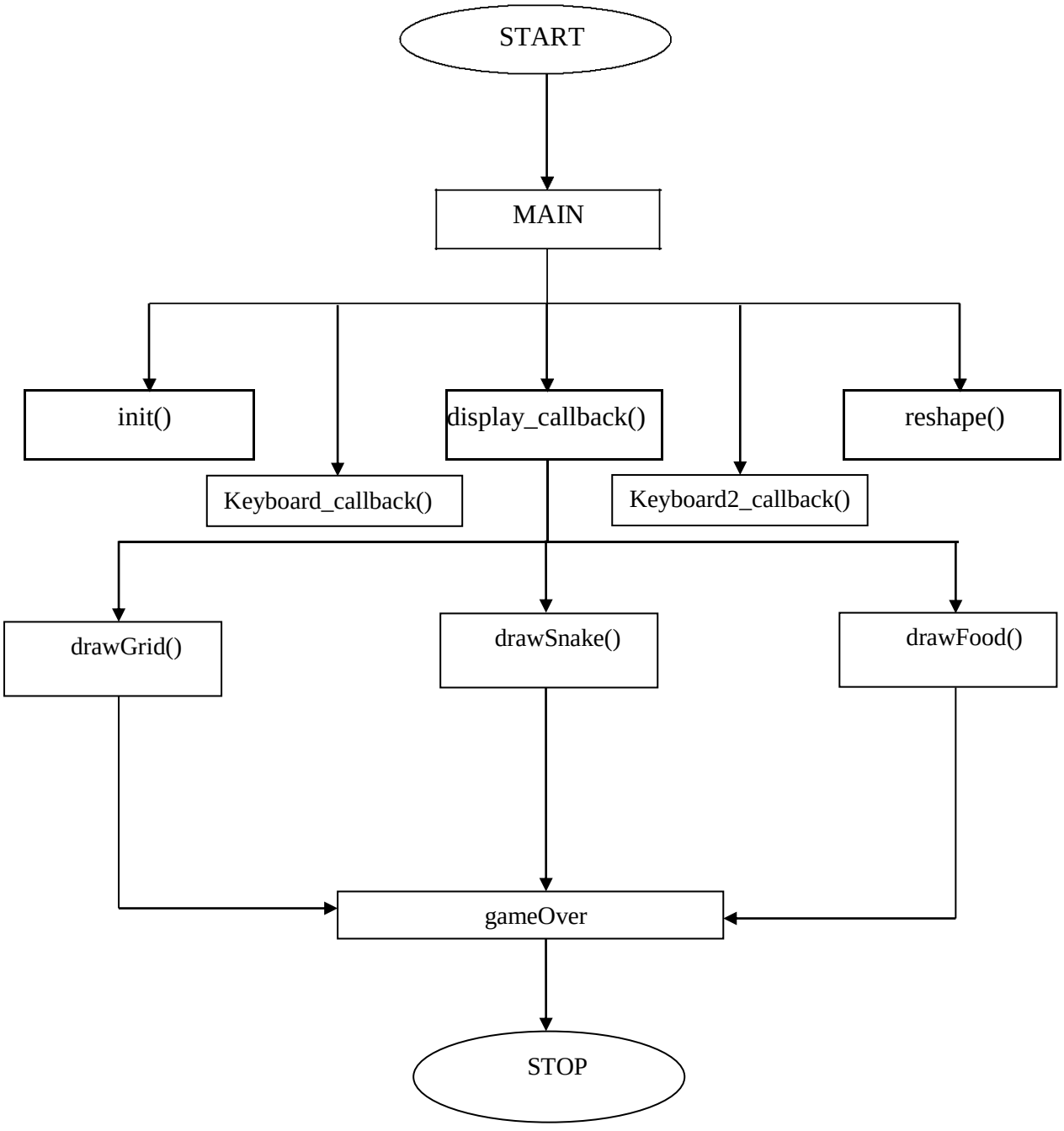
3.1 System Analysis: The project implements grids, vertices, linked list for designing and playing the game created. We have created a separate file for each source and game. The source is to extract and import the contents and files present in the game.cpp text.

3.2 System design:

User Defined functions:

- `initGrid()`
- `drawGrid()`
- `unit()`
- `drawFood()`
- `drawSnake()`
- `random()`
- `display_callback()`
- `init()`
- `output()`
- `disp_start()`
- `display_callback()`
- `reshape_callback()`
- `Timer_callback()`
- `keyboard_callback()`

3.3 Control Flow Diagram:



Chapter 4

IMPLEMENTATION

4.1 Header Files

Stdio.h: The C programming language provides many standard library functions for the input and output. It consists of operations such as file access, unformatted input/output, formatted input/output, file positioning and error handling.

GL/glut.h: The OpenGL Utility Toolkit (GLUT) handles most of the system dependent actions required to display a window, put OpenGL graphics in it, and accept mouse and keyboard input. glut.h includes gl.h and glu.h.

4.2 Built in functions used in OpenGL:

glClear ():
The glClear function clears buffers to preset values.

Syntax: glClear (GLbitfieldmask);

Where 'mask' is bitwise OR operators of masks that indicate the buffers to be cleared. The masks used are as follows:

GL_COLOR_BUFFER_BIT: The buffers currently enabled for writing.

GL_DEPTH_BUFFER_BIT: The depth buffer.

glMatrixMode ():
The glMatrixMode function specifies which matrix is the current matrix.

Syntax: void glMatrixMode (GLenum mode);

Where the 'mode' indicates the matrix stack that is the target for subsequent matrix operations. The mode parameter can assume values like GL_PROJECTION which applies subsequent matrix operations to the projection matrix stack or GL_MODELVIEW etc.

glClearColor():

Sets the present RGB clear color used hen clearing the color buffer.The variables of type GLclampf are floating point numbers between 0.0 and 1.0.

Syntax: void glClearColor (GLclampf r, GLclampf g, GLclampf b, GLclampf a);

glFlush():

Forces any buffered OpenGL command to execute.It ensures that points rendered to the screen as soon as possible.

Syntax:void glFlush ();

glutInit():

Initialises GLUT and processes any command-line arguments.The command line options are dependent on the window system.glutInit () should be called before any other GLUT routine.

Syntax:void glutInit(int argc, char **argv);

glutInitDisplayMode():

glutInitDisplayMode function sets the initial display mode.

Syntax: void glutInitDisplayMode (unsigned int mode);

Where ‘mode’ is normally the bitwise OR-ing of display mode bit masks such as GLUT_RGB ,GLUT_SINGLE or GLUT_DOUBLE, and any of the buffer-enabling flags: GLUT_DEPTH.The default value is GLUT_RGB|GLUT_SINGLE.

glutInitWindowPosition() and glutInitWindowSize():

glutInitWindowPosition and glutInitWindowSize functions set the initial window position and size respectively.

Syntax: void glutInitWindowPosition (int x, int y);

void glutInitWindowSize (int width, int height);

glutCreateWindow():

glutCreateWindow creates a top-level window with the name contained in the string ‘title’.

Syntax: int glutCreateWindow (char *title);

glutMainLoop():

Enters the GLUT processing loop, which is an infinite loop.

Syntax:void glutMainLoop();

glutDisplayFunc():

glutDisplayFunc function sets the display call back for the current window. Syntax: void glutDisplayFunc (void (*func)(void));

glutCreateMenu():

This function creates a new pop up menu and returns a unique small integer identifier. The range of allocated identifiers starts at one.
Syntax: int glutCreateMenu (void (*func)(int value));

glTranslatef():

This function is used to displace an object by a fixed distance in a given direction. x, y, z are the components of the displacement vector.
Syntax: void glTranslatef (GLfloat x, GLfloat y, GLfloat z);

glPushMatrix() and glPopMatrix():

These functions are used to push the transformation matrix on to the stack and later recover the original condition respectively.glPushMatrix () pushes the current matrix stack down by one, duplicating the current matrix.That is, after a glPushMatrix() call, the matrix on top of the stack is identical to the one below it.glPopMatrix() pops the current matrix stack,replacing the current matrix with the one below it on the stack. Syntax: void glPushMatrix (void);

void glPopMatrix(void);

glutPostRedisplay():

This function requests that the display call back be executed after the current call back returns i.e, it marks the current window as needing to be redisplayed. Syntax: void glutPostRedisplay(void);

glLoadIdentity():

This function replaces the current matrix with the identity matrix.

Syntax: void glLoadIdentity (void);

glRasterPos2f():

This function is used to position pixel and bitmap write operations. X and y specify current x and y raster position respectively.

Syntax: void glRasterPos2f (GLfloat x, GLfloat y);

glutBitmapCharacter():

Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font.

Syntax: void glutBitmapCharacter (void *font, int character);

glColor3f():

These function variants specify new red, green, and blue values explicitly and set the current alpha value to 1.0 implicitly.

glVertex2f():

These commands are used within glBegin()/glEnd() pairs to specify point,line, and polygon vertices.The current color,normal and texture coordinates are associated with the vertex when glVertex() is called.

Syntax: void glVertex2f(GLfloat x, GLfloat y);

glBegin() and glEnd():

glBegin () and glEnd() delimit the vertices that define a primitive or a group of like primitives.

Syntax: void glBegin(GLenum mode);

voidglEnd ();

glLineWidth():

This function specifies the width of rasterized lines. It specifies the rasterized width of both aliased and antialiased lines.

Syntax: void glLineWidth(GLfloat width);

glOrtho():

glOrtho()— multiply the current matrix with an orthographic matrix

Syntax: void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom,GLdouble top, GLdouble near, GLdouble far);

glViewport():

It specifies the affine transformation of x and y from normalized device coordinates to window coordinates.

Syntax:void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);

glutReshapeFunc(reshape):

It sets the reshape call back for the current window.

Syntax:void glutReshapeFunc(void(*func)(int width,int height));

glutSwapBuffers():

Performs a buffer swap on the layer in use for the current window.

Syntax: void glutSwapBuffers(void);

4.3 User Defined Functions

- void initGrid()

  This function is used to initialize the grid which serves as the skeleton of the game.

- void drawGrid()

  This function is used to display the grid that has been initialized.

- void unit()

  This function is used to give the units according to which the grid is to be created.

- void drawFood()

  This function is used to display the food that the snake is supposed to eat.

- void random()

  This function is implemented as a supplementary to drawFood() so as to make the food appear at random location among the grids.

- void drawSnake()

  This function is used to make the protagonist of the game i.e. the snake.

- void display_callback()

  This function is used to display the content of the whole game.

- void init()

  This function is used to initialize the imported initGrid() which is used to display the grid..

- void output()

  This function is implemented to display the attributes of the front page of the project.

- void reshape_callback()
  This function is implemented whenever size or shape of the application window changes.

- void timer_callback()

- void keyboard_callback()

  This function is implemented to direct the snake in the direction it is to travel during the course of the game.

Chapter 5

SCREENSHOTS

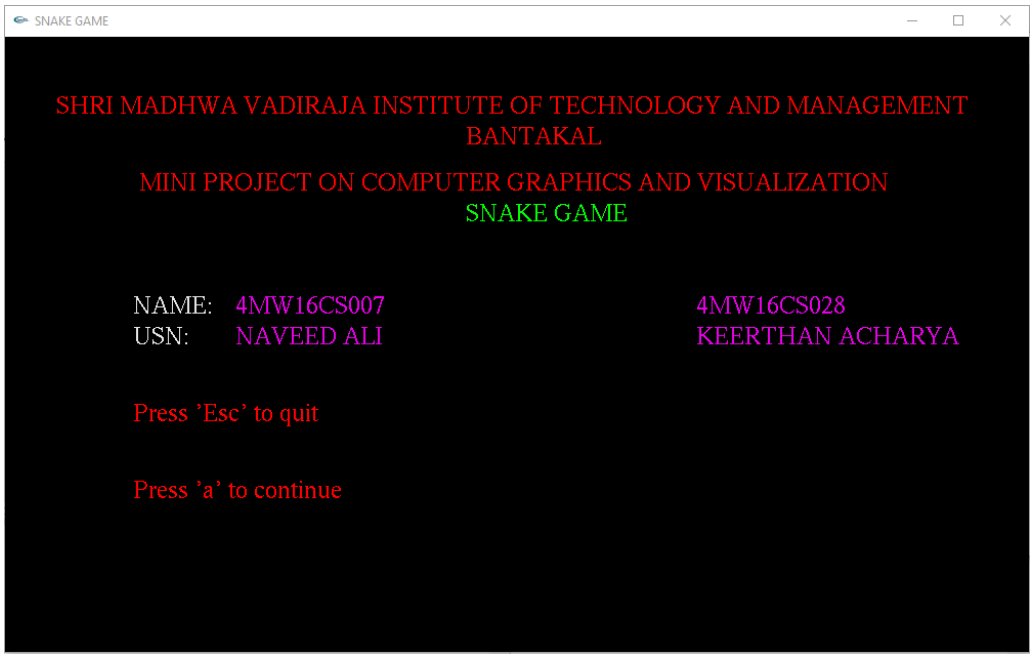


Fig 1: Front page

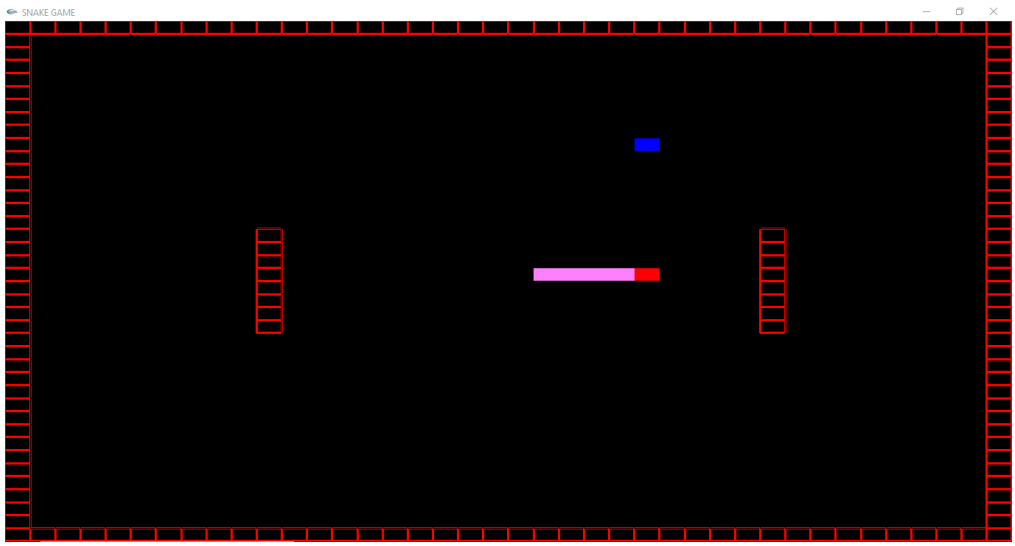


Fig 2: Basic dimensions of the game

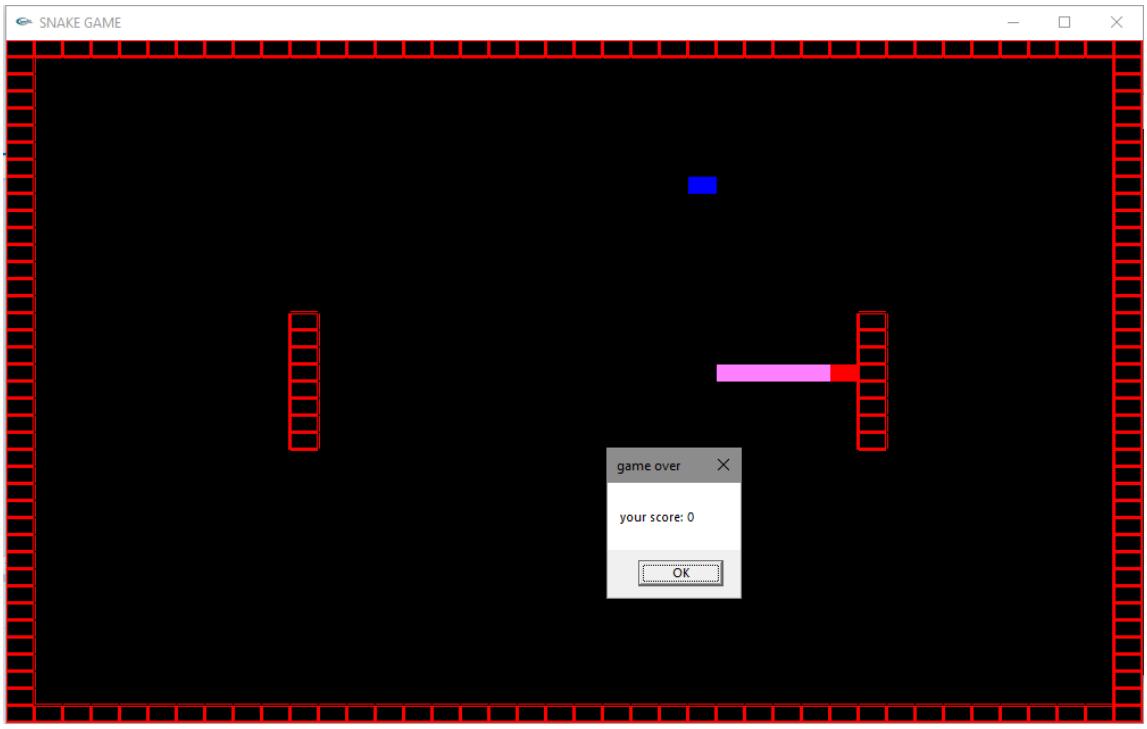


Fig 3: The score you obtain when the snake touches the wall.

Chapter 6

CONCLUSION

Computer generated models are often used as educational aids. This project helps in better understanding of working of SNAKE GAME under different scenarios.

By developing computer graphics based application helped us to understand use of inbuilt openGL functions. It also helped us to understand coordinate systems in computer graphics. As the project work was in progress, and OpenGL methodology was employed, we obtained a good experience in OpenGL software development.