## [VISVESVARAYA TECHNOLOGICAL UNIVERSITY](http://www.vtu.ac.in/)

##### “JNANA SANGAMA”, BELAGAVI-590018, KARNATAKA



**A Mini Project Report On**

***“SNAKE CLONE GAME”***

##### Submitted in the partial fulfillment of the requirement for the award of degree of

**BACHELOR OF ENGINEERING IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By**

|  |  |
| --- | --- |
| **Vaibhav Mehta** | **Siddharth Singh** |
| **(1VA15CS060)** | **(1VA15CS051)** |

**Under the Guidance of**

|  |  |  |
| --- | --- | --- |
| **Mr. Sukruth Gowda M A Assistant Professor**  **Dept. of CS&E** |  | **Mrs. Sreelatha PK**  **In-charge HOD & Assistant Professor**  **Dept. of CS&E** |

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## SAI VIDYA INSTITUTE OF TECHNOLOGY

**(Affiliated to Visvesvaraya Technological University, Belagavi | Recognized by Govt. of Karnataka | Approved by AICTE, New Delhi)**

##### RAJANUKUNTE, BENGALURU – 560064

**2017-18**

**SAI VIDYA INSTITUTE OF TECHNOLOGY**

**(Affiliated to Visvesvaraya Technological University, Belagavi | Recognized by Govt. of Karnataka | Approved by AICTE, New Delhi)**

**Rajanukunte, Bengaluru- 560064**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

Certified that the Mini Project work entitled *“****SNAKE CLONE GAME****”* carried out by **Mr. VAIBHAV MEHTA (1VA15CS060), Mr. SIDDHARTH SINGH (1VA15CS051)** a bona fide student of **SAI VIDYA INSTITUTE OF TECHNOLOGY**, Bengaluru, in partial fulfillment for the award of Bachelor of Engineering in Computer Science & Engineering of **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**, Belagavi during the year **2017-18.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the said Degree.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| --------- **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Mrs. Sreelatha PK**  In-charge HOD & Assistant Professor,  Dept of CS&E | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Mr. Sukruth Gowda M A**  Assistant Professor,  Dept. of CS&E | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Dr. HS Ramesh Babu**  Principal & Professor,  Dept. of CS&E |

**External Viva:**

|  |  |
| --- | --- |
| **Name** | **Signature** |
| **1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

# ABSTRACT

The aim of the mini project is to implement the Classic Snake Clone game. The Snake Clone game is for single player, where the player controls a dot, a square, or an object on a bordered plane or none bordered plane. As it moves forward, it leaves a trail behind, resembling a moving snake. After the release of this game in Nokia phones this concept gained interest among larger audience.

Snake Clone is not a very challenging game for human beings. If you’re an enthusiast, you’ve probably moved from the basic game to some variant where the game was assembled in a three-dimensional format with more possibilities of difficulty levels, but the basic concept still remains the same.

The concept in here is as the snake moves, it feeds upon the dot (food) appearing at random places on plane, thereby adding the dot to the length of snake resulting in the increase in the length of snake. Instead of running into the appeared dot if the snake runs into itself or the crosses out of window, the player loses the game. The variant is available with the menu driven themes, for changing the background colors along with automatically changing snake speed as the player goes on to score a certain number of points.

The completion of project brings with and sense of satisfaction, but it is never completed without thanking the persons who are all responsible for its successful completion. First and foremost we wish to express our deep sincere feelings of gratitude to our Institution, **Sai Vidya Institute of Technology**, for providing us an opportunity to do our education.

We would like to thank the **Management** and **Mr. M R Holla,** Director, Sai Vidya Institute of Technology for providing the facilities.

We extend our deep sense of sincere gratitude to **Dr. H S Ramesh Babu**, Principal & Professor, Sai Vidya Institute of Technology, Bengaluru, for having permitted to carry out the project work on Snake Clone Game successfully**.**

We are thankful to **Mr. A M Padma Reddy,** Professor and Dean (Student affairs), Department of Computer Science and Engineering, Sai Vidya Institute of Technology, for his constant support and motivation.

We express our heartfelt sincere gratitude to **Mrs. Sreelatha P K**, Assistant Professor and In-charge Head, Department of Computer Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for her valuable suggestions and support.

We convey our special in-depth, heartfelt, sincere gratitude to **Mr. Sukruth Gowda M A,** Asst. Prof. Dept., of CS&E, Sai Vidya Institute of Technology, Bengaluru, for his constant support in completing the project.

Finally, we would like to thank all the Teaching, Technical faculty and supporting staff members of Department of Computer Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for their support.

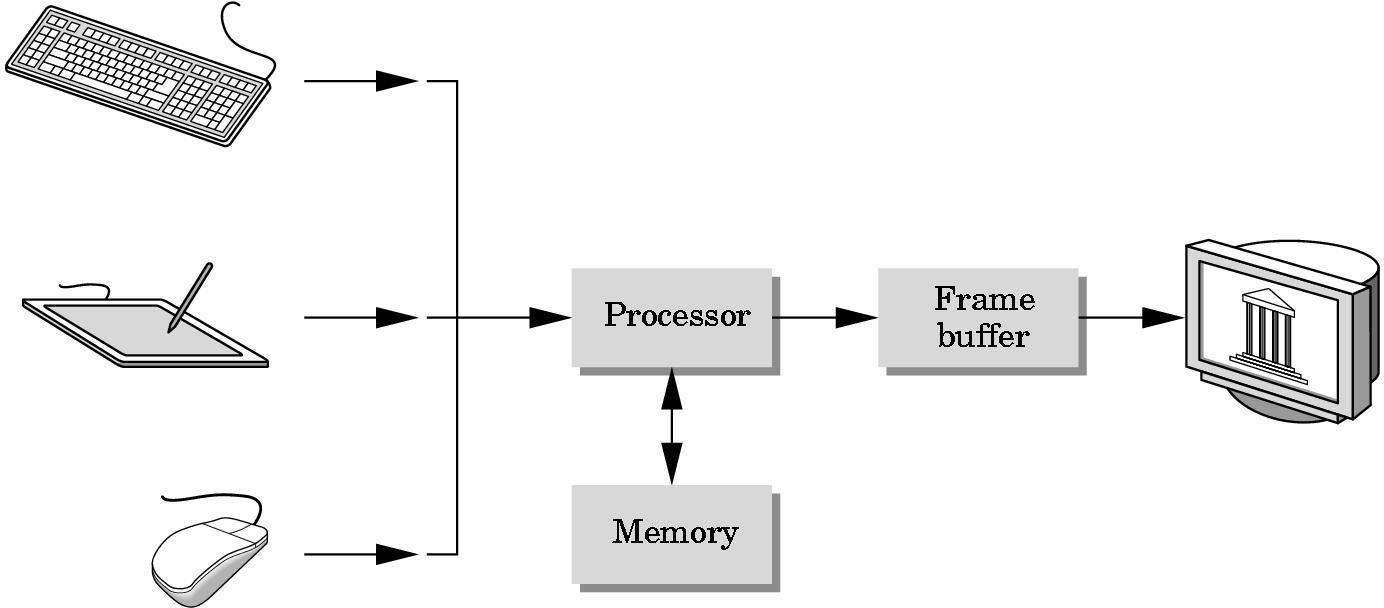
|  |  |
| --- | --- |
| Vaibhav Mehta | 1VA15CS060 |
| Siddharth Singh | 1VA15CS051 |

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Chapter Title** | **Page No.** |
| **CHAPTER 1** | **INTRODUCTION** | 1-4 |
| 1.1 | Computer Graphics | 1 |
| 1.2 | Uses of Computer Graphics | 2 |
| 1.3 | OpenGL Technology | 2 |
| **CHAPTER 2** | **REQUIREMENTS AND SPECIFICATION** | 5 |
| 2.1 | Hardware Requirements | 5 |
| 2.2 | Software Requirements | 5 |
| **CHAPTER 3** | **ANALYSIS COMPARISIONS & FUNCTIONS** | 6-9 |
| 3.1 | Basic Gameplay | 6 |
| 3.2 | Upgraded Version | 6-7 |
| 3.3 | Functions | 7-9 |
| **CHAPTER 4** | **DESIGN FLOWCHART** | 10 |
| **CHAPTER 5** | **IMPLEMENTATION** | 11-14 |
| **CHAPTER 6** | **CONCLUSION & FUTURE ENHANCEMENT** | 15 |
| **BIBLIOGRAPHY** | |  |
| **APPENDIX A** | **ABBREVIATIONS & ACRONYMS** |  |
| **APPENDIX B** | **SNAPSHOTS** |  |

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **CAPTION** | **PAGE NO.** |
| 1.1 | A Graphic System | 1 |
| 1.3.1  1.3.2 | Library Organization  OpenGL Graphics Architecture | 3  3 |
| 4.1 | Flowchart | 10 |
|  | Snapshot 1: Front window |  |
|  | Snapshot 2: Instruction window |  |
|  | Snapshot 3: Interactive Menus |  |
|  | Snapshot 4: ”Fun and Learning” theme |  |
|  | Snapshot 5: ”Google” theme |  |
|  | Snapshot 6: ”Cohesive colors-II” theme |  |

## CHAPTER 1

**INTRODUCTION**

* 1. **Computer Graphics**
     + Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.
     + Computers have become a powerful medium for the rapid and economical production of pictures.
     + There is virtually no area in which graphical displays cannot be used to some advantage.
     + Graphics provide a so natural means of communicating with the computer that they have become widespread.
     + Interactive graphics is the most important means of producing pictures since the invention of photography and television.
     + We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.
     + A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

##### Figure 1.1 A graphics system

## 1.2 Uses of computer graphics:

#### User interface:

* + - * It is now a well-established fact that graphical interfaces provide an alternative and easy interaction between users and computers the built in graphics provided with user interfaces use the control items.
      * In industry, business government and education organization’s computer graphics is most commonly used to create 2D and 3D graphs of mathematical, physical and economic functions in the form of histograms, bars and pie charts which are very useful in decision making.

#### Computer aided drafting and design:

* + - * The computer aided drafting uses the graphics to components and systems. Electrical, mechanical and electronic devices such as automobile bodies, structure of airplane, ships, buildings.

#### Simulation and animation for scientific visualization and Env.:

* + - * Use of graphics in simulation makes mathematical models and mechanical systems more realistic and easy to study. The interactive graphics supported by animation

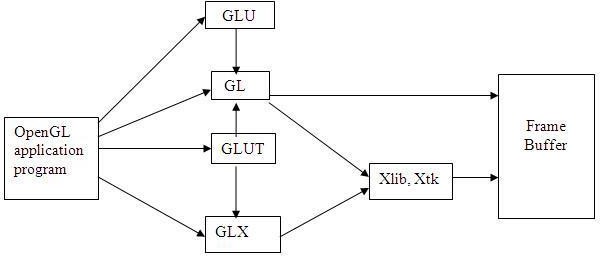
software proved their use in production of animated movies and cartoon films.

## OpenGL Technology

* + - OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. 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.
    - OpenGL Available Everywhere: Supported on all UNIX workstations, and shipped standard with every Windows 95/98/2000/NT and MacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.

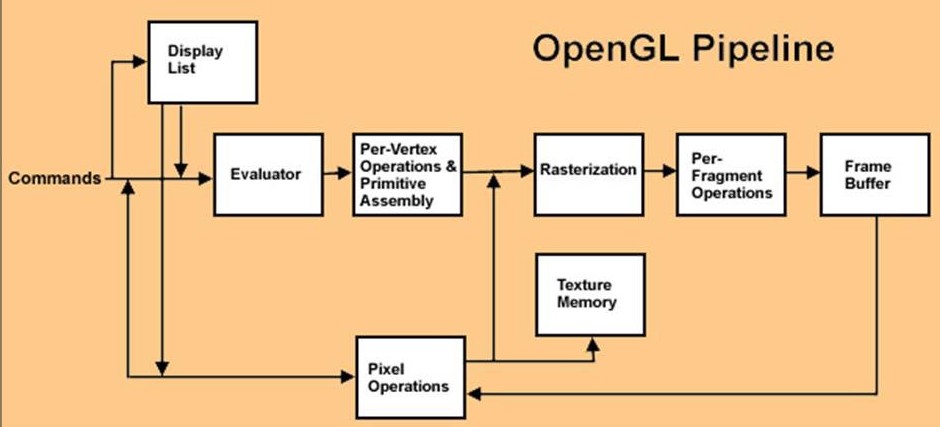
##### The OpenGL interface

Our application will be designed to access OpenGL directly through functions in three libraries namely: gl, glu, and glut.



##### Figure 1.3.1 Library Organization

* + 1. **OpenGL Graphics Architecture:**



**Figure 1.3.2 OpenGL Graphics Architecture**

* + - * **Display Lists:**

All data, whether it describes geometry or pixels, can be saved in a display list for current or later use. When a display list is executed, the retained data is sent from the display list just as if it were sent by the application in immediate mode.

#### Evaluators:

All geometric primitives are eventually described by vertices. Parametric curves and surfaces may be initially described by control points and polynomial functions called basis functions.

#### Per Vertex Operations:

For vertex data, next is the "per-vertex operations" stage, which converts the vertices into primitives. Some vertex data are transformed by 4 x 4 floating-point matrices. Spatial coordinates are projected from a position in the 3D world to a position on your screen.

#### Primitive Assembly:

Clipping, a major part of primitive assembly, is the elimination of portions of geometry which fall outside a half space, defined by a plane.

#### Pixel Operation:

While geometric data takes one path through the OpenGL rendering pipeline, pixel data takes a different route. Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components. Next the data is scaled, biased, and processed by a pixel map. The results are clamped and then either written into texture memory or sent to the Rasterization step.

#### Rasterization:

Rasterization is the conversion of both geometric and pixel data into fragments. Each fragment square corresponds to a pixel in the frame buffer. Color and depth values are assigned for each fragment square.

#### Fragment Operations:

Before values are actually stored into the frame buffer, a series of operations are performed that may alter or even throw out fragments. All these operations can be enabled or disabled.

# CHAPTER 2

# REQUIREMENTS AND SPECIFICATION

## Hardware Requirements

* + - The standard output device is assumed to be a **Colour Monitor**. It is quite essential for any graphics package to have this, as provision of colour options to the user is a must.
    - The **mouse**, the main input device, has to be functional i.e. used to give input in the game.
    - A **keyboard** is used for controlling and inputting data in the form of characters, numbers i.e. to change the user views.
    - Apart from these hardware requirements there should be sufficient hard disk space and primary memory available for proper working of the package to execute the program.
    - Pentium III or higher processor, 16MB or more RAM. A functional display card.
    - **Minimum Requirements** expected are cursor movement, creating objects like lines, squares, rectangles, polygons, etc. Transformations on objects/selected area should be possible. Filling of area with the specified colour should be possible.

## Software Requirements

* + - The editor has been implemented on the OpenGL platform and mainly requires an appropriate version of Microsoft Visual Studio to be installed and functional in Windows.
    - Though it is implemented in OpenGL, it is very much performed and independent with the restriction, that there is support for the execution of C and C++ files. Text Modes is recommended.
    - **Developed Platform:** Windows 10 version 1703
    - **Language Used In Coding:** C-language
    - **Tool Used In Coding:** Microsoft Visual Studio 2015/ Dev-C++

# CHAPTER 3

# ANALYSIS COMPARISON & FUNCTIONS USED

This chapter explains the changes implemented in the basic version of the available snake clone game in order to make the game more presentable and user friendly along with that is also specifies the functions used in the code.

**3.1 Basic Gameplay**

The concept of the snake game originated in 1976 arcade game Blockade was led through the tons of versions, before its one of the variant was developed for Nokia mobile phones in 1998. Here, we made use of the basic concept where snake travels through the provided space, feeding upon the food before clashing to itself or moving out of the space.

Earlier the gameplay was only the basic structure where the snake maneuvers in line to reach the food. In here graphics consists of red squares which represents food, and it has 4 directions. As the snake goes on grabbing the food its length increases

**3.2 Upgraded Version**

Following are the upgradations implemented in the previous available version:

* The start hand of the game displays a window which displays an option upon choosing which it opens a new window that shows the basic set of instruction to be followed while playing the game along with that it also provides second choice to start the game straight away if user already has the knowledge about the game and third option to terminate the game.
* The foremost variation made in the gameplay was to constantly display the score earned by the player as the game goes on.
* The next alteration made to the available game was to provide the interactive menus to the user, where the person playing the game has the convenience to choose the background color during the gameplay. This was implemented with the help of menus function specific to OpenGL where a programmer has the efficient method to include various option which can either be interactive or reflective based upon the gameplay, one intends to provide.
* In addition to interactive menu, a reflective change is included for which a code snippet has been added into the basic version which changes the background color automatically as soon as snake already feeds upon a certain numbers of food, or rather as soon as the player earns a certain number of points and continues to reflect the same as the player goes on earning the points further.
* Furthermore, an interesting parameter that has been suggested by this version of gameplay is to change the speed of the snake after the player has earned considerable number of points in order to raise the game’s difficulty.

**3.3 Functions**

#### Void glColor3f (float red, float green, float blue);

This function is used to mention the color in which the pixel should appear. The number 3 specifies the number of arguments that the function would take. ‘f ’ gives the type that is float. The arguments are in the order RGB (Red, Green, Blue). The color of the pixel can be specified as the combination of these 3 primary colors.

#### Void glClearColor (int red, int green, int blue, int alpha);

This function is used to clear the color of the screen. The 4 values that are passed as arguments for this function are (RED, GREEN, BLUE, ALPHA) where the red green and blue components are taken to set the background color and alpha is a value that specifies depth of the window. It is used for 3D images.

#### Void glutKeyboardFunc ();

void glutKeyboardFunc (unsigned char key, int x, int y);

Here glutKeyboardfunc is the new keyboard callback function. It sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII key strokes in the window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks.

#### Void glFlush ();

Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. **glFlush** empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

#### 

#### Void glMatrixMode (GLenum mode);

Where mode specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted:

**GL\_MODELVIEW**, **GL\_PROJECTION**, and **GL\_TEXTURE**.

The initial value is **GL\_MODELVIEW**.

**GL\_MODELVIEW** Applies subsequent matrix operations to the modelview matrix stack.

**GL\_PROJECTION** Applies subsequent matrix operations to the projection matrix stack.

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

Where x, y Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0, 0).Width, height Specify the width and height of the viewport. When a GL context is first attached to a surface (e.g. window), width and height are set to the dimensions of that surface. glViewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates. Let (xnd, ynd) be normalized device coordinates. Then the window coordinates (xw, yw) are computed as follows:

|  |
| --- |
| xw = ( xnd + 1 ) width/2 + x |
| yw = ( ynd + 1 ) height/2 + y |

Viewport width and height are silently clamped to a range that depends on the implementation. To query this range, call **glGetInteger** with argument GL\_MAX\_VIEWPORT\_DIMS.

#### void glutInit (int \*argc, char \*\*argv);

glutInit will initialize the GLUT library and negotiate a session with the window system. glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. Examples of this situation include the failure to connect to the window system, the lack of window system support for OpenGL, and invalid command line options. glutInit also processes command line options, but the specific options parse are window system dependent.

#### void glutReshapeFunc (void (\*func)(int width, int height));

glutReshapeFunc sets the reshape callback for the current window. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay forthe window is established. The width and height parameters of the callback specify the new window size in pixels. Before the callback, the current window is set to the window that has been reshaped.If a reshape callback is not registered for a window or NULL is passed to glutReshapeFunc (to deregister a previously registered callback), the default reshape callback is used. This default callback will simply call glViewport(0,0,width,height) on the normal plane (and on the overlay if one exists).If an overlay is established for the window, a single reshape callback is generated. It is the callback's responsibility to update both the normal plane and overlay for the window (changing the layer in use as necessary).

#### void glutMainLoop (void);

glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

#### glutPostRedisplay ();

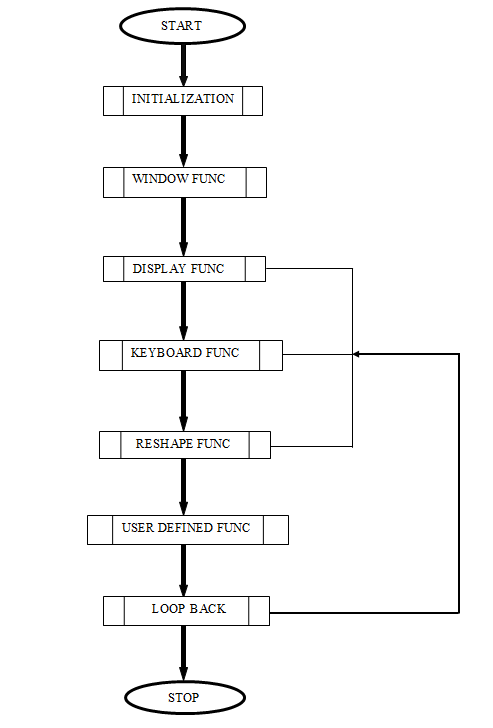
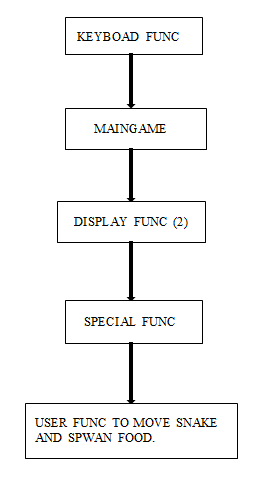
Mark the normal plane of *current window* as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display callback opportunity generates only a single redisplay callback. glutPostRedisplay may be called within a window's display or overlay display callback to re-mark that window for redisplay.

**CHAPTER 4**

**DESIGN FLOWCHART**

The sequential execution of the OpenGL program is represented by the following flowchart.

**Functions called in main().**

**Fig: 4.1: Flowchart**

**Functions called in Keyboard func of main() to maneuver the snake.**

In addition to these, a Timer Func is also added in the maingame(), that facilitates the automatic movement of snake in the direction specified each time in Special Func. The Timer Func() is passed parameters such as move\_speed of the snake and moveAuto() that is responsible for the movement of snake in the play area.

# CHAPTER 5

# IMPLEMENTATION

# Following are the modules with code snippet included in the source code that facilitates the snake clone game execution:

# Function to initialize snake body:

# void initGL()

# {

# glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

# glMatrixMode(GL\_PROJECTION);

# glLoadIdentity();

# // Specify the coordinates to each part of the snake

# int initSize = 3;

# for(int a = 1; a <= initSize; a++)

# {

# std::deque<float> row;

# row.push\_back(0.0f);

# row.push\_back((map\_half\_length + 2.0f + (initSize \* 2)) - (a \* 2));

# part\_coords.push\_front(row);

# }

# srand(time(NULL));

# }

# Function to make the snake move automatically:

# void moveSnakeAuto(int value)

# {

# if(!moved)

# {

# if(direction == UP)

# moveSnake(UP);

# else if(direction == DOWN)

# moveSnake(DOWN);

# else if(direction == LEFT)

# moveSnake(LEFT);

# else if(direction == RIGHT)

# moveSnake(RIGHT);

# }

# else

# moved = false;

# }

# glutTimerFunc(move\_speed, moveSnakeAuto, 0);

# }

# Function to move the snake:

# void moveSnake(int new\_direction){

# direction = new\_direction;

# int last\_part = part\_coords.size() - 1;

# std::deque<float> new\_head = part\_coords[last\_part];

# if(direction == UP){

# // Did we slither into ourself?

# for(unsigned int a = 0; a < part\_coords.size(); a++){

# if(part\_coords[0][0] == part\_coords[a][0] &&part\_coords[0][1] + 2.0f =part\_coords[a][1])

# exit(0);

# }

# if(part\_coords[0][1] == map\_half\_length) // Did we slither into a wall?

# exit(0);

# // Did we get food?

# if(part\_coords[0][0] == food\_coords[0] && part\_coords[0][1] + 2.0f == food\_coords[1]){

# growth\_stage++;

# food\_available = false;

# }

# new\_head[1] = part\_coords[0][1] + 2.0f;

# } else if(direction == DOWN){

# // Above process is repeated, but for negative Y axis with minor changes.

# 

# if(direction == LEFT){

# // Did we slither into ourself?

# for(unsigned int a = 0; a < part\_coords.size(); a++)

# if(part\_coords[0][0]-2.0f == part\_coords[a][0] && part\_coords[0][1] == part\_coords[a][1])

# exit(0);

# // Did we slither into a wall?

# if(part\_coords[0][0] == -map\_half\_length)

# exit(0);

# // Did we get food?

# if(part\_coords[0][0] - 2.0f == food\_coords[0] && part\_coords[0][1] == food\_coords[1])

# {

# growth\_stage++;

# food\_available = false;

# }

# new\_head[0] = part\_coords[0][0] - 2.0f;

# }

# else if(direction == RIGHT){

# // Same above process is repeated, but for positive X-axis with minor changes.

# glutPostRedisplay();

# }

# Function to spawn on food:

# void spawnFood(){

# if(!food\_available){

# while(true){

# bool collides = false;

# // Produce a temporary random coordinate

# int temp\_food\_coords[2] = { food\_coords[0] = 2 \* (rand() % ((int) map\_half\_length + 1)) - (int)map\_half\_length ,

# food\_coords[1] = 2 \* (rand() % ((int) map\_half\_length + 1)) - (int) map\_half\_length };

# //Does it collide with the snake?

# for(unsigned int a = 0; a < part\_coords.size(); a++){

# if(temp\_food\_coords[0] == part\_coords[a][0] &&temp\_food\_coords[1] == part\_coords[a][1])

# collides = true;

# }

# // If it doesn't collide with the snake, then make it the real food coordinates

# if(collides == false){

# food\_coords[0] = temp\_food\_coords[0];

# food\_coords[1] = temp\_food\_coords[1];

# food\_available = true;

# score=score+1;

# count++;

# if(count==6)

# { glClearColor(0.2,0.73,0.6,1.0);

# move\_speed=90;

# }

# if(count==10)

# {

# glClearColor(0.23,0.10,0,1.0);

# move\_speed=70;

# }

# //Similar if conditions can be added for further increase in values of count.

# break;

# }

# }

# }

# CHAPTER 6

# CONCLUSION

The code we have implemented for our project is working well to the best of our knowledge. This project is both informative and entertaining. This project provided an opportunity to learn the various concepts of the subject in detail and provided us a platform to express our creativity and imagination come true.

The full designing and creation of **Snake Clone** has been executed under Windows operating system using MS Visual Studio, this platform provides and satisfies the basic need of a good compiler. Use of glut.h library and built in functions make it easy to design good graphics package such as this simple game.

**Scope of Enhancement.**

The Snake Clone game still has a lot scope to be enhanced further from the user’s gameplay point of view such as:

* Graphically it can be further developed and improved to make it more appealing and attractive for which we can make use of better color combinations and color diffusion properties.
* More number of interactive choices can be included such as to provide users with the level options, choosing which user can have different level of difficulties and snake speed to play with.
* Levels to be provided to the user can have variety of obstacles in the play area, collision with which can lead to termination of the game. These obstacles/walls can be randomly generated or fixed at a particular positon in the play area.

# BIBLIOGRAPHY

### Reference Books

* “Interactive Computer Graphics – A Top-Down Approach using OpenGL” by EDWARD ANGEL.
* “Computer Graphics & Visualization” by A.P. Godse, D.A Godse.
* “Computer Graphics with OpenGL” by Donald D.Hearn.

### Internet source

* [www.wikipedia.com](http://www.wikipedia.com)
* [www.google.co.in](http://www.google.co.in)
* www.opengl.org
* [www.stackexchange.com](http://www.stackexchange.com)
* [www.stackoverflow.com](http://www.stackoverflow.com)

**APPENDIX A**

**ABBREVATION AND ACRONYMS**

**Short Form Expansion**

OpenGL Open Graphics Library

GLUT Graphics Library Utility Toolkit

GLU Graphics Language Utility

Env Environment

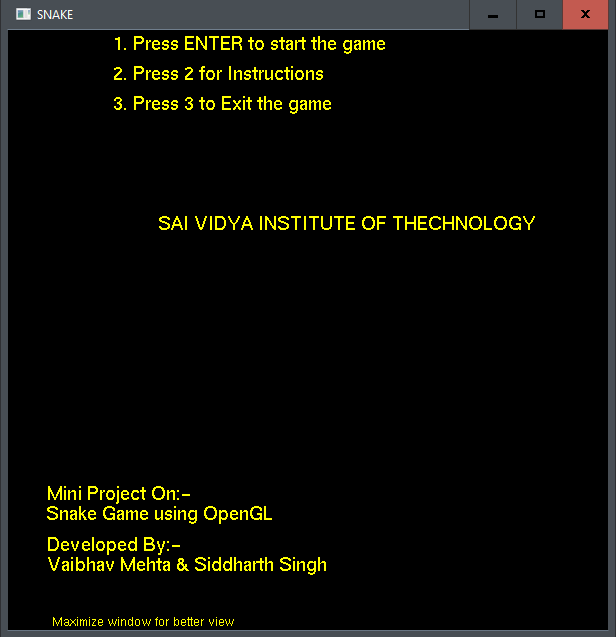
UI User Interface

API Application Programming Interface

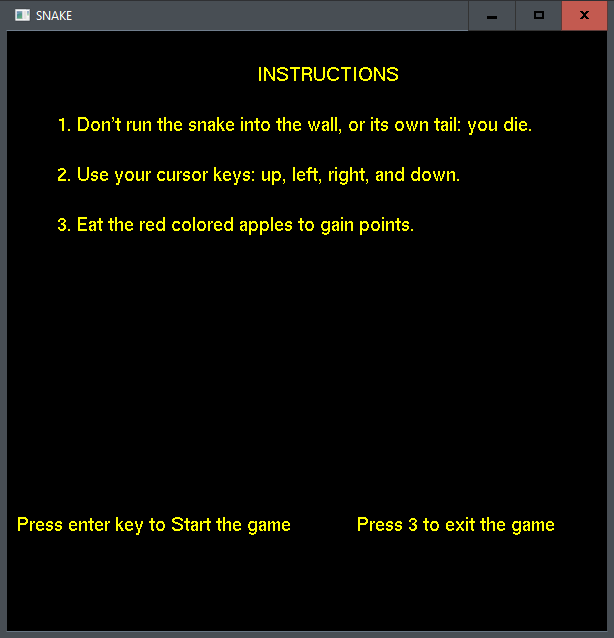
GL Graphics Library

## APPENDIX B

**SNAPSHOTS**

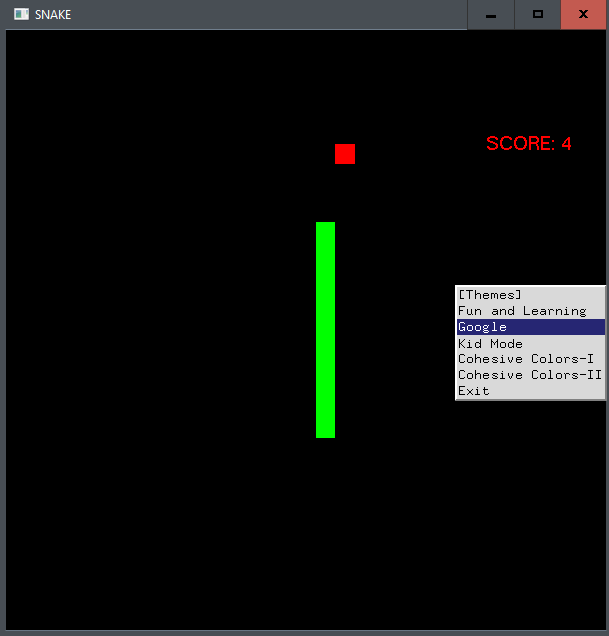


**Snapshot 1:** Front window of the game.

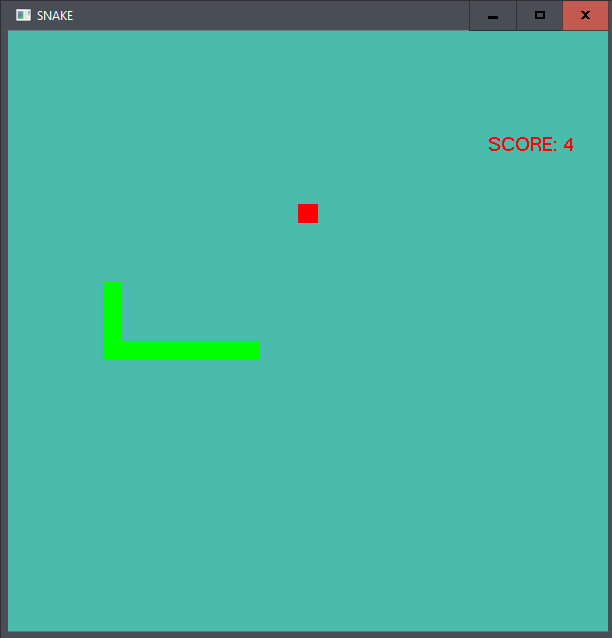


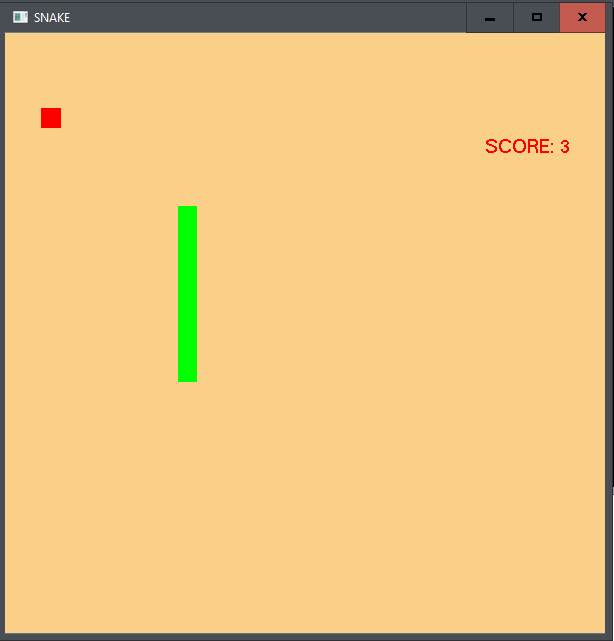
**Snapshot 2:** Instruction window of the game.

.



**Snapshot 3**: Interactive menus, where the person playing the game has the convenience to choose the background color during the gameplay.



**Snapshot 4:”**Fun and Learning” theme 

**Snapshot 5:”**Google” theme



**Snapshot 6:”**Cohesive Colors-II” theme