CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO OPENGL

Computer Graphics is a complex and diversified technology. To understand the technology it is necessary to subdivide it into manageable Parts. This can be accomplished by considering that the end product of computer graphics is a picture. The picture may, of course, be used for a large variety of purposes; e.g., it may be an engineering drawing, an exploded parts illustration for a service manual, a business graph, an architectural rendering for a proposed construction or design project, an advertising illustration, or a single frame from an animated movie. The picture is the fundamental cohesive concept in computer graphics.

Consider how: Pictures are represented in computer graphics.

- Pictures are prepared for presentation.
- Previously prepared pictures are presented.
- Interaction with the picture is accomplished.

Here “picture” is used in its broadest sense to mean any collection of lines, points, text, etc. displayed on a graphics device.

1.2 Computer Graphics

The totality of computer graphics software encompasses the concepts from data structures, from data base design and management, from the psychology, ergonometric of the man-machine interface, from programming languages and operating system.

Numerous computer graphics standards can be grouped following categories.

- First is the graphics application interface, where ideas are translated into a form that is understandable by a computer system. Current representative standards are the GKS, GKS-3D, and the Programmer’s Hierarchical Interactive Graphics Standards (PHIGS).
- The Second is concerned with the storage and transmission of data between graphics systems and between graphics-based computer aided design and computer aided manufacturing systems. The current standard in this area is the Initial Graphics Exchange Specification (IGES).

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.

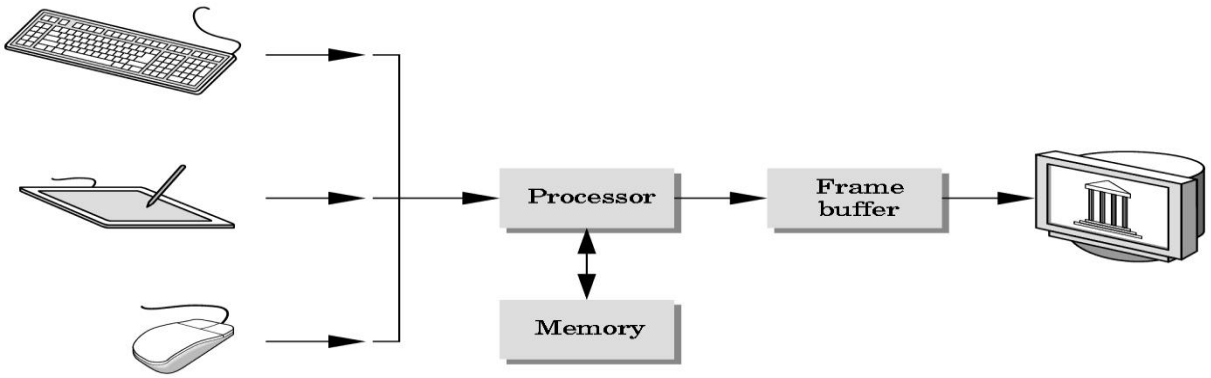

Fig. 1.1: A graphics system

1.1 A Brief History Of OpenGL

OpenGL was developed by Silicon Graphics and is popular in the video games industry where it competes with Direct3D on Microsoft Windows IrisGL, a propietary graphics API, is the precursor of OpenGL. It is developed by Silicon Graphics Inc. (SGI). IrisGL was used as the starting point of an open standard for computer graphics that would save time porting applications by avoiding direct hardware access. After SGI cleaned up IrisGL and opened up the standard to other companies, OpenGL was born.

In 1992 the OpenGL Architectural Review Board (OpenGL ARB) was established. The OpenGL ARB is a group of companies that maintain and update the OpenGL standard.

In 2003 the first OpenGL (Exchange Specification) ES specification was released. OpenGL ES is a subset of OpenGL designed for mobile phones, embedded devices and video game systems.

In 2004 the OpenGL 2.0 specification was released, including the GLSL (OpenGL Shading Language) specification.

In August 2008, the OpenGL 3.0 specification was released.

1.2.1 What is OpenGL?

- OpenGL is a Software Interface to Graphics Hardware
- OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms
- It Consists of about 250 Distinct Commands.
- It is Hardware-independent Interface
    - No command for windows or user input handling
    - Does not include low-level I/O management
- It is developed primarily by SGI
- It consists of 2D/3D graphics, lower-level primitives (polygons)
- It is Basis for higher-level libraries/toolkits

What does it do?

- 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).

1.2.2 Programming using OpenGL: A first Introduction

OpenGL is an API …

• Application programmers' interface: link between

- low-level: graphics hardware
- high-level: application program you write

OpenGL is a …

Library for 2D and 3D graphics programming

• 200+ functions for building application programs

• Portable to many platforms (Win, Mac, Unix, Linux)

• Callable from many programming languages (C, Java,Perl, Python)

• Primarily concerned with modeling and rendering

Operations

- Specify geometric primitives (lines, pixels, polygons ...)
- Apply geometric transformations

- Specify camera, light, color, texture information, etc.

• No windowing or (platform-specific) input/interaction

- functions— these are the jobs of GLUT

1.3 OpenGL-Related Libraries

OpenGL provides a powerful but primitive set of rendering commands, and all higher-level drawing must be done in terms of these commands. Also, OpenGL programs have to use the underlying mechanisms of the windowing system. A number of libraries exist to allow you to simplify your programming tasks, including the following:

- The OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces. GLU routines use the prefix glu.
- The OpenGL Utility Toolkit (GLUT) is a window system-independent toolkit, written by Mark Kilgard, to hide the complexities of differing window system APIs. GLUT is the subject of the next section, and it’s described in more detail in Mark Kilgard’s book OpenGL Programming forthe X Window System (ISBN 0-201-48359-9). GLUT routines use the prefix glut. "How to Obtain the Sample Code" in the Preface describes how to obtain the source code for GLUT, using ftp.

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 Hardware Requirements:

- Intel® Pentium 4 CPU and higher versions
- 128 MB or more RAM.
- A standard keyboard, and Microsoft compatible mouse
- VGA monitor.

2.2 Software requirements:

- The graphics package has been designed for OpenGL; hence the machine must Have code blocks
- Software installed preferably 6.0 or later versions with mouse driver installed.
- GLUT libraries, Glut utility toolkit must be available.
- Operating System**:** Windows
- Version of Operating System**:** Windows XP, Windows NT and Higher
- Language**:** C
- Code::Blocks: cross-platform Integrated Development Environment (IDE)

2.3 MISCELLANEOUS REQUIREMENTS:

All the required library and header files should be available in the include directories. The files associated with this editor should be placed in either the same folder or in a specified folder.

2.4 LANGUAGE USED IN CODING:

C/C++ and OpenGL as an API.

CHAPTER 3

SYSTEM DESIGN

Design of any software depends on the architecture of the machine on which that software runs, for which the designer needs to know the system architecture. Design process involves design of suitable algorithms, modules, subsystems, interfaces etc.

3.1 System Architecture

CONTROL FLOW DIAGRAM

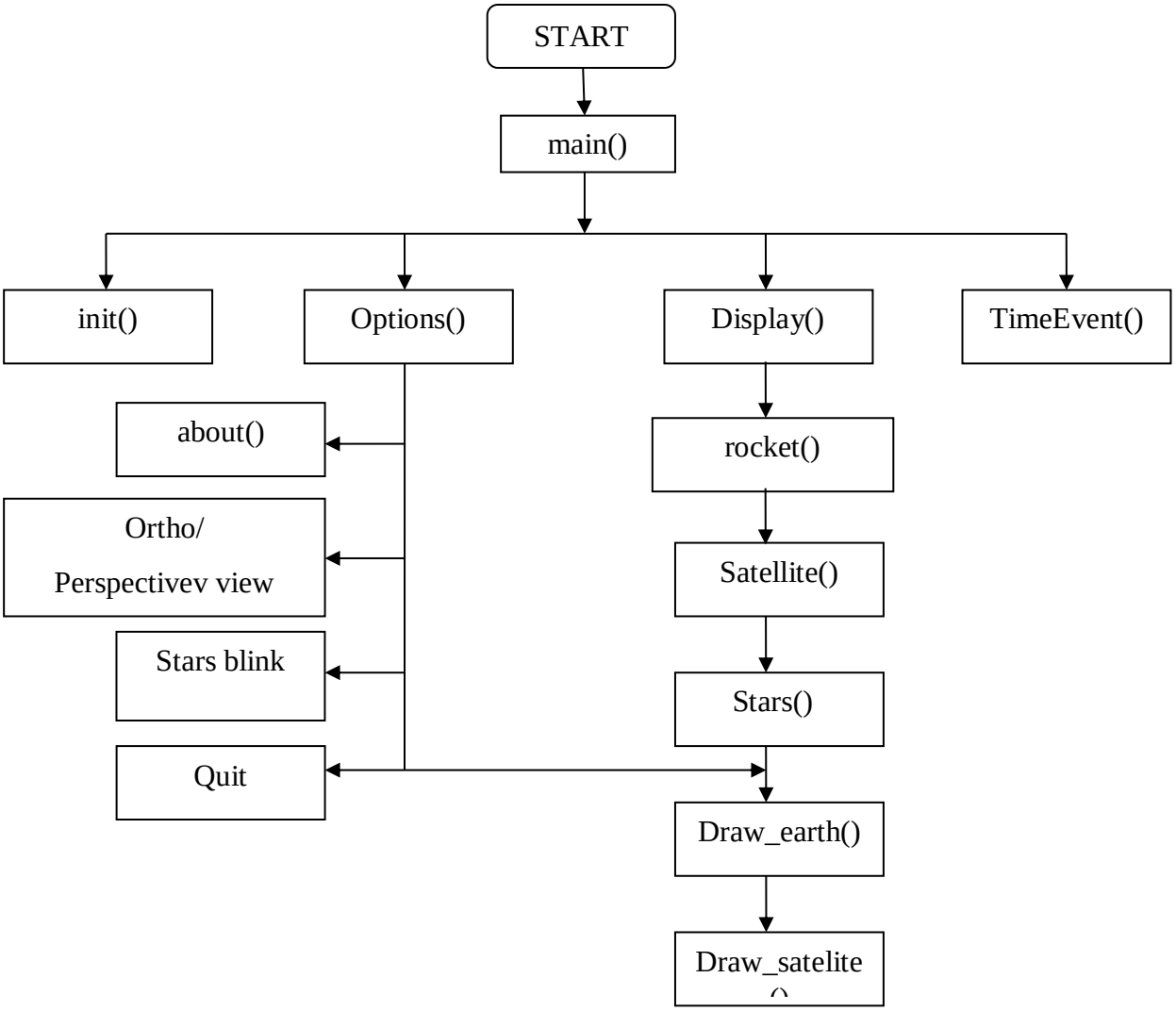

Fig. 3.1: System Control flow diagram

CHAPTER 4

IMPLEMENTATION

The implementation stage of this model involves the following phases.

- Implementation of OpenGL built in functions.
- User defined function Implementation.

4.1 Implementation of OpenGL Built In Functions used:

1. glutInit( ):

glutInit is used to initialize the GLUT library.

Usage: void glutInit(int *argcp, char **argv);

Description: glutInit will initialize the GLUT library and negotiate a session with the window system.

2. glutInitDisplayMode( ):

glutInitDisplayMode sets the initial display mode.

Usage: void glutInitDisplayMode (unsigned int mode);

Mode - Display mode, normally the bitwise OR-ing of GLUT display mode bit masks.

Description: The initial display mode is used when creating top-level windows, sub windows, and overlays to determine the OpenGL display mode for the to-be-created window or overlay.

3. glutCreateWindow( ):

glutCreateWindow creates a top-level window.

Usage: int glutCreateWindow(char *name);

Name - ASCII character string for use as window name.

Description: glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window. Each created window has a unique associated OpenGL context.

4. glutDisplayFunc( ):

glutDisplayFunc sets the display callback for the current window.

Usage: void glutDisplayFunc(void (*func)(void));

Func - The new display callback function.

Description: glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

5. glutMainLoop( ):

glutMainLoop enters the GLUT event processing loop.

Usage: void glutMainLoop(void);

Description: 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.

6. glMatrixMode( ):

The two most important matrices are the model-view and projection matrix. At any time, the state includes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.

.

4.2 Implementation of User Defined Functions:

1. void init(void):

Here the initialization function init() is used to set the OpenGL state variables dealing with viewing and attributes-parameters that is prefered to set once, independently of the display function.

2. void stroke_output(GLfloat x, GLfloat y, char *format,...):

This is the function which is used on the display screen to display the choices and to select those choices for the required output. Like if we press ‘P’ rocket will launch and if we press ‘S’ satellite will rotate around the earth.

3. void satellite():

This is the function which is used to create the satellite core and its panels and its feature is after rocket launches the satellite comes out of the rocket and starts rotating around the earth.

4.void main(int argc, char ** argv):

This is the function from where the execution of the program begins.In this function we have certain "gl" functions used to initialize the basic glut window with desired properties. After initialization the display function is called with respect to the window and finally the last statement of this section is the glutMainLoop(). Which is an event processing function that enters into an infinite loop.

5. void rocket():

This is the function which is used to create the main soul of the project that is rocket and it is scaled in such a way that it changes its state for different values of x.

6. **void stars():**

This is the function which is used to create the stars in the sky which has a blinking effect and which makes the outlook the project well.

CHAPTER 5

SNAPSHOTS

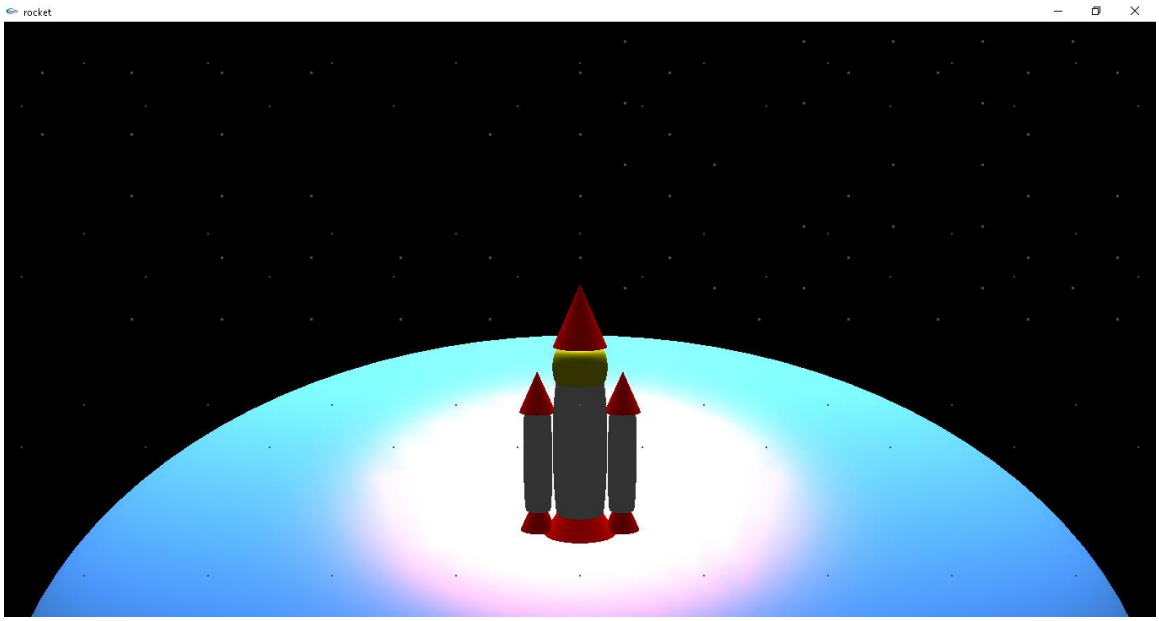

Fig. 5.1: Initial Stage Of Rocket

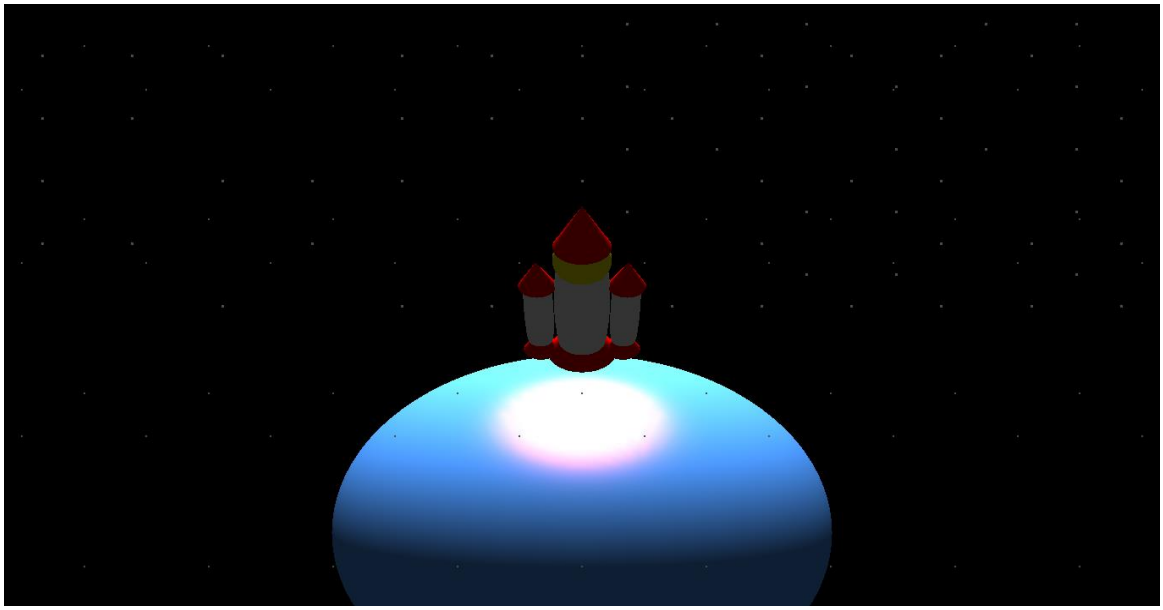

Fig. 5.2: Rocket Launch

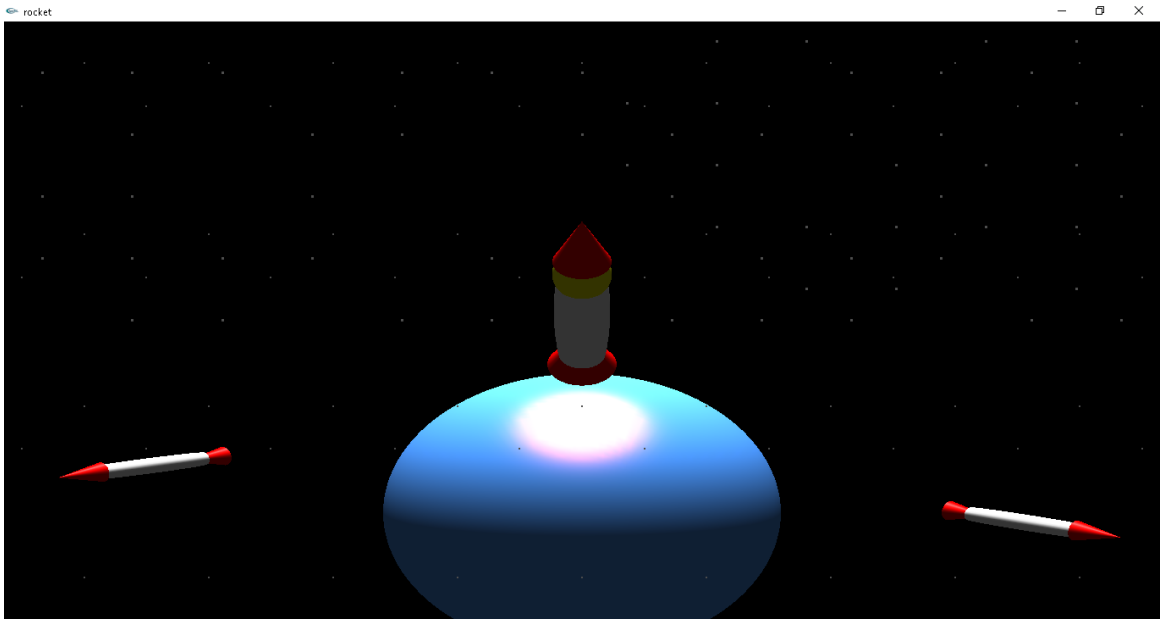

Fig. 5.3: Splitting of components from rocket

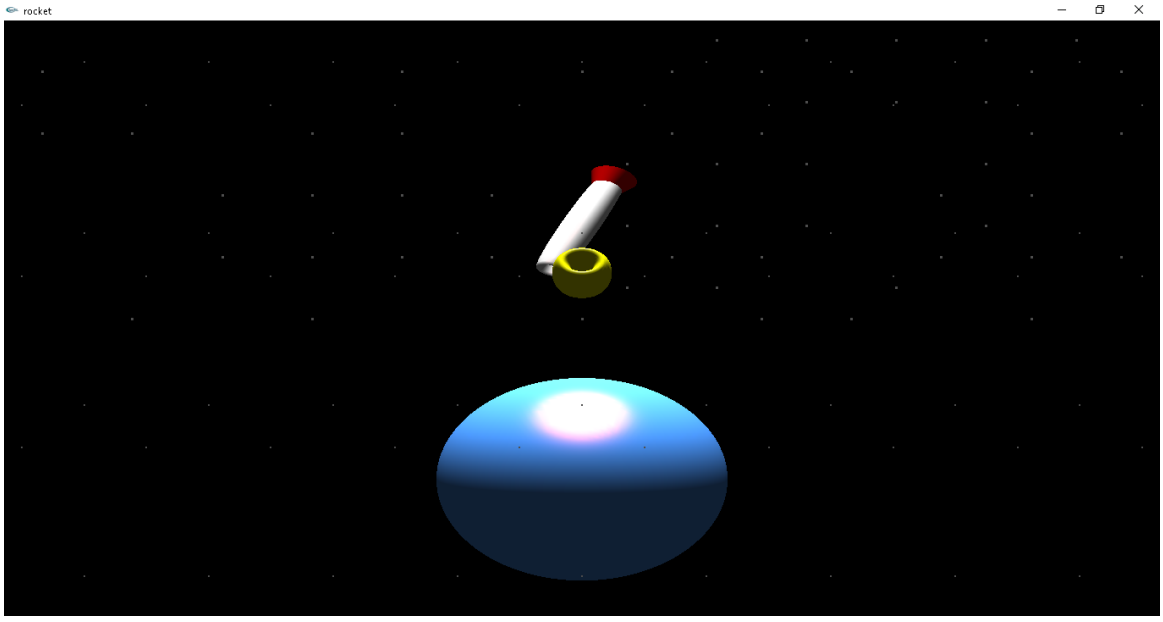

Fig. 5.4: Emergence of satellite component from rocket

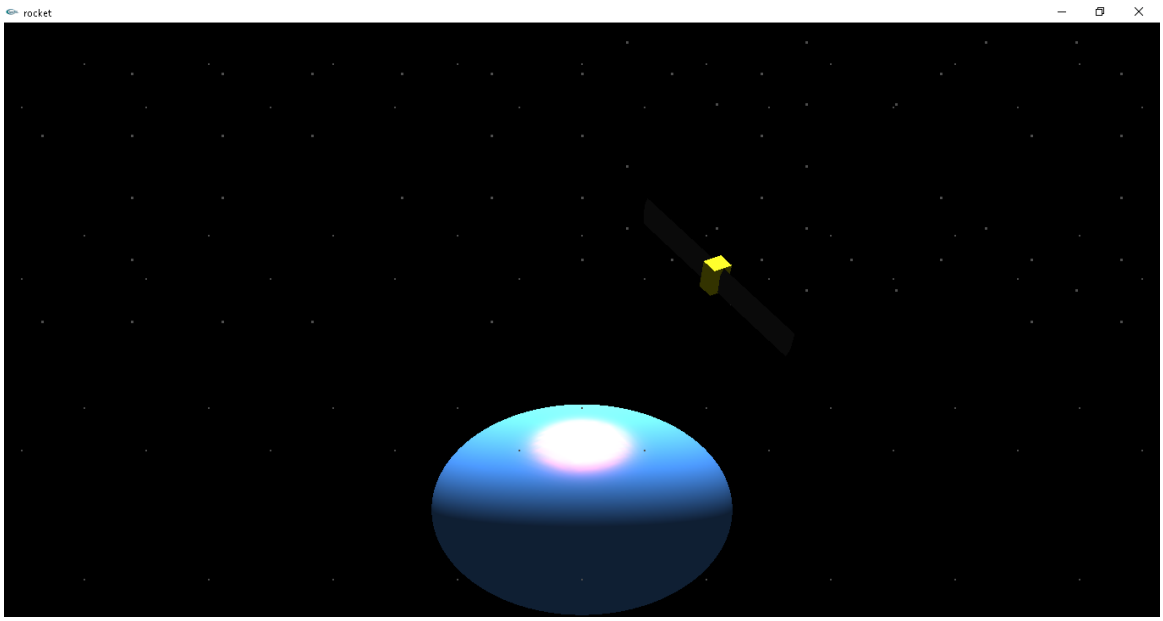

Fig. 5.5: Emergence of satellite from rocket

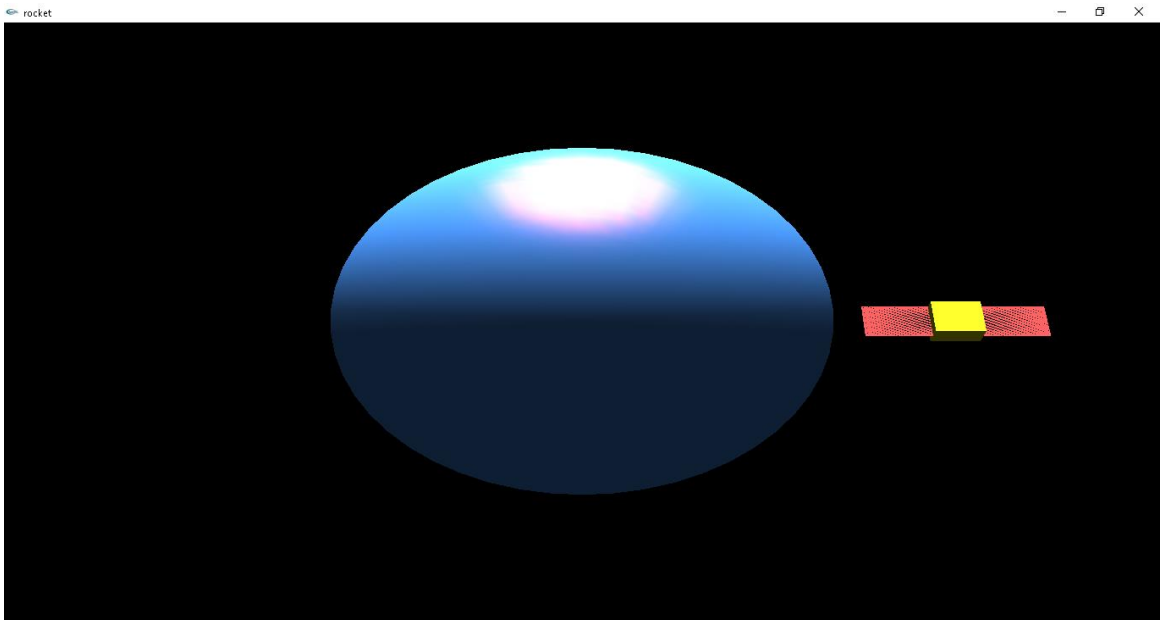

Fig. 5.6: Rotation of satellite around earth

CHAPTER 6

FUTURE ENHANCEMENT

We have implemented the essential features of the Analog Clock to our best knowledge. Even still, we would like to enhance the quality and appearance of the clock in the following ways:

- 3D View in a better way
- Enhanced appearance
- Lighting and shading
- Implementation of transmission from satellite

CHAPTER 7

CONCLUSION

Thus in this project we have acquired a lot of knowledge about various techniques in OpenGL programming. We have explored many new concepts on the World Wide Web, such as Texture mapping, randomizing, color swapping etc. elaborate the paragraph with the content.

Finally we conclude that this program clearly illustrate the Rocket launch by using OpenGL and pre-built objects and has been completed successfully and is ready to be demonstrated.

REFERENCES

Books:

Interactive Computer Graphics, 5th Edition, Edward Angel

Computer Graphics and Multimedia, Udit Sharma

Websites:

- http://www.amazon.in/Computer-Graphics-Multimedia-Udit-Agarwal/dp/935014316X?tag=googinhydr18418-21
- http://en.wikipedia.org/wiki/Computer_graphics
- http://stackoverflow.com/questionsquickly
- http://www.opengl-tutorial.org/intermediate-tutorials/
- http://www.opengl-tutorial.org/
- https://open.gl/
- http://www.cs.uccs.edu/~ssemwal/indexGLTutorial.html
- http://www.videotutorialsrock.com/
- http://ogldev.atspace.co.uk/
- https://www.opengl.org/sdk/docs/tutorials/
- http://learnopengl.com/
- http://lazyfoo.net/tutorials/OpenGL/
- http://en.wikibooks.org/wiki/OpenGL_Programming

APPENDIX-A

A.1 Source code

```
#include <windows.h>
#include<string.h>
#include<stdarg.h>
#include<stdio.h>
#include <GL/glut.h>
static double x=0.0,x1=0.0,y1=0.1,z1=0.0,a1=0,y2=0,z2=0;
static float rx[100]={0}, ry[100]={0};
static double w1=0,w2=0,w3=0;
static bool transmit=false;
//static double move=-60;
//static bool seperate=false;
void stroke_output(GLfloat x, GLfloat y, const char *format,...)
{
        va_list args;
        char buffer[200], *p;
        va_start(args, format);
        vsprintf(buffer, format, args);
        va_end(args);
        glPushMatrix();
        glTranslatef(-2.5, y, 0);
        glScaled(0.003, 0.005, 0.005);
        for (p = buffer; *p; p++)
   glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
        glPopMatrix();
}
void waves(){
        glPushMatrix();
```

```
        glTranslatef(0,1,0);
        glScaled(0.05,0.5,0.1);
        glutSolidCube(0.5);
        glPopMatrix();

        glPushMatrix();
        glRotatef(-8,0,0,1);
        glTranslatef(0.01,1,0);
        glScaled(0.05,0.5,0.1);
        glutSolidCube(0.5);
        glPopMatrix();

        glPushMatrix();
        glRotatef(8,0,0,1);
        glTranslatef(-0.01,1,0);
        glScaled(0.05,0.6,0.1);
        glutSolidCube(0.5);
        glPopMatrix();
}
void stars1(){
int count = 0;
glPointSize(2.5);
        for(int j=0;j<100;j++)
        {   count++;
                for(int i=0;i<100;i++)
                {
                 rx[j]=rand()/500;
                                ry[i]=rand()/500;
        glBegin(GL_POINTS);
                        glColor3f(1,1,1);
```

```
if(count%2 == 0){
          glVertex3f(-6+rx[j],ry[i],-5);
         }
          else{
           glVertex3f(0.5+rx[j],ry[i]+0.5,-5);}
                                glEnd();}                    }
}
void stars(){
int count = 0;
glPointSize(1.5);
           for(float s1=-5;s1<=500; s1+=1.0){
          count++;
                    for(float s2=-6;s2<=6;s2+=1.0){
           glPushMatrix();
           glBegin(GL_POINTS);
           //if(!(s1<0 && s1 > -10)){
           if(count%2 == 0){
           glVertex3f(s2+0.5,s1+0.5,0);
           }else{
                    glVertex3f(s2,s1,0);}
 int count1 = 0;
           for(float s3=100;s3<=100; s3+=1.0){
          count1 ++;
                    for(float s4=-6;s4<=6;s4+=1.0){
           glPushMatrix();
           glBegin(GL_POINTS);
                    glVertex3f(s3,s4,0);
glEnd();
glPopMatrix();}
}
```

A.2 User Manual

Right click on the visualizer window, you will be asked for four options

1. Launch
2. Rotate
3. Quit

Depending on the selection the corresponding output will be displayed.

A.3 Personal Details

NAME: ANUSHA ANIL SHET

USN: 1DT16CS012

SEMESTER AND SECTION: 6^{TH} SEM, A SEC

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

EMAIL ID: anushaanil44@gmail.com