VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018

A COMPUTER GRAPHICS & VISUALIZATION MINI PROJECT REPORT ON

"WORKING OF SATELLITE VISUALIZATION"

A Project report Submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

BHAVANA SK (1RG20CS019)

MADHU S (1RG20CS033)

Under The Guidance of

MRS. BHAGYASHRI WAKDE

Assistant Professor

Dept. of CSE RGIT,

Bengaluru- 32

Department of Computer Science & Engineering

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

Cholanagar, R. T. Nagar Post, Bengaluru-560032

2022-2023

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University)

Cholanagar, R.T. Nagar Post, Bengaluru-560032

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the Mini Project Report entitled **“WORKING OF SATELITE VISUALIZATION”** is a bonafide work carried out by **Ms. BHAVANA SK (1RG20CS019), Mr. MADHU S (1RG20CS033)** in partial fulfillment for the award of **Bachelor of Engineering in Computer Science Engineering** under **Visvesvaraya Technological University, Belagavi,** during the year **2022-2023.** It is certified that all corrections/suggestions given for Internal Assessment have been incorporated in the report. This Computer Graphics & Visualization Mini Project report has been approved as itsatisfies the academic requirements.

Signature of guide	Signature of HOD	Signature of Principal
Mrs. Bhagyashri Wakde	**Mrs. Arudra A**	**Dr.D.G. Anand**
Assistant Professor,	Professor & HOD,	Principal,
Dept. of CSE	Dept. of CSE	RGIT,
RGIT, Bengaluru- 32	RGIT, Bengaluru-32	Bengaluru-32

External viva

Name of Examiner	Signature with date
1.	
2.	

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We hereby declare that the mini project work entitled **"WORKING OF SATELLITE VISUALIZATION"** submitted to the **Visvesvaraya Technological University, Belagavi** during the academic year **2022-2023**, is record of an original work done by us under the guidanceof **MRS. BHAGYASHRI WAKDE, Assistant Professor, Department of Computer Science and Engineering, Rajiv Gandhi Institute of Technology, Bengaluru** and this project work is submitted in the partial fulfillment of requirements for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering.** The results embodied in this thesis have notbeen submitted to any other University or Institute for award of any degree or diploma.

BHAVANA SK (1RG20CS018)

MADHU S (1RG20CS033)

ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude and respect to the **Rajiv Gandhi Institute of Technology, Bengaluru** for providing us an opportunity to carry out my project work.

We express our sincere regards and thanks to **Dr. D.G ANAND,** Principal, RGIT, Bengaluru, and **Mrs. ARUDRA A**, Professor and HOD, Department of Computer Science & Engineering, RGIT, Bengaluru, for their encouragement and support throughout the Project.

With profound sense of gratitude, we acknowledge the guidance and support extended by **MRS.BHAGYASHRI WAKDE**, Assistant Professor, Department of Computer Science & Engineering, RGIT, Bengaluru. Her incessant encouragement and valuable technical support have been of immense help in realizing this project. Her guidance gave us the environment to enhance our knowledge, skills and to reach the pinnacle with sheer determination, dedication and hard work.

We also extend our thanks to the entire faculty of the Department of CSE, RGIT, Bengaluru, who have encouraged throughout the course of Bachelor Degree.

BHAVANA S K (1RG20CS019)

MADHU S (1RG20CS033)

ABSTRACT

The “**Working of Satellite Visualization”** architecture model is depicted using the computer graphics, whichis developed on OpenGL programming interface. The project is a Visualization and it indicates the modelof working of satellite. The main aim of this mini project is to illustrate the concepts of working of a satellite in OpenGL. A satellite is an object which has been placed into orbit by human endeavor. Such objects are sometimes called artificial satellites to distinguish them from natural satellites, such as the moon satellites are used for a large number of purposes. We have used input devices like the mouse andkeyboard to interact with the program. We also used SolidCube for forming a complete network setup which helps to understand the concept of congestion control very well. To differentiate between objects, we have used different colors for different objects.

TABLES OF CONTENTS

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is concerned with all aspect of producing picture or images using computer the field began humbly almost 50 years ago with the display of few lines in cathode ray tube. Now we can create by computer, pictures that are indistinguishable from photograph of real object. We routinely train pilot with simulated airplane generating graphical display of a virtual environment in real time. Feature length movies made entirely by computer have been successful both critically and financially. Massive multi player game can involve tens of thousands ofconcurrent participants.

1.2 OPENGL

OpenGL is a graphics application programming interface (API) which was originally developed by Silicon Graphics. OpenGL is not in itself a programming language, like C++, but functions as an API which can be used as a software development tool for graphics applications. The term Open is significant in that OpenGL is operating system independent. GL refers to graphics language. OpenGL also contains a standard library referred to as the OpenGL Utilities (GLU). GLU contains routines for setting up viewing projection matrices and describing complex objects with line and polygon approximations. OpenGL gives the programmer an interface with the graphics hardware. OpenGL is a low-level, widely supported modeling and rendering software package, available on all platforms. It can be used in a range of graphics applications, such as games, CAD design, or modeling.

OpenGL is the core graphics rendering option for many 3D games, such as Quake 3. The providing of only low-level rendering routines is fully intentional because this gives the programmer a great control and flexibility in his applications. These routines can easily be used to build high-level rendering and modeling libraries. The OpenGL Utility Library (GLU) does exactly this, and is included in most OpenGL Unlike DirectX, OpenGL is only a graphics API; it doesn't include support for functionality such as sound, input, or networking (or anything not related to graphics). OpenGL was originally developed in 1992 by Silicon Graphics, Inc, (SGI) as a multi-purpose, platform independent graphics API. Since 1992 all of the development of OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three- dimensional applications. OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL.

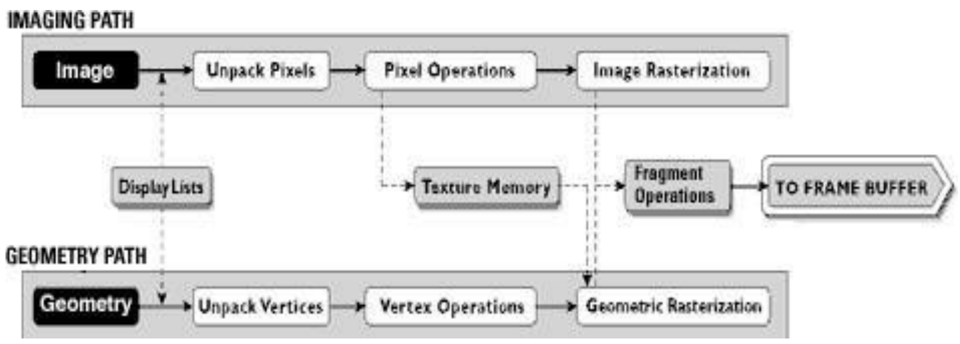


Fig1: OpenGL operates on image data as well as geometric primitives**.**

Elements of the OpenGL state—even the contents of the texture memory and the frame buffer canbe obtained by an OpenGL application. OpenGL also supports visualization applications with 2D images treated as types of primitives that can be manipulated just like 3D geometric objects. As shown in the OpenGL visualization programming pipeline diagram above, images and vertices defining geometric primitives are passed through the OpenGL pipeline to the frame buffer.

Basic OpenGL Operation

The figure shown below gives an abstract, high-level block diagram of how OpenGL processes data. In the diagram, commands enter from the left and proceed through what can be thought of as a processing pipeline. Some commands specify geometric objects to be drawn, and others control how the objects are handled during the various processing stages.

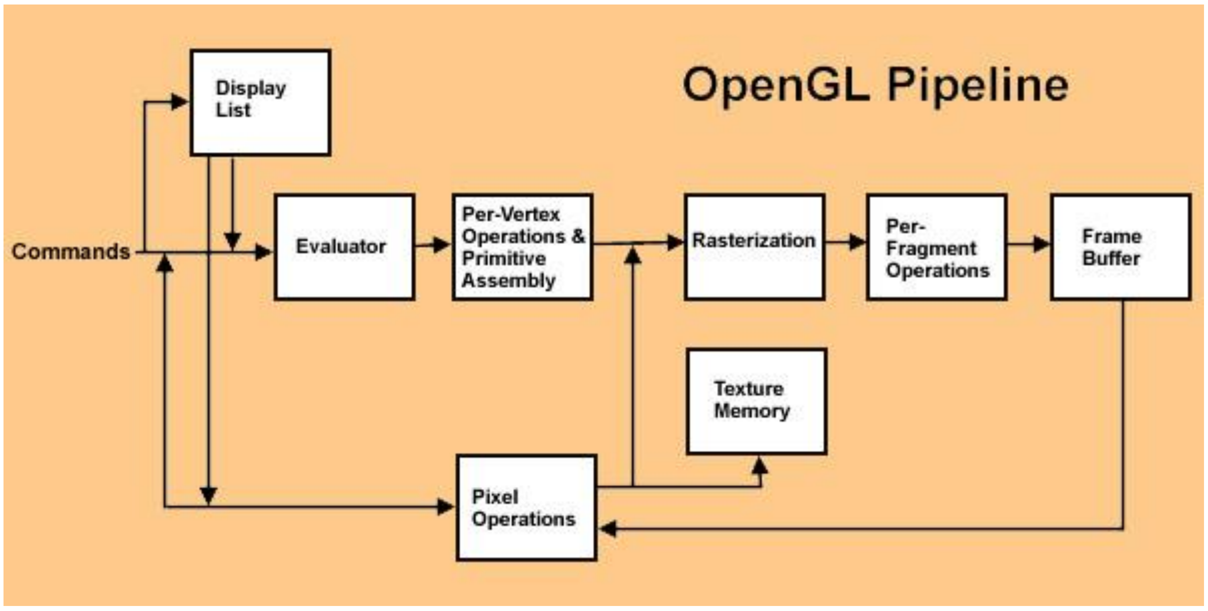


Fig2: OpenGL Block Diagram

As shown by the first block in the diagram, rather than having all commands proceed immediately through the pipeline, you can choose to accumulate some of them in a display list for processing at a later time. Rasterization produces a series of frame buffer addresses and associated values using a two-dimensional description of a point, line segment, or polygon. Each fragment so produced is fed into the last stage, per-fragment operations, which performs the final operations on the data before it's stored as pixels in the frame buffer. These operations include conditional updates to the frame buffer based on incoming and previously stored z-value s (for z-buffering) and blending of incoming pixel colors with stored colors, as well as masking and other logical operations on pixel values. All elements of OpenGL state, including the contents of the texture memory and even of the frame buffer, can be obtained by an OpenGL application.

1.3 APPLICATION PROGRAMMING INTERFACE

API is used to study computer graphics. It includes openGL functions. Thus, API can be described through the functions in its library. A good API may contain hundreds of functions, so it is helpful to divide them into seven majorgroups.

1.Primitive functions

2.Attribute functions

3.Viewing functions

4.Transformation function

5.Input functions

6.Control functions

7.Query functions

The primitive functions define the low-level objects that the system display. Depending on the API, the primitives can include points, line, segments, polygons and various types of curves and surfaces. The attribute functions help to perform operations ranging from choosing the color with which a line segment can display, a pattern can pick within a polygon and for the titles of graph. The viewing functions allow specifying various views, although APIs differ in the degree of flexibility they provide in choosing a view. The transformation functions allow carrying out transformations of objects such as rotation, translation and scaling. The input functions allow dealing with the diverse forms of input that characterize modern graphics systems. The control functions enable to communicate with window system, to initialize programs, and to deal with any errors that take place during the execution of programs.

1.4 ABOUT THE PROJECT

Eyes in the sky, space mirror bouncing phone call round earth, heavenly composes helping us home – these are just three of the things that satellites do for us.

What is a satellite?

“Satellite”, means a smaller, space-based object moving in the loop (an orbit) around a larger object. For example: Moon is a natural satellite of earth, because gravity locks it in orbit around our planet.

How does satellite help us?

1. If you want to make a phone call from a north pole, you can fire a signal into space and back down again, using a communications satellite as a mirror to bounce the signal back to the earth and its destination.

2. If you want to survey crops or ocean temperatures, you could do it this from plane, but a satellite can capture more data more quickly because it's a higher up and further along.

Similarly, if you want to drive somewhere you have never been before, you could study maps or ask random strangers for directions, or you could use signals from satellites to guide you instead.

What do satellites do for us?

A satellite does usually determine the things like, how far away from earth it needs to be, how fast it has to move, and the orbit it as to follow.

The three main uses of satellites are:

1.Communications.

2.Photography, imaging and scientific surveying.

3.Navigation.

The main aim of this project is to illustrate the concepts of working of a satellite in OPENGL. A satellite is an object which has been placed into orbit by human endeavor. Such objects are sometimes called artificial satellites to distinguish them from natural satellites such as the moon. This pushed the entire network into a 'congestion collapse' where most packets were lost and the resultant throughput was negligible. We have added menu which makes the program more interactive. In this project we have used a small SolidCube to represent a data, which travels as data transfer from source to destination. We have used font family for indicating the name of objects aswe can see in this project.

CHAPTER 2

OPENGL SYNTAX

1.glutInit():interaction between the window in system and OPENGL is initiated.

2.glutInitDisplayModel(): used when double buffering is required and depth information is required.

3.glutCreateWindow() :this opens the OPENGL window and displays the title at top of the window.

4.glutInitWindowSize():specifies the size of the window.

5.glutInitWindowPosition() :specifies the position of the window in screen co-ordinates.

6.glutKeyboardFunc() handles normal ascii symbols.

7.glutReshapeFunc():sets up the callback function for reshaping the window.

8.glutIdleFunc() :this handles the processing of the background.

9.glutDisplayFunc(): this handles redrawing of the window.

10.glutMainLoop():this starts the main loop, it never returns.

11.glVertex3fv():used to set up the points or vertices in three dimensions.

12.glcolor3fv() used to render color to faces.

13.glFlush() used to flush the pipeline.

14.glutPostRedisplay() used to trigger an automatic redrawal of the object.

15.glMatrixMode() used to set up the required mode of the object.

16.glLoadIdentity():used to load or initialize to the identity matrix.

17.glTranslatef():used to translate or move the rotation centre from one point to another in three dimensions.

18.glRotatef(): used to rotate an object through a specified rotation angle.

19.glutSpecialFunc():handles special keyboard keys.

20.glViewPort() : used to set up the viewport.

SYSTEM REQUIREMENTS

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 SOFTWARE REQUIREMENTS

Programming Language: C++ using OpenGL.

Operating System: Windows Operating System.

Compiler: GCC compiler.

Graphics library: GL/glut.h, OpenGL 17.12.

3.2 HARDWARE REQUIREMENTS

RAM: 2GB and higher>

Hard Disk: 40GB and higher.

Monitor: 1024 x 768 display Resolution.

SYSTEM IMPLEMENTATION

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 SOURCE CODE

```
#include<windows.h>
#include <string.h>
#include <stdarg.h>
#include <stdio.h>
#include <glut.h>
#include <math.h>

Static double x=0.0; Static double move= -60;
Static float rx[100]={0},ry[100]={0};

//control waves

Static double w1=0,w2=0,w3=0;Static bool transmit=false;

Void *font;
Void *currentfont;

Void setFont(void *font)

{
Currentfont=font;
}

Void drawstring(float x,float y,float z,char *string)

{
Char *C; glRasterPos3f(x,y,z); for(C=string;*C!='\0';C++)

{
glColor3f(0.0,1.0,1.0); glutBitmapCharacter(currentfont,*c);
}
}
```

```
    Void Stroke_output(GLfloat x,GLfloat y,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 satellite(){ glRotatef(60,1,0,0);
    //body

    glPushMatrix();
    glColor3f(0.2,0.2,0.2);
    glScaled(1,0.6,1);
    glTranslatef(3.0,0,0.0);
    glutSolidCube(0.4);
    glPopMatrix();

    //Solar Panels

    glPushMatrix();
     glColor3f(0.3,0.3,0.3);
    glTranslatef(3,0,0.0);
    //glRotatef(45,1,0,0);
    glScaled(3.7,0.0,1);
     glutSolidCube(0.4);
     glPopMatrix();

    glPushMatrix();
     glColor3f(0.2,0.1,0.1);
    glTranslatef(3.0,0,-0.4);
    glScaled(0.5,0.5,0.5);
     glutSolidSphere(0.3,50,50);
    glPopMatrix();
```

```
}
Void sat1(){
glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0f,0.0f,-13.0f);
    //glRotatef(x,0.0f,1.0f,0.0f);

          //Moon

          glPushMatrix();
          glColor3f(1,1,1);
          glTranslatef(-
          3.8,2.8,0);
          glScaled(0.5,0.5,0.1);
          glutSolidSphere(0.6,50
          ,50);glPopMatrix();

          //Earth

          glPushMatrix();
          glColor3f(0.2,0.
          2,1);
          glTranslatef(0,-
          12,0);
          //glScaled(0.8,0.04,0.8);
          glutSolidSphere(10.0,50,50);
          glPopMatrix();

          //Building Center

          glPushMatrix();
          glColor3f(0,1,1);
          glRotatef(10,1,0,
          0); building(1.2,-
          1.2,3.2);
          glPopMatrix();

          //Building Left

          glPushMatrix();
          glColor3f(0,1,1)
          ;
          glRotatef(5,0,0,
          1); building(-
          3.8,-1.2,0);
          glPopMatrix();
```

```
//Signal glPushMatrix();
glColor3f(0,0,1);
if(transmit){

glRotatef(-25,0,0,1);
glTranslatef(-1.25,-
1.6+w1,0);
}else
glTranslatef(1,20,3.3);
Waves();

glPopMatrix();


//Main Dish

//Tower

glPushMatrix();
glColor3f(1,1,1);
glTranslatef(-1,-2,4);
glRotatef(270,1,0,0);
glScaled(1.0,1,2.0);
glutWireCone(0.5,1.4
,10);glPopMatrix();

//Dish

glPushMatrix();
glColor3f(1,1,1);
glTranslatef(-1.08,0.2,3);
glRotatef(60,1,0,0);
glScaled(0.7,1.3,0.7);
glutSolidCone(0.4,0.5,20
,20);glPopMatrix();

//Building Right

glPushMatrix(
);
glColor3f(1,1,
1); glRotatef(-
5,0,0,1);
building(3.8,-
1.2,0);
glPopMatrix();

//Satellite

glPushMatrix();
glTranslatef(-
   3,3.0,0);
   satellite();
   glPopMatrix();
 //Stars

 glPointSize(5);
 for(int
 j=0;j<100;j++)
 {
         for(int i=0;i<100;i++)
          {
                 rx[j]=rand()/500;
                          ry[i]=rand()/500
                 ;
                 glBegin(GL_POINTS);
                 glColor3f(0,2,2);
                 glvertex3f(-
                 6+rx[j],ry[i],-5);
          }
  }

          glPushMatrix();

  //glScaled(1.1,2.0,0.1);
  glTranslatef(0.0,0.0,-2.0);
  setFont(GLUT_BITMAP_TIMES_ROMAN_24);
  glColor3f(1,1,1);
  drawstring(1,3.7,-
  1.0,"Satellite");
  setFont(GLUT_BITMAP_TIMES_ROMAN_24);
  glColor3f(1,1,1);
  drawstring(-4.4,.5,-1.0,"Receiver");
  setFont(GLUT_BITMAP_TIMES_ROMAN_24);
  glColor3f(1,1,0);
  drawstring(0,-
  2,7,"Receiver");
  setFont(GLUT_BITMAP_TIMES_ROMAN_24);
  glColor3f(1,1,1);
  drawstring(-1.5,-1,-1.0,"Transmitter");
  setFont(GLUT_BITMAP_TIMES_ROMAN_24);
  glColor3f(1,1,1);
  drawstring(3.2,1,3,"Receiver");
  glPopMatrix();

  glFlush();

glutSwapBuff

ers();
```

```
}

    Satellite(); glFlush();
    glutSwapBuffers();
}
/*select the projection matrix and reset it then setup our view perspective */

glMatrixMode(GL_PROJECTTION);
glLoadIdentity();
glutPerspective(30.0f,(GLfloat)640/(GLfloat)480,0.1f,200.0f);

/*Select the modelview matrix, which we alter with

rotate() */glMatrixModel(GL_MODELVIEW);
glLoadIdentity();
glCleardepth(2.0f);
glEnable(GL_DEPTH_TE
ST);
                    glEnable(GL_COLOR_MATERIAL);
glDepthFunc(GL_LEQUAL);
}

Void display()

{

   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glTranslatef(0.0f,0.0f,-13.0f);
   stroke_output(-2.0,1.7,"s/S--
   >Start");
   stroke_output(-2.0,0.9,"t-->Transmit");
   stroke_output(-2.0,0.0,"q/Q-->Quit");

   GLfloat
   mat_ambient={0.0f,1.0f,2.0f,1.0f};
   GLfloat
   mat_diffuse[]={0.0f,1.5f,.5f,1.0f};
   GLfloat
   mat_specular[]={5.0f,1.0f,1.0f,1.0f};
   GLfloat mat_shininess[]={50.0f};
   glMaterialfv(GL_FRONT,GL_AMBIENT,mat_ambient);
   glMaterialfv(GL_FRONT,GL_DIFFUSE,mat_diffuse);
   glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
   glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);

   GLfloat
   lightIntensity[]={1.7f,1.7f,1.7f,1.0f};
   GLfloat
   light_position3[]={0.0f,8.0f,10.0f,0.0f};
```

```
  glLightfv(GL_LIGHT0,GL_POSITION,light_position3);
  glLightfv(GL_LIGHT0,GL_DIFFUSE,lightIntensity);

  GLfloat
  lightIntensity1[]={1.7f,1.7f,1.7f,1.0f};
  GLfloat light_position31[]={-
  2.0f,8.0f,10.0f,0.0f};
  glLightfv(GL_LIGHT1,GL_POSITION,light_position31);
glEnable(GL_COLOR_MATERIAL);

glFlush();

glutSwapBuffers();

}

Void menu(int id)
{
    Switch(id)
    {
     Case 1: glutIdleFunc(s);
                 Break;
     Case 2: glutIdleFunc(s);
                 Br
     eak;
     Case 3:
     exit(0);
                 Break;
     }

     glFlush();
     glutSwapBuffers();
     glutPostRedisplay()
     ;

}

Void mykey(unsigned char key,int x,int y)

 {

      if(key=='s')

      {
             glutIdleFunc(s);
       }

      if(key=='S')

      {
```

```
                glutIdleFunc(S);
             }

            if(key==’e’)
            {
                glutIdleFunc(e);
             }

            if(key==’t’)

            {
               transmit=!transmit;

               if(!transmit)
{

w1=0;w2=0;w3=0;

}

if(key==’q’|key==’Q’)
{
                    Exit(0);
              }
    }

     Void sat3(double ang)
    {
        glClear(GL_COLO
        R);
        glLoadIdentity();
        gltranslatef(0.0f,0.0
        f,);
        glRotatef(ang,0.0f,1
        .0f);


        //Earth
        glPushMatrix();
        glColor3f(0.3,0.6
        ,1);
        //glScaled(0.8,0.04,0.8);
        //glTranslatef(0.0,0.0,0.
        0);
        glutSolidSphere(2.0,50
        ); glPopMatrix();

          satellite();

       glFlush();
       glutSwapBuff
       ers();

    }

    Void e()
    {

          x-=0.07;
          sat2(x);
    }

    Void s()
    {
          x-=0.07;
          sat2(x);
    }

    Void S()
    {
          x+=.07;

          if(transmit)
          {
          if(w1<
          =4.2)
          w1+=0.
          01;

          if(w1>=2.5&&w2<
          =6.9)w2+=0.01;

          if(w1>=2.5&&w3
          <=5)w3+=0.01;
          }

          Sa

          t1

          ();

     }
```

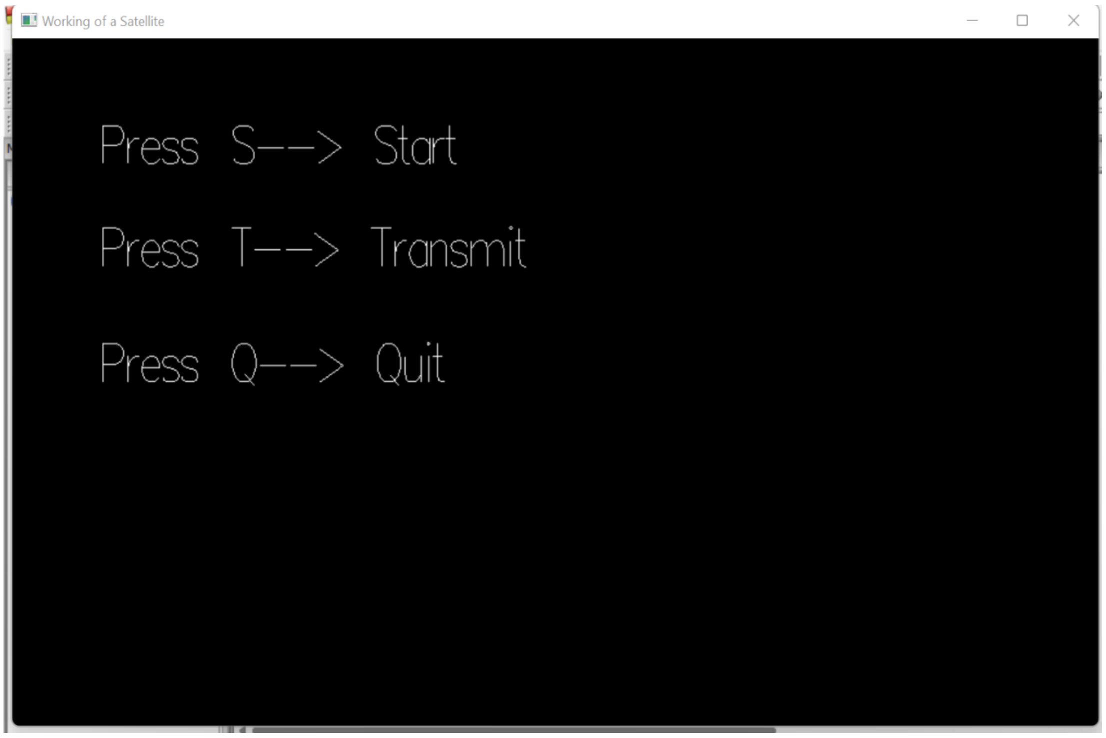
SCREENSHOTS

CHAPTER 5

SCREENSHOTS

FIRST VIEW:

This is the first scene which appears when the program is executed.



Working of a Satellite

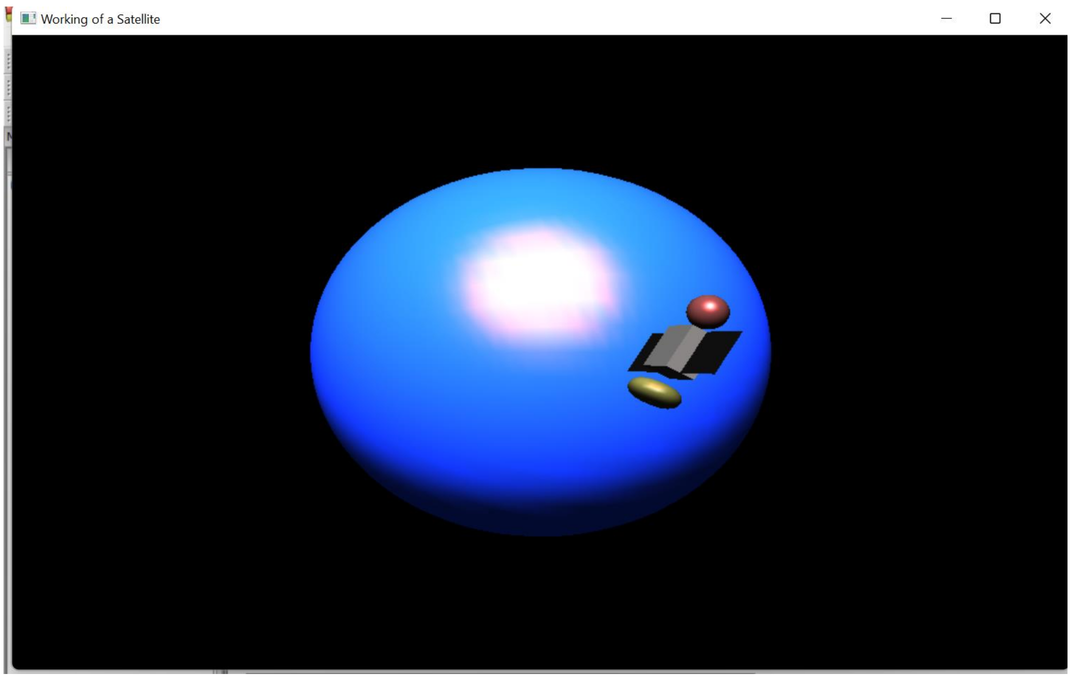
```
Press  S--> Start

Press  T--> Transmit

Press  Q--> Quit
```

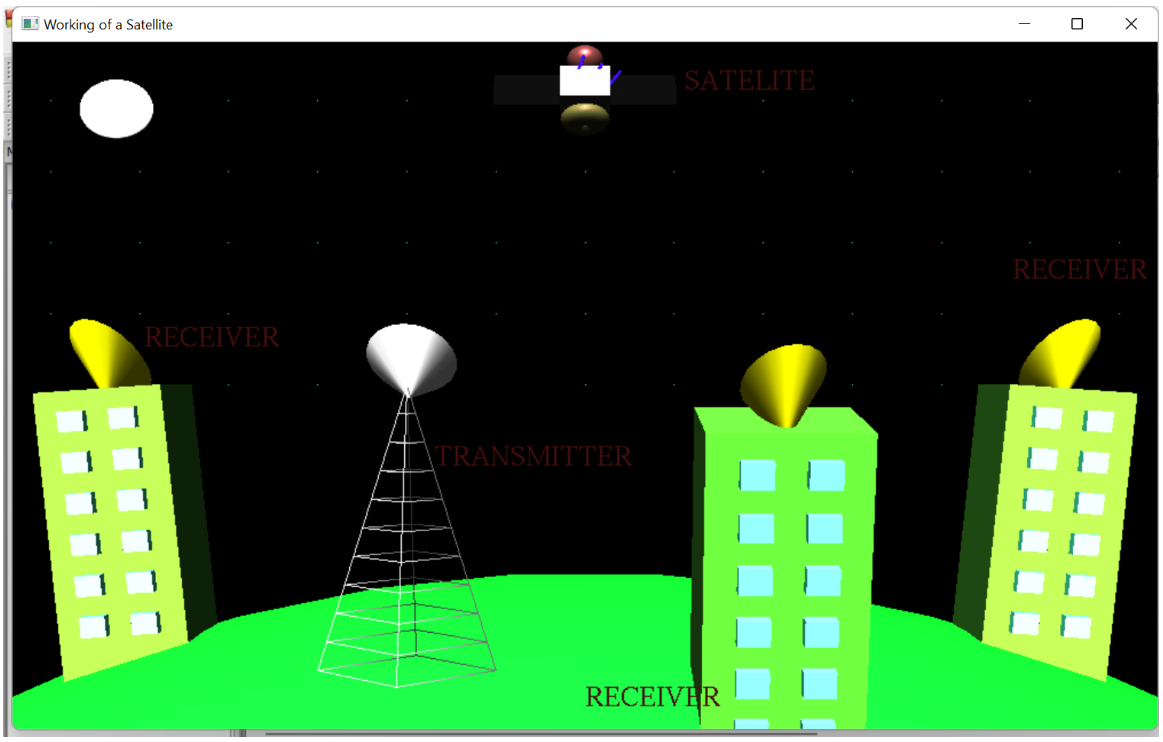
SECOND VIEW:

Satellite revolving around the planet earth.



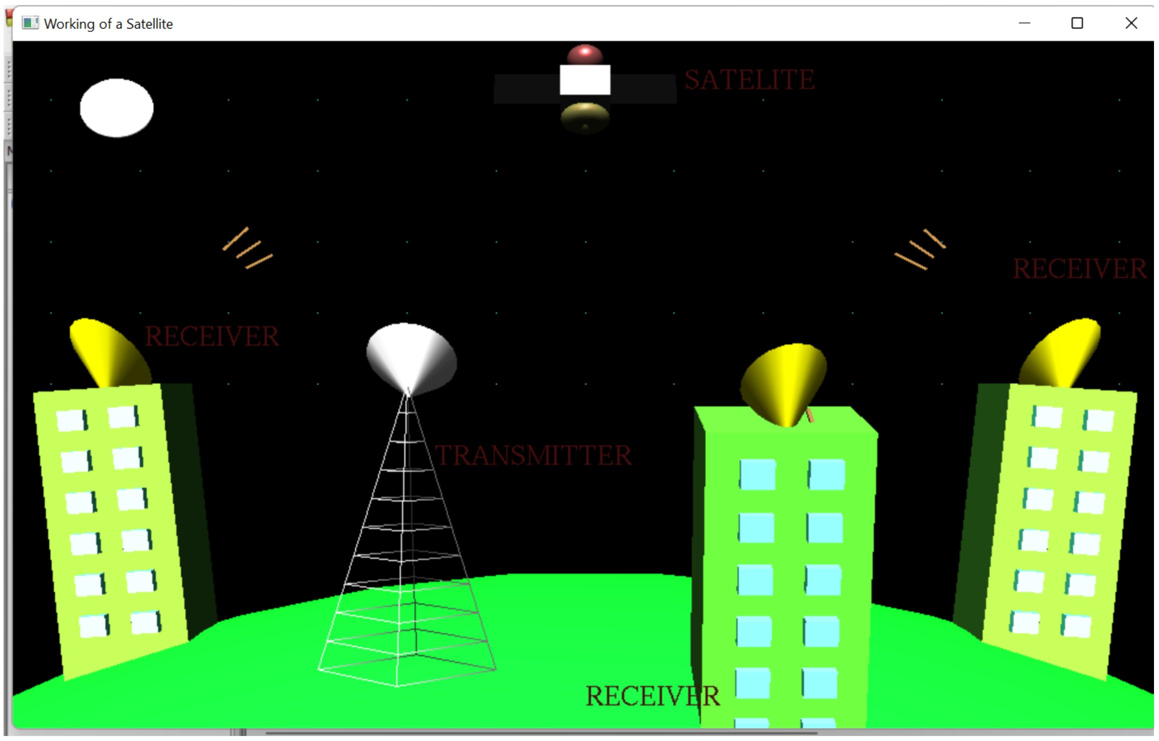
THIRD VIEW:

Data is transmitting through the transmitter.



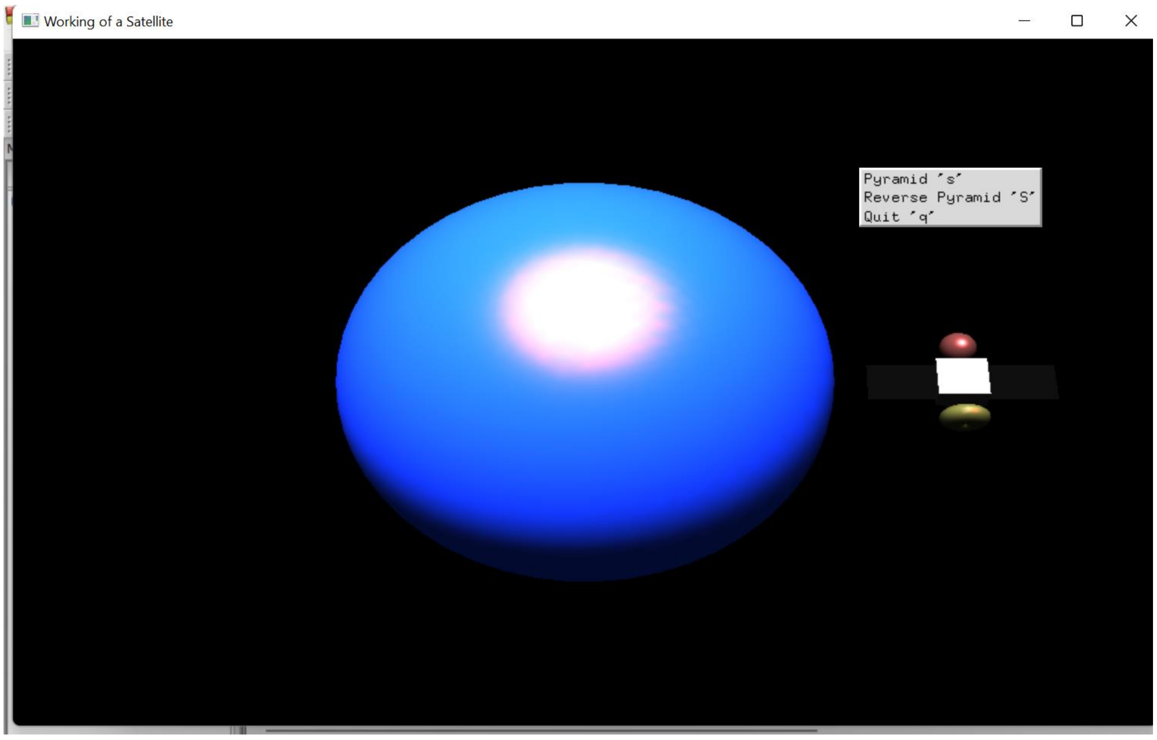
FOURTH VIEW:

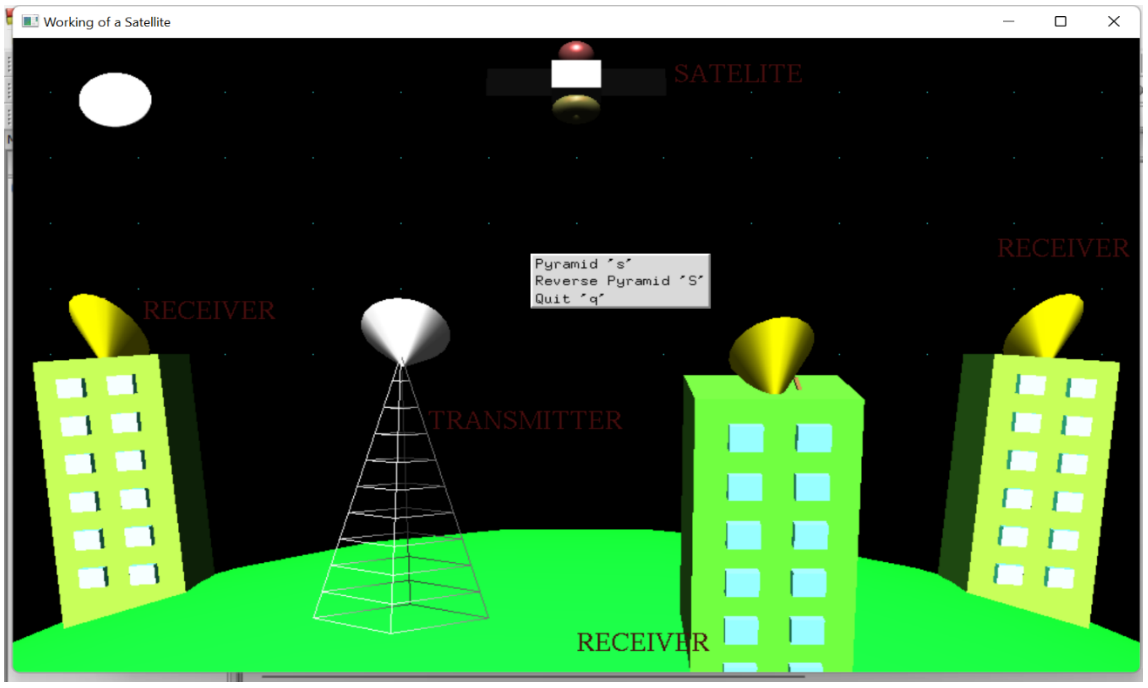
Data is receiving through the receivers.



FIFTH VIEW:

Key statements used as menu to visualize next or before output slide.





CONCLUSION

CHAPTER 6

CONCLUSION

The project “WORKING OF SATELLITE VISUALIZATION in OpenGL” was designed and implemented by us its creators as the exercise of “Computer Graphics and Visualization” Laboratory. More features to make it a competitive. Further work on projects like this would enable greater knowledge in OpenGL. An attempt has been made to develop a OpenGL graphics package, which meets the necessary requirements of the user successfully. This project has given us an insight into the use of Computer Graphics. As we have had to use many built-in and user defined functions, we have managed to get a certain degree of familiarity with these functions and have generally understood the power of these functions and were able to comprehend the true nature of the most powerful tool graphics in OpenGL and also have understood to a reasonable extent the reason why Graphics is so powerful for animation programmers. We can now converse with a certain degree of confidence about Graphics in OpenGL.

BIBLIOGRAPHY

CHAPTER 7

BIBLIOGRAPHY

Books References

1.Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd/4th Edition, Pearson Education, 2011.

2.Edward Angel: Interactive Computer Graphics- A Top-Down approach with OpenGL, 5th edition. Pearson Education, 2008.

WEBSITES

1. http://www.opengl-tutorial.org/
2. https://hackr.io/tutorials/learn-opengl
3. https://github.com/JoeyDeVries/LearnOpenGL
4. https://www.geeksforgeeks.org/getting-started-with-opengl/