CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO COMPUTER GRAPHICS

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention
of photography and television.

Applications of Computer Graphics

1. Display of information
2. Design
3. Simulation and animation
4. User interfaces

The Graphics Architecture

Graphics Architecture can be made up of seven components:

1. Display processors
2. Pipeline architectures
3. The graphics pipeline
4. Vertex processing
5. Clipping and primitive assembly
6. Rasterization
7. Fragment processing

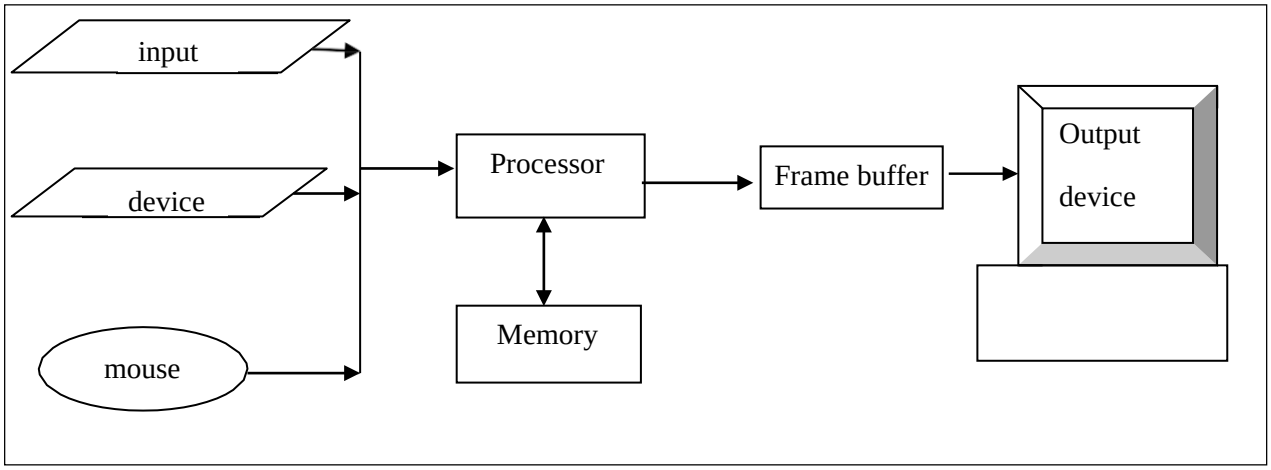

Figure 1.1: Components of Graphics Architecture and their working

1.2 INTRODUCTION TO OPENGL

OpenGL is software used to implement computer graphics. The structure of OpenGL is similar to that of most modern APIs including Java 3D and DirectX. OpenGL is easy to learn, compared with other.

APIs are nevertheless powerful. It supports the simple 2D and 3D programs. It also supports the advanced rendering techniques. OpenGL API explains following 3 components

1. Graphics functions
2. Graphics pipeline and state machines
3. The OpenGL interfaces

There are so many polygon types in OpenGL like triangles, quadrilaterals, strips and fans. There are 2 control functions, which will explain OpenGL through,

1. Interaction with window system
2. Aspect ratio and view ports

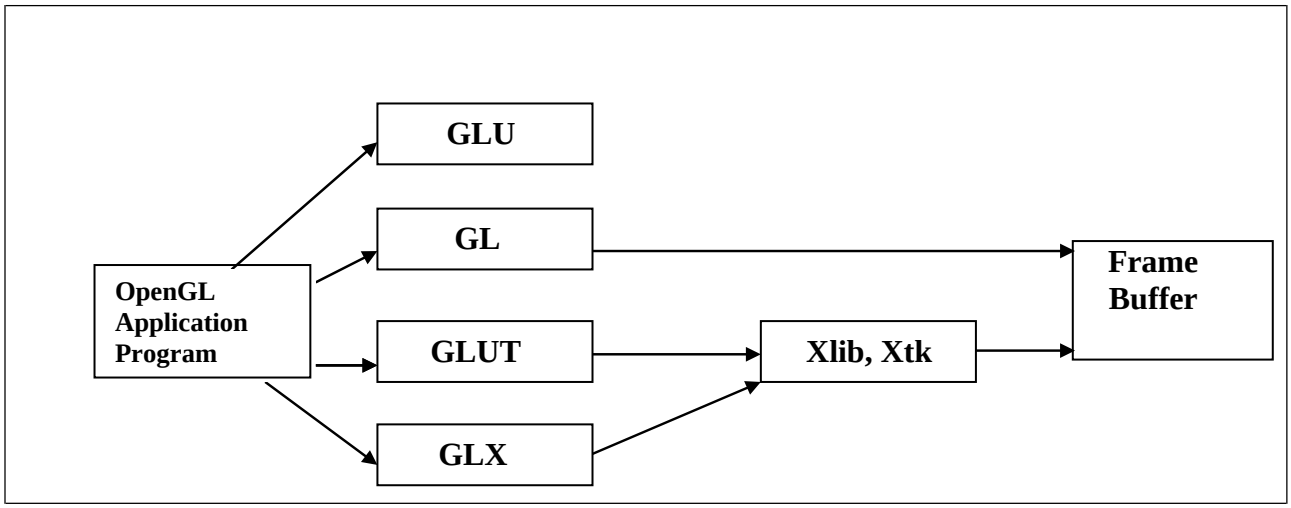

Figure 1.2: OpenGL Library organization

Most implementations of OpenGL have a similar order of operations, a series of processing stages called the OpenGL rendering pipeline. This ordering, as shown in Figure 1.2, is not a strict rule of how OpenGL is implemented but provides a reliable guide for predicting what OpenGL will do. The following diagram shows the assembly line approach, which OpenGL takes to process data. Geometric data (vertices, lines, and polygons) follow the path through the row of boxes that includes evaluators and per-vertex operations, while pixel data (pixels, images, and bitmaps) are treated differently for part of the process. Both types of data undergo the same final steps before the final pixel data is written into the frame buffer.

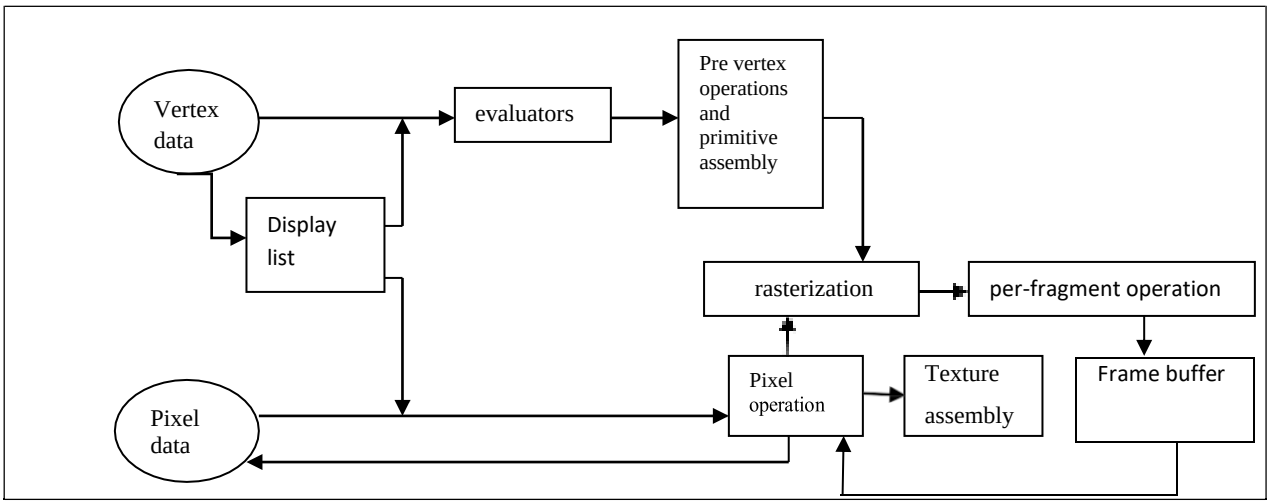

Figure 1.3: OpenGL Order of Operations

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

- Operating system – Windows 10
- Visual Studio
- OPENGL library files – GL, GLU, GLUT
- Language used is C/C++

2.2 HARDWARE REQUIREMENTS

- Intel®Pentium or AMD
- Memory – 1GB RAM
- Mouse or other pointing device
- QWERTY Keyboard
- Display device

CHAPTER 3

SYSTEM DEFINITION

3.1 PROJECT DESCRIPTION

OpenGL is software which provides a graphical interface. It is an interface between the application program and the graphics hardware.

Simulation of Farming involves implementation of several stages of farming such as ploughing of the field, sowing of seeding, irrigation and finally harvesting the grown crops. Ploughing is the act of tilling the land before sowing of seeds. Sowing is the process of planting seeds. Irrigation is the application of controlled amounts of water to plants. Harvesting is the process of gathering the crops from the fields. The program is menu driven which provides the simulations for the different stages of farming.

3.2 USER DEFINED FUNCTIONS

- **void basicWindow() :** This function is used to create the basic view of the farm.
- **void ploughField() :** This function is used to plough the field.
- **void seedingField() :** This function is used to sow the seeds on the field.
- **void wateringField() :** This is used to irrigate the field.
- **void sapling() :** This function is used to generate the saplings in the field.
- **void plantField() :** This function is uses the plant() function to generate the plants in the field.

3.3 DATA FLOW DIAGRAM

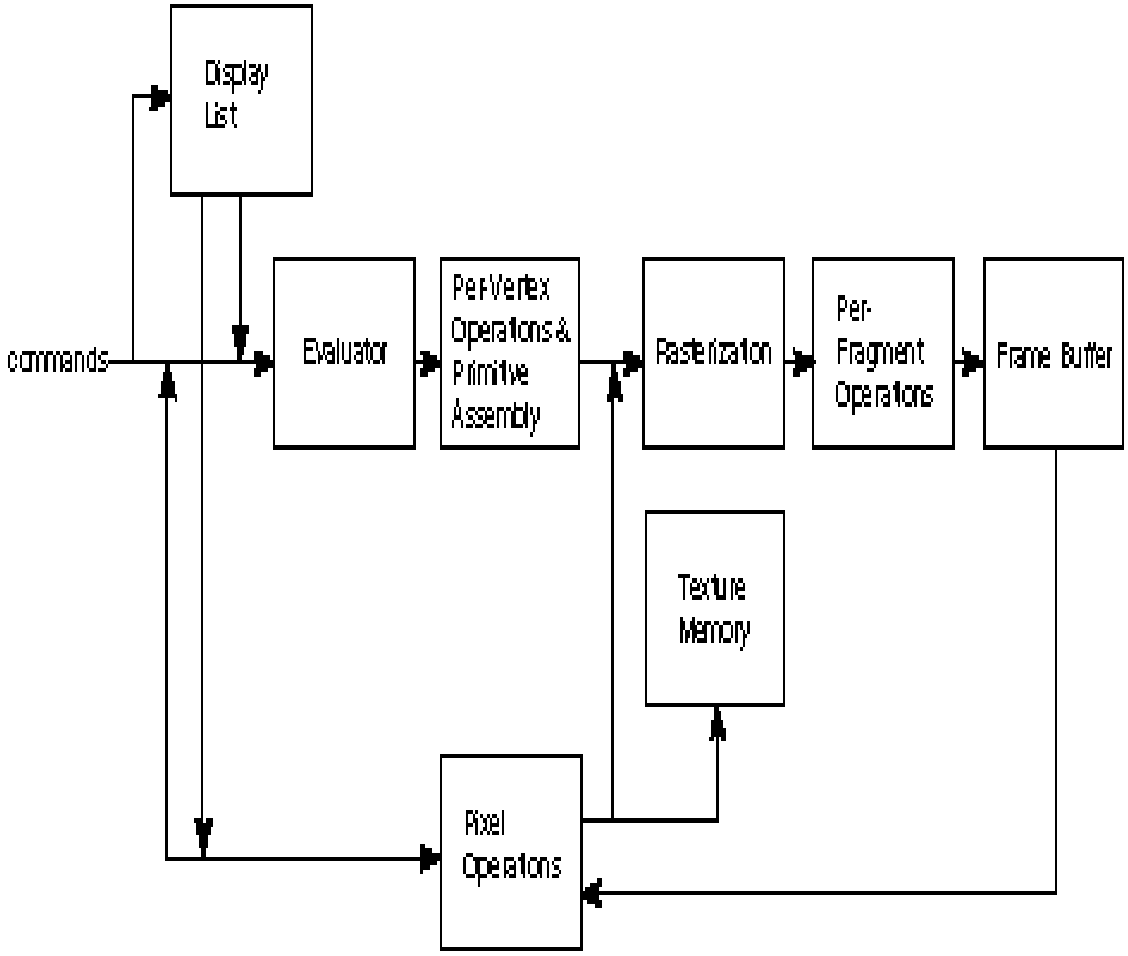

Figure 3.3: Data Flow Diagra

CHAPTER 4

IMPLEMENTATION

4.1 SOURCE CODE

```
#include<GL/glut.h>
#include<string.h>
int i,flag=0,flagb=1,flags=0,flagt=0,flagp=0,flagw=1,flagx=0;
float
a=0.0f,b=0.0f,c=0.0f,m=0.0f,n=0.0f,o=0.0f,p=0.0f,q=0.0f,r=0.0f,x=0.0f,y=0.0f,z=0.0f,a1=0.0,a2=0.0,a3
=0.0;
float j;
void *currentfont;
void setFont(void *font)
{
    currentfont=font;
}
void drawstring(char string[],float x1,float y1,float z1)
{
   int i,j;
    j=strlen(string);
    glRasterPos3f(x1,y1,z1);
    for (i=0;i<j;i++)
    {
      glutBitmapCharacter(currentfont, string[i]);
    }
}
void screen1()
{
   glClearColor(0.0,0.0,0.0,0.0);
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   setFont(GLUT_BITMAP_TIMES_ROMAN_24);
   glColor3f(1.0,1.0,1.0);
   char str1[]="GLOBAL ACADEMY OF TECHNOLOGY";
   drawstring(str1,-0.5,0.9,0.0);
   glColor3f(0.9,0.9,0.9);
   char str2[]="DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING";
   drawstring(str2,-0.7,0.8,0.0);
   glColor3f(0.8,0.8,0.8);
   char str3[]="A MINI PROJECT ON";
   drawstring(str3,-0.3,0.7,0.0);
   glColor3f(1.0,0.7,0.7);
   char str4[]="-- VERTICAL LIFT BRIDGE SIMULATION --";
   drawstring(str4,-0.65,0.6,0.0);
```

```
    glColor3f(0.2,0.8,0.5);
    char str5[]="BY  :";
    drawstring(str5,-0.5,0.5,0.0);
    glColor3f(1.0,1.0,0.0);
    char str6[]="J  M  CHINMAI  JYOTHY     (1GA19CS063)";
    drawstring(str6,-0.5,0.4,0.0);
    char str8[]="GUIDES  :";
    drawstring(str8,-0.5,0.1,0.0);
    glColor3f(1.0,1.0,0.0);
    char str9[]="Mr . KRISHNA PRASAD    (Asst.prof. Dept of CSE)";
    drawstring(str9,-0.5,0.0,0.0);
    char str10[]="Mrs . SUSHMITHA   (Asst.prof. Dept of CSE)";
    drawstring(str10,-0.5,-0.1,0.0);
    glColor3f(1.0,0.9,1.0);
    char str11[]="INSTRUCTIONS ";
    drawstring(str11,-0.3,-0.5,0.0);
    char str12[]="-----------------";
    drawstring(str12,-0.35,-0.55,0.0);
    char str13[]=" * * *   PRESS     -ENTER -     TO  START  AND     -ESC-     TO EXIT   * * *";
    drawstring(str13,-0.9,-0.7,0.0);
    char str14[]="PRESS     -S-     TO  START  ANIMATION   AND  PRESS    -T-     TO  STOP
ANIMATION";
    drawstring(str14,-1.05,-0.85,0.0);
    glFlush();
}
void screen3()
{
    glClearColor(0.0,0.0,0.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    setFont(GLUT_BITMAP_TIMES_ROMAN_24);
    glColor3f(0.0,0.0,0.0);
    char str1[]=" * * *  THANK YOU  * * * ";
    drawstring(str1,-0.2,0.2,0.0);
    char str2[]=" ------------------------ ";
    drawstring(str2,-0.3,0.1,0.0);
    glFlush();
}
void water()
{
    glBegin(GL_QUADS);
    glColor3f(0.0,0.4,0.7);
    glVertex3f(-5.0,-0.415,5.0);
    glVertex3f(5.0,-0.415,5.0);
    glVertex3f(5.0,-0.415,-5.0);
    glVertex3f(-5.0,-0.415,-5.0);
    glEnd();
}
```

```
void lines()
{
   float t1,t2;
   glBegin(GL_LINES);
   for(t1=0.0;t1<=10.0;t1+=0.4)
   {
    for(t2=0.0;t2<=10.0;t2+=0.4)
    {
      glColor3f(0.7,0.7,0.7);
      glVertex3f(-5.0+t2,-0.41,-4.5+t1);
      glVertex3f(-4.95+t2,-0.41,-4.5+t1);
    }
   }
   glEnd();
}
void new1()
{
   glTranslatef(a,b,c);
   bridge();
}
void new2()
{
   glTranslatef(m,n,o);
   ship();
}
void new3()
{
   glTranslatef(p,q,r);
   train();
}
void new4()
{
   glTranslatef(x,y,z);
   aero();
}
void new5()
{
   glTranslatef(a1,a2,a3);
   lines();
}

void update(int value)
{
   if(flagx==1)
   {
    if(flagb==1)
    {
```

```
        b+=0.02f;
        if(b>0.5)
        {
            flagb=2;
            flags=1;
        }
      }
      if(flags==1)
      {
        o+=0.07f;
        if(o>2.0)
            flagp=1;
        if(o>6.0)
        {
            flagb=0;
        }
      }
      if(flagb==0)
      {
        b-=0.02f;
        if(b<0.01)
        {
            flagb=1;
            flagt=1;
        }
      }
      if(flagt==1)
      {
          p-=0.05f;
      }
      if(flagp==1)
      {
        x+=0.03;
      }
       if(flagw==1)
       {
        a1+=0.006;
       }
     }
    glutPostRedisplay();
    glutTimerFunc(100,update,0);
}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glClearColor(0.0,0.5,1.0,0.0);
    glPushMatrix();
```

```
glRotatef(20.0,0.25,0.5,0.0);
base();
glPopMatrix();

glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
pillars();
glPopMatrix();
glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
earth();
glPopMatrix();
glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
track();
glPopMatrix();
glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
new1();
glPopMatrix();
glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
new2();
glPopMatrix();
glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
new3();
glPopMatrix();
glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
new4();
glPopMatrix();
glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
water();
glPopMatrix();
glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
new5();
glPopMatrix();
glPushMatrix();
glColor3f(1.0,1.0,0.0);
glTranslatef(1.2,0.9,-5.1);
glutSolidSphere(0.08,20,20);
glPopMatrix();
glPushMatrix();
glRotatef(20.0,0.25,0.5,0.0);
```

```
    lighthouse();
    glTranslatef(0.28,-0.05,-5.0);
    glColor3f(0.0,0.0,0.0);
    glutSolidSphere(0.02,20,20);
    glPopMatrix();

    glPushMatrix();
    glRotatef(20.0,0.25,0.5,0.0);
    signal();
    glPopMatrix();
    glPushMatrix();
    glRotatef(20.0,0.25,0.5,0.0);
    light();
    glPopMatrix();
    glFlush();
}
void mydisplay()
{
    if(flag==0)
       screen1();
     if(flag==1)
      display();
    if(p<-6.0)
        screen3();
    if(p<-6.8)
      exit(0);
}
void mykeyboard(unsigned char key,int x,int y)
{
    switch(key)
    {
        case 13 :flag=1;
                 break;
        case 83 :if(flag==1)
                    flagx=1;
                 break;
        case 115:if(flag==1)
                    flagx=1;
                 break;
        case 84 :flagx=0;
                 break;
        case 116 :flagx=0;
                 break;
        case 27:exit(0);
    }
}
void reshape(int w,int h)
```

```
    {
       glViewport(0,0,w,h);
       glMatrixMode(GL_PROJECTION);
       glLoadIdentity();
       if(w<=h)
          glOrtho(-1.1,1.1,1.1*(GLfloat)h/(GLfloat)w,1.1*(GLfloat)h/(GLfloat)w,-10.0,10.0);
       else
          glOrtho(-1.1*(GLfloat)w/(GLfloat)h,1.1*(GLfloat)w/(GLfloat)h,-1.1,1.1,-10.0,10.0);
       glMatrixMode(GL_MODELVIEW);
       glLoadIdentity();
    }

    int main(int argc,char **argv)
    {
       glutInit(&argc,argv);
       glutInitDisplayMode(GLUT_SINGLE| GLUT_RGB | GLUT_DEPTH);
       glutInitWindowSize(1500,1000);
       glutCreateWindow("VLBS");
       glClearColor(0.0,0.0,0.0,0.0);
       glEnable(GL_DEPTH_TEST);
       glutReshapeFunc(reshape);
       glutDisplayFunc(mydisplay);
       glutKeyboardFunc(mykeyboard);
       glutTimerFunc(200,update,0);
       glutMainLoop();
       return 0;
}
```

CHAPTER 5

TESTING AND RESULTS

5.1 DIFFERENT TYPES OF TESTING

1. Unit Testing

Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components.

2. Module Testing

A module is a collection of dependent components such as a object class, an abstract Data type or some looser collection of procedures and functions. A module related Components, so can be tested without other system modules.

3. System Testing

This is concerned with finding errors that result from unanticipated interaction between Sub-system interface problems.

4. Acceptance Testing

The system is tested with data supplied by the system customer rather than simulated test data.

5.2 TEST CASES

The test cases provided here test the most important features of the project.

Table 5.2.1: Test Case

Sl No	Test Input	Expected Results	Observed Results	Remarks
1	Click Enter	Start the simulator	Simulator started	Pass
2	Click S	Start the movement	Movement started	Pass
3	Click T	Terminate/Pause the moment	Movement paused	Pass
4	Click ESC	Stops the simulator	Simulator syopped	Pass

CHAPTER 6

SNAPSHOTS

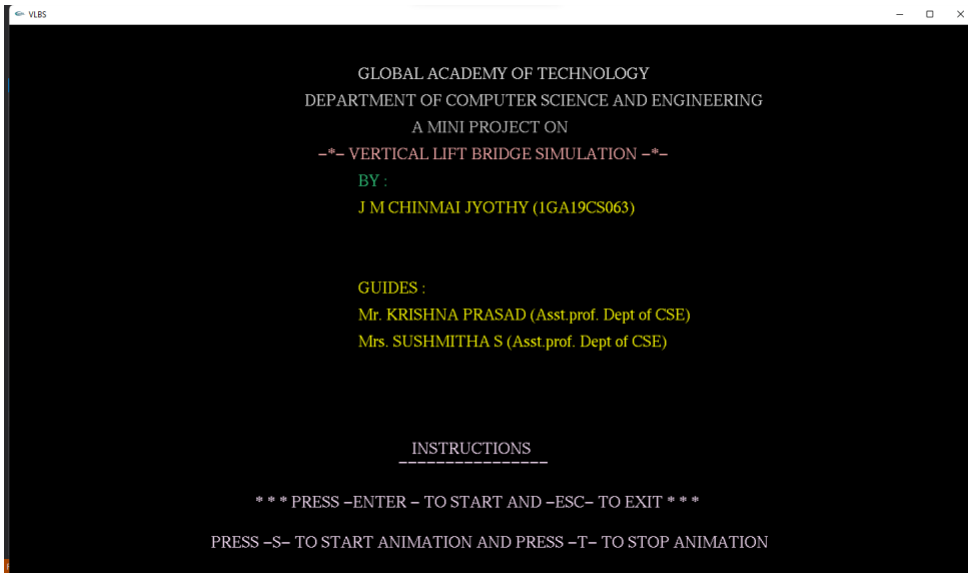

Figure 6.1: WELCOME SCREEN

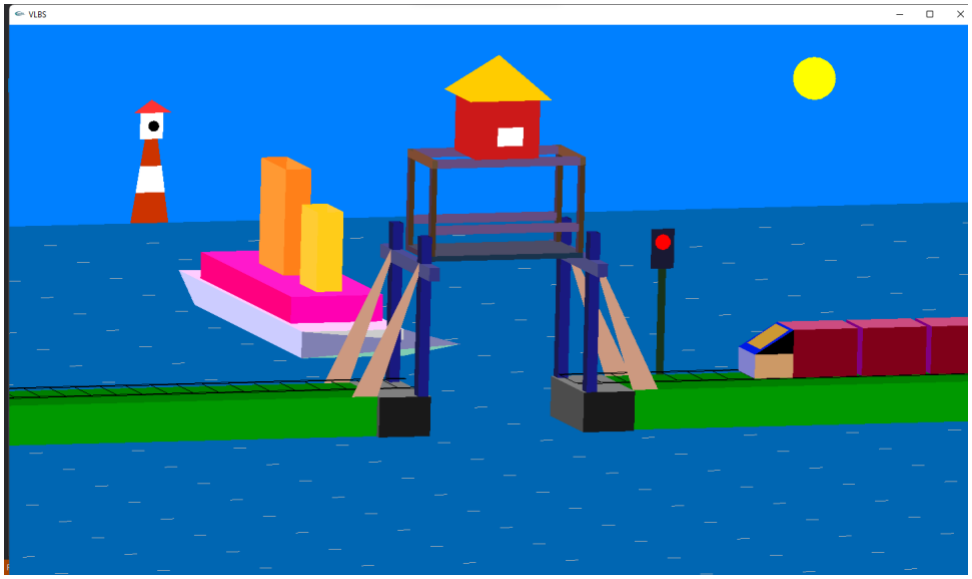

Figure 6.2: SIGNAL RED AND LIFT OF THE BRIDGE

Figure 6.3: SHIP CROSSING THE BRIDGE

Figure 6.4: SIGNAL GREEN TRAIN CROSSING BRIDGE

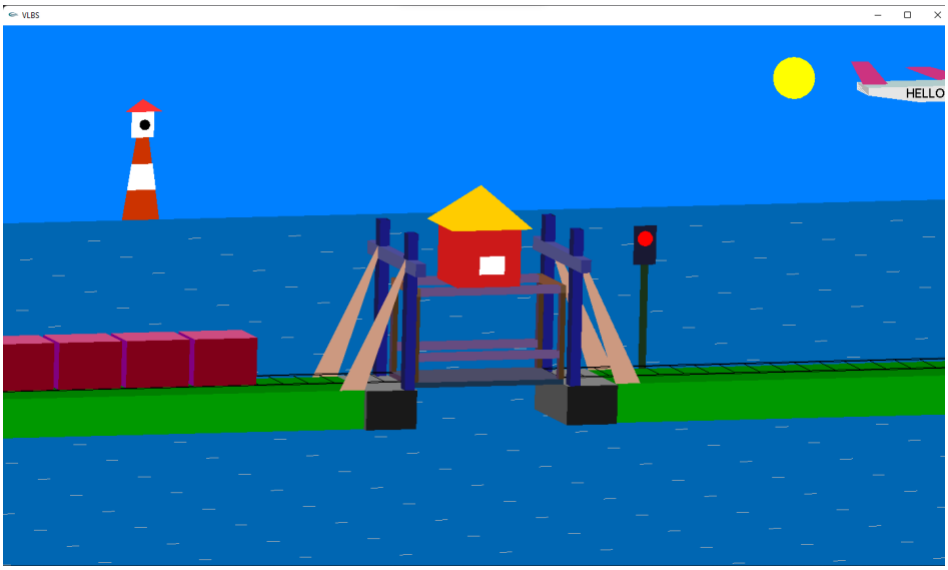

Figure 6.5: TRAIN CROSSING THE BRIDGE

Figure 6.6: EXIT SCREEN

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

Designing and implementing project in graphics is a great experience. We understood and analyzed about the concepts of OpenGL which is very useful for our future.

"VERTICAL LIFT BRIDGE SIMULATION" is developed to provide a GUI. An attempt has been made to develop an openGL package which meets necessary requirements of the user.

The development of the mini project has given us a good exposure to openGL by which we have learnt some of the techniques which help in the development of interactive application

5.2 FUTURE SCOPE

This application is like open source where anyone can design and add his own codes to modify. Even more features can be included.

BIBLIOGRAPHY

TEXT BOOKS:

[1]. Edward Angel, Interactive Computer Graphics A Top-Down Approach Using OpenGL,Pearson Education Asia,Fifth Edition,2008.

WEBSITE:

[1]. www.OpenGL.org

[2]. www.Github.com

[3]. www.stackoverflow.com

[4]. www.youtube.com

[5]. www.openglprojects.in