VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi – 590 018

A

Mini Project Report

on

"3D HOUSE"

Submitted in partial fulfillment of Computer Graphics Laboratory with Mini project 18CSL67 in Computer Science and Engineering for the Academic Year 2020-2021

Submitted by

PARISMITA DEVI
1GA18CS103

Under the Guidance of

Mrs. Reshma S
Assistant Professor

GLOBAL ACADEMY OF TECHNOLOGY

Department of Computer Science and Engineering

Rajarajeshwarinagar, Bengaluru - 560 098

2020 – 2021

GLOBAL ACADEMY OF TECHNOLOGY

Department of Computer Science and Engineering

CERTIFICATE

Certified that the V1 Semester Mini Project in Computer Graphics Laboratory with Mini project Entitled **"3D House"** carried out by **Ms. Parismita Devi** , bearing **USN 1GA18CS103** is submitted in partial fulfillment for the award of the **Bachelor of Engineering** in Computer Science and Engineering from **Visvesvaraya Technological University, Belagavi** during the year 2020-2021.The Computer Graphics with Mini project report has been approved as it satisfies the academic requirements in respect of the mini project work prescribed for the said degree.

Mrs. Reshma S Assistant Professor, Dept of CSE, GAT, Bengaluru.	**Dr. Bhagyashri R Hanji** Professor & Head, Dept of CSE, GAT, Bengaluru.

Name of the Examiner	Signature with date
1. _____	_____
2. _____	_____

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance crowned our efforts with success.

I consider myself proud, to be part of **Global Academy of Technology** family, the institution which stood by our way in endeavors.

I express my deep and sincere thanks to our principal **Dr. N. Rana Pratap Reddy** for his support.

I would be grateful to **Dr. Bhagyashri R Hanji,** Professor and HOD, Dept Of CSE who is source of inspiration and of invaluable help in channelizing my efforts in right direction.

I wish to thank my internal guide **Prof. Reshma S**, Assistant Professor, Dept of CSE for guiding and correcting various documents of mine with attention and care. They have taken lot of pain to go through the document and make necessary corrections as and when needed.

I would like to thank the faculty members and supporting staff of the Department of CSE, GAT for providing all the support for completing the Project work.

Finally, I am grateful to my parents and friends for their unconditional support and help during the course of my Project work.

PARISMITA DEVI

1GA18CS103

ABSTRACT

In this project, I will draw the 3D Home. We are using the OpenGL library to implement the project. OpenGL has a lot of inbuilt functions which makes the drawing of any geometric objects quite easy. Here the display of a complete 3D view of the house along with the view of the interior of the house will be presented. I used different OpenGL functions and commands to display the various objects inside the house. I used these methods to create a realistic view of the interior rooms as well as the exterior view. I made the use of the menu function to switch between different views inside and outside the house.

TABLE OF CONTENTS

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

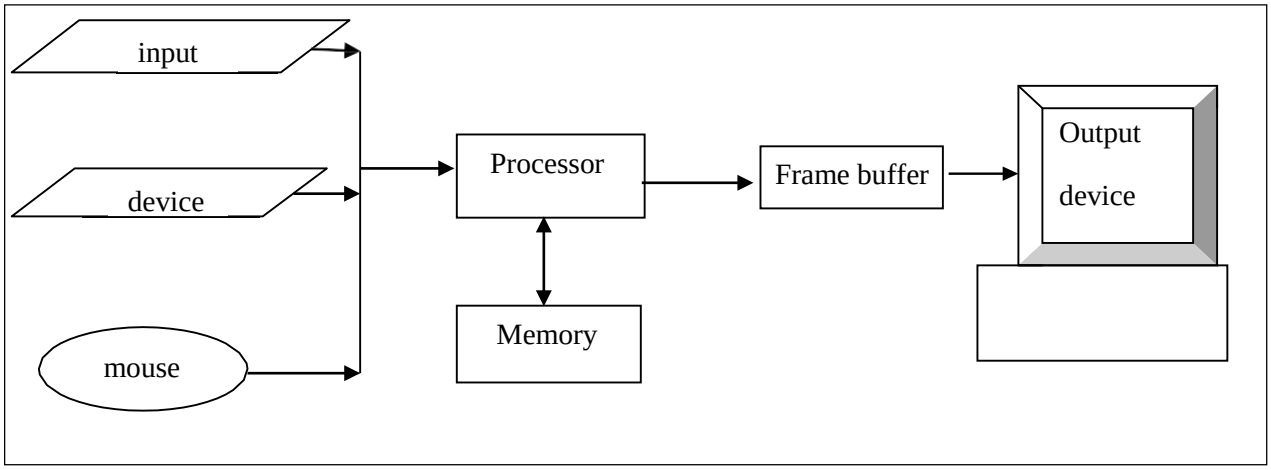


Figure 1.1: Components of Graphics Architecture and their working

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

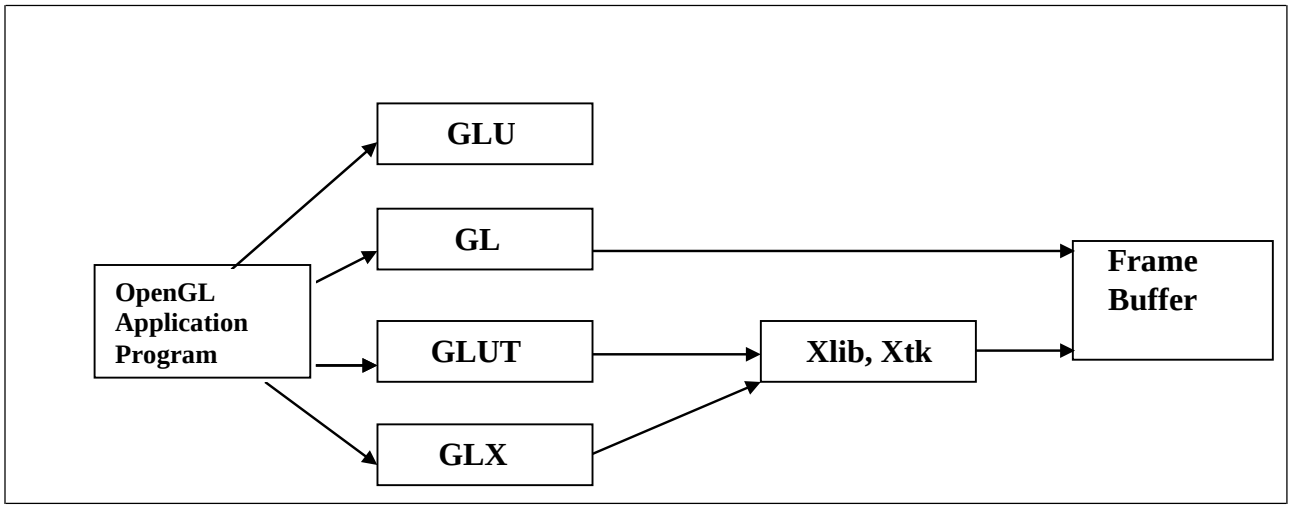


Figure 1.2: OpenGL Library organization

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.

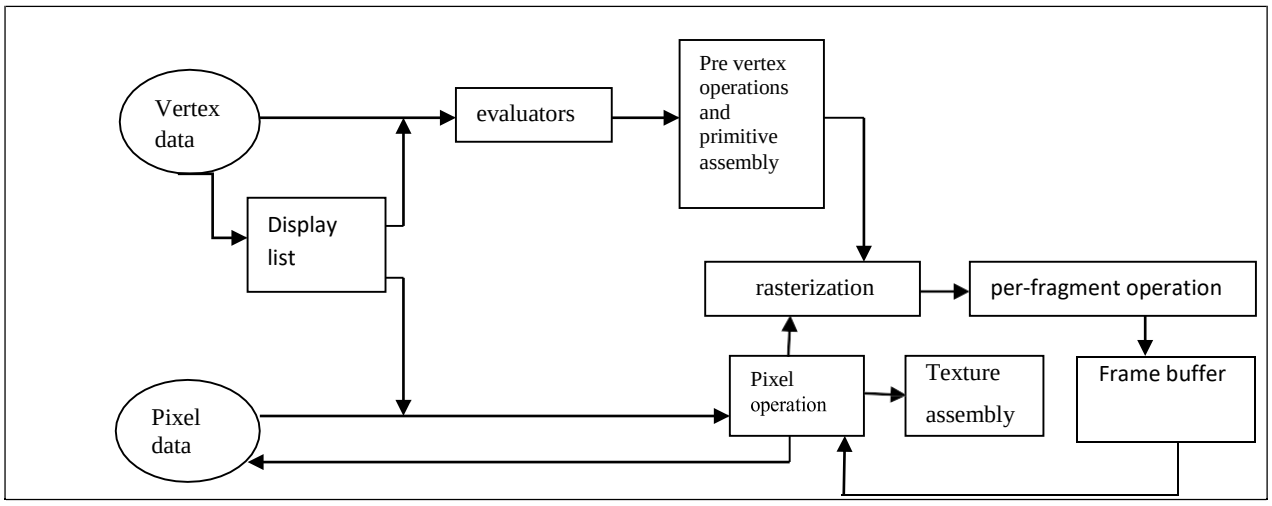


Figure 1.3: OpenGL Order of Operations

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

Minimum software specification

- Operating system: Windows 10
- Tool Used: Visual Studio 2019
- OPENGL Library
- X86
- X64(WOW)
- Mouse Driver
- Graphics Driver
- C Language

2.2 HARDWARE REQUIREMENTS

Minimum hardware specification

- Microprocessor: 1.0 GHz and above CPU based on either AMD or INTEL Microprocessor Architecture
- Main memory: 512 MB RAM
- Free Space: 40 GB
- Keyboard: QWERTY Keyboard
- Mouse :2 or 3 Button mouse
- Monitor: 1024 x 768 display resolution

CHAPTER 3

SYSTEM DEFINITION

3.1 PROJECT DESCRIPTION

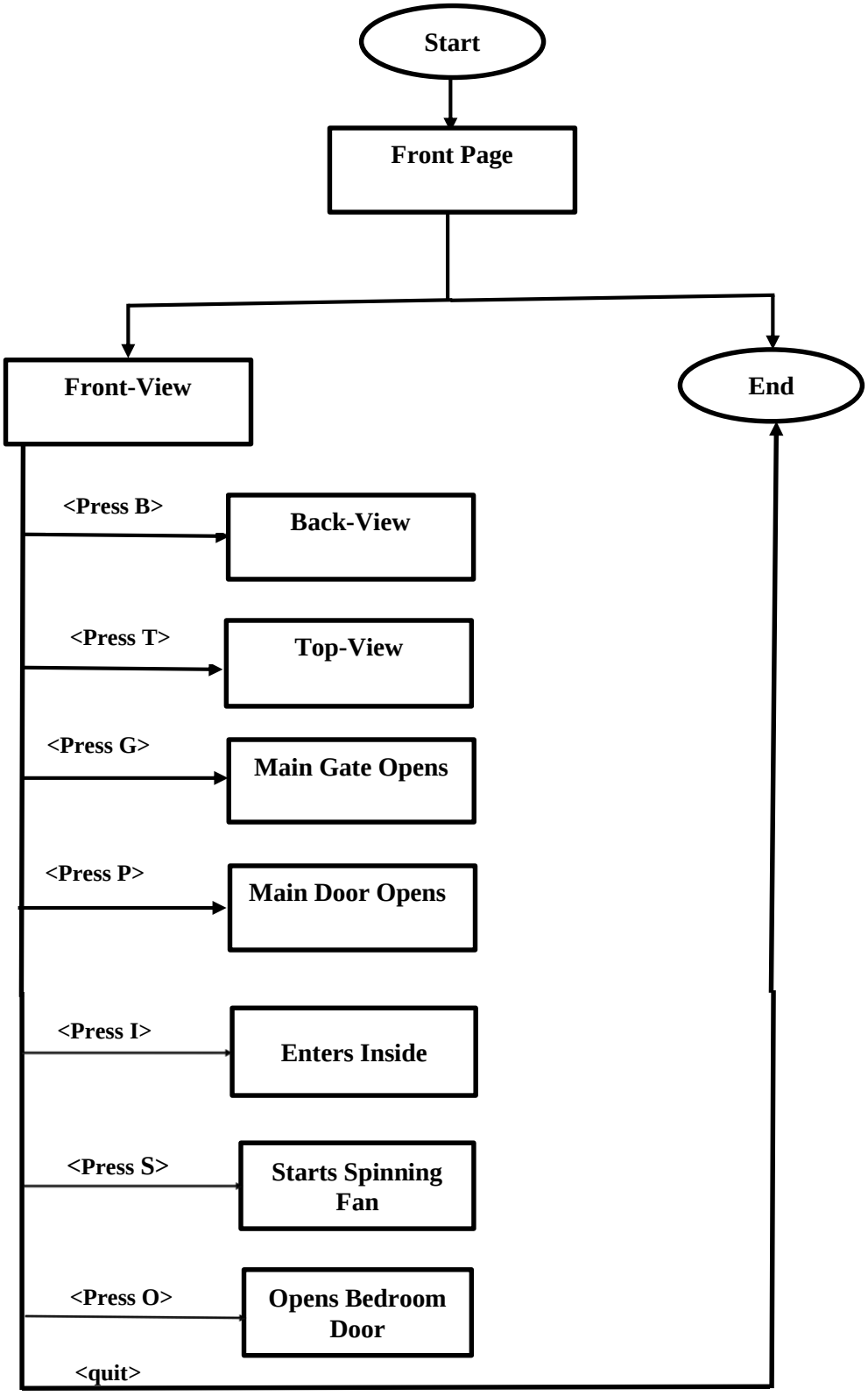
OpenGL is software which provides a graphical interface. It is an interface between the application program and the graphics hardware.

3D Home Architect is a property designing program. Harneet's guide to 3D Home Architect comes in three designs for specific purposes: Home and Landscape Design Suite, Home Design Deluxe, and Landscape Design Deluxe. Home Design Deluxe simulates home designs, Landscape Design Deluxe simulates landscape designs, and Home and Landscape Design Suite is used for both.

3.2 USER DEFINED FUNCTIONS

- **void wall() :** This function is used to create house wall.
- **void wall2() :** This function is used to create compound wall.
- **void earth() :** This function is used to create ground.
- **void compound() :** This function is used to create complete boundary compound wall.
- **void terece() :** This function is used to create the terrace border.
- **void fanwing() :** This function is used to create the fan blades.
- **void fanbottom() :** This function is used to create the center circle of fan.
- **void fan() :** This function is used to create complete fan.
- **void myclock() :** This is used to create a wall clock.
- **void gate() :** This is used to create the main gate.
- **void house() :** This function is used to create the complete house.

3.3 DATA FLOW DIAGRAM



CHAPTER 4

IMPLEMENTATION

4.1 SOURCE CODE

```
#define _CRT_SECURE_NO_WARNINGS
#include<windows.h>
#include<glut.h>
#include <stdio.h>
#include <time.h>
double view[3] = { 2,2,12.9 };
double look[3] = { 2,2,2 };
int flag = -1;
void steps(void);
void window(void);
void gate(void);
double angle = 0, speed = 5, maino = 0, tro = 0, romo = 0, mgo = 0;
//declarating quadric objects
GLUquadricObj* Cylinder;
GLUquadricObj* Disk;

struct tm* newtime;
time_t ltime;

GLfloat angle1;

//initialisation
void myinit(void)
{
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum(-1.0, 1.0, -1 * 1200 / 600, 1 * 1200 / 600, 1, 200.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  //defining new quadric object
  Cylinder = gluNewQuadric();
  //to set drawing style
  gluQuadricDrawStyle(Cylinder, GLU_FILL);
  //to set automatic normals
  gluQuadricNormals(Cylinder, GLU_SMOOTH);


  Disk = gluNewQuadric();
```

```
   gluQuadricDrawStyle(Disk, GLU_FILL);
   gluQuadricNormals(Disk, GLU_SMOOTH);
   GLfloat gam[] = { 0.2,.2,.2,1 };
   glLightModelfv(GL_LIGHT_MODEL_AMBIENT, gam);

}

//set material property
void matprop(GLfloat amb[], GLfloat dif[], GLfloat spec[], GLfloat shi[])
{
   glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, amb);
   glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, dif);
   glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, spec);
   glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, shi);
}

//to create wall
void wall(double thickness)
{
   glPushMatrix();
   glTranslated(2, .5 * thickness, 2);
   glScaled(4.0, thickness, 4.0);
   glutSolidCube(1.0);
   glPopMatrix();
}
//to create compound wall
void wall2(double thickness)
{
   glPushMatrix();
   glTranslated(.8, .5 * thickness * 4, 3.5);
   glScaled(1.6, thickness * 4, 7.0);
   glutSolidCube(1.0);
   glPopMatrix();
}
//to create earth
void earth(void)
{
   GLfloat ambient[] = { 1,0,0,1 };
   GLfloat specular[] = { 0,1,1,1 };
   GLfloat diffuse[] = { .2,.9,.5,1 };
   GLfloat shininess[] = { 50 };


   matprop(ambient, diffuse, specular, shininess);
```

```
   GLfloat lightIntensity[] = { .7,.7,.7,1 };
   GLfloat light_position[] = { 2,5,-3,0 };
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);
   glLightfv(GL_LIGHT0, GL_DIFFUSE, lightIntensity);

   glPushMatrix();
   glTranslated(0, -.25, 0);
   glScaled(10000, .5, 1000000);
   glutSolidCube(1.0);
   glPopMatrix();
   glFlush();
}

void compound(void)      //carpaser gher
{

   GLfloat ambient[] = { 1,0,0,1 };
   GLfloat specular[] = { 0,1,1,1 };
   GLfloat diffuse[] = { .7,.7,.7,1 };
   GLfloat shininess[] = { 50 };


   matprop(ambient, diffuse, specular, shininess);
   GLfloat lightIntensity[] = { .7,.7,.8,1 };
   GLfloat light_position[] = { 2,6,1.5,0 };
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);
   glLightfv(GL_LIGHT0, GL_DIFFUSE, lightIntensity);

   //left wall of compound
   glPushMatrix();
   glPushMatrix();
   glTranslated(-4, 0, -1 - .04);
   glRotated(90.0, 0, 0, 1);
   wall2(0.08);
   glPopMatrix();
   //right wall of compound
   glPushMatrix();
   glTranslated(8, 0, -1 - .02);
   glRotated(90.0, 0, 0, 1);
   wall2(0.08);

   glPopMatrix();
   //back wall of compound
   glPushMatrix();
```

```
    glTranslated(2, .8, -1);
    glRotated(-90, 1, 0, 0);
    glScaled(12, .02 * 4, 1.6);
    glutSolidCube(1.0);
    glPopMatrix();
    //front left wall of compound
    glPushMatrix();
    glTranslated(-3, .8, 6 - .08);
    glRotated(-90, 1, 0, 0);
    glScaled(2, .02 * 4, 1.6);
    glutSolidCube(1.0);
    glPopMatrix();
    //front right wall of compound
    glPushMatrix();
    glTranslated(3.6, .8, 6 - .08);
    glRotated(-90, 1, 0, 0);
    glScaled(8.2, .02 * 4, 1.6);
    glutSolidCube(1.0);
    glPopMatrix();

    glPopMatrix();

    GLfloat ambient2[] = { 0,1,0,1 };
    GLfloat specular2[] = { 1,1,1,1 };
    GLfloat diffuse2[] = { .2,.6,0.1,1 };
    GLfloat shininess2[] = { 50 };
    matprop(ambient2, diffuse2, specular2, shininess2);

    //floor
    glPushMatrix();
    glTranslated(-4, -0.05, -1);
    glScaled(3, 3, 1.7);
    wall(0.08);
    glPopMatrix();

    gate();
    glFlush();

}

void terece(void)  //chader boder
{

    GLfloat     ambient1[] = { 1,0,1,1 };
```

```
GLfloat specular1[] = { 1,1,1,1 };
GLfloat diffuse1[] = { 0,0,0.502,1 };
GLfloat mat_shininess[] = { 50 };

matprop(ambient1, diffuse1, specular1, mat_shininess);
glPushMatrix();
glTranslated(0, 4, 0);
glScaled(1, .1, 1);

//left wall
glPushMatrix();
glTranslated(0, 0, -.02 - .25);
glScaled(1, 1, 1.1);
glRotated(90.0, 0, 0, 1);
wall(0.08);
glPopMatrix();
//right wall
glPushMatrix();
glTranslated(6 + .12, 0, -.02 - .27);
glScaled(1, 1, 1.1);
glRotated(90.0, 0, 0, 1);
wall(0.08);
glPopMatrix();
//back wall
glPushMatrix();
glTranslated(-.08, 0, -.21);
glScaled(1.5 + .05, 1, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();
//front wall
glPushMatrix();
glTranslated(-.08, 0, 4 + .11);
glScaled(1.5 + .05, 1, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();
glPushMatrix();
glTranslated(-.04, 2, 4);
glScaled(.08, 4, .1);
glutSolidCube(1);
glPopMatrix();

glPopMatrix();
```

```
}

void fanwing(float winglen)// fan er pakha
{
   glPushMatrix();

   glRotated(90, 1, 0, 0);
   glRotated(90, 0, 1, 0);
   glScaled(1, 15, 1);
   gluCylinder(Cylinder, .01, .01, 1, 4, 1);
   glPopMatrix();
}

void fanbottom()
{
   glPushMatrix();

   glTranslated(1, 3.2, 1);
   glPushMatrix();
   glRotated(90, 1, 0, 0);
   gluCylinder(Cylinder, .2, .2, .05, 128, 16);
   glPopMatrix();

   glPushMatrix();
   glTranslated(0, 0.00025, 0);
   glRotated(90, 1, 0, 0);
   gluDisk(Disk, 0, .2, 32, 16);
   glPopMatrix();

   glPushMatrix();
   glTranslated(0, -.05, 0);
   glRotated(90, 1, 0, 0);
   gluCylinder(Cylinder, .2, .15, .1, 128, 16);
   glPopMatrix();

   glPushMatrix();
   glTranslated(0, -.1, 0);
   glRotated(90, 1, 0, 0);
   gluDisk(Disk, 0, .15, 32, 16);
   glPopMatrix();

   glPushMatrix();
   glTranslated(0.1, 0.0, 0);
   fanwing(.6);
```

```
   glPopMatrix();
   glPushMatrix();
   glRotated(120, 0, 1, 0);
   glTranslated(.1, 0, 0);
   fanwing(.6);
   glPopMatrix();
   glPushMatrix();
   glRotated(240, 0, 1, 0);
   glTranslated(0.1, 0.0, 0);
   fanwing(.6);
   glPopMatrix();
   glPopMatrix();
}
void fan(void)
{
   glPushMatrix();
   glTranslated(2.5, 1.9, 0);
   glScaled(.5, .5, .5);
   GLfloat mat_ambient[] = { .5,0,0,1 };
   GLfloat mat_specular[] = { 0,1,1,0 };
   GLfloat mat_diffuse[] = { 0,.502,0,1 };
   GLfloat mat_shininess[] = { 50 };


   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);


   if (flag == -1)
   {
         glPushMatrix();
         fanbottom();
         glPopMatrix();
   }
   else

   {

         angle += speed;
         glPushMatrix();
         glTranslated(1, 0, 1);
         glRotated(angle, 0, 1, 0);
```

```
            glTranslated(-1, 0, -1);
            fanbottom();
            glPopMatrix();
      }

      glPushMatrix();
      glTranslatef(1, 3.3, 1);
      glRotated(-90, 1, 0, 0);
      gluCylinder(Cylinder, .1, 0.005, .25, 16, 16);
      glPopMatrix();
      glPushMatrix();

      glTranslatef(1, 4, 1);
      glRotated(90, 1, 0, 0);
      gluCylinder(Cylinder, .006, 0.006, .6, 16, 16);
      glPopMatrix();

      glPushMatrix();
      glTranslatef(1, 3.96, 1);
      glRotated(90, 1, 0, 0);
      gluCylinder(Cylinder, .1, 0.005, .25, 16, 16);
      glPopMatrix();
      glPopMatrix();
      if (flag == 1)
            glutPostRedisplay();
}
void cleg(float cllen, float clwid)
{
      glRotated(90, 1, 0, 0);
      gluCylinder(Cylinder, clwid, clwid, cllen, 32, 32);

}
void sleg(float len, float thk)
{
      glScaled(thk, len, thk);
      glutSolidCube(1);

}
void solar(void)
{
      GLfloat     ambient1[] = { .1,.1,.1,1 };
      GLfloat specular1[] = { 1,1,1,1 };
      GLfloat diffuse1[] = { 1,1,1,1 };
      GLfloat mat_shininess[] = { 50 };
```

```
   matprop(ambient1, diffuse1, specular1, mat_shininess);
   GLfloat lightIntensity[] = { .7,.7,.7,1 };
   GLfloat light_position[] = { -20,4,60,0 };
   glLightfv(GL_LIGHT2, GL_POSITION, light_position);
   glLightfv(GL_LIGHT2, GL_DIFFUSE, lightIntensity);
   glEnable(GL_LIGHT2);
}

void myclock()
{


   GLfloat mat_ambient[] = { .4,.8,.4,1 };
   GLfloat mat_specular[] = { 1,1,1,1 };
   GLfloat mat_diffuse[] = { 0,.749,1,1 };
   GLfloat mat_shininess[] = { 50 };
   matprop(mat_ambient, mat_diffuse, mat_specular, mat_shininess);


   int hour_ticks, sec_ticks;
   glPushMatrix();
   glTranslated(2, 3.2, -.02);
   glScaled(.03, .06, .03);



   glPushMatrix(); // clock face
   glTranslatef(0, 0, 1.0);
   gluDisk(Disk, 0, 7, 32, 16);

   glPopMatrix();
   GLfloat mat_ambien[] = { 1,0,0,1 };
   matprop(mat_ambien, mat_diffuse, mat_specular, mat_shininess);

   glPushMatrix();
   glTranslatef(0, 0, 1.95);
   gluDisk(Disk, 0, .8, 32, 16);

   glPopMatrix();

   GLfloat     ambient[] = { 0,0,0,1 };
   GLfloat specular[] = { 1,1,1,1 };
   GLfloat diffuse[] = { 0,0,0,1 };
```

```
   matprop(ambient, diffuse, specular, mat_shininess);
   // hourer kata
   glPushMatrix();
   glColor3f(1.0, 0.5, 0.5);
   glTranslatef(0, 0, 1.5);
   glRotatef(-(360 / 12) * (newtime->tm_hour + newtime->tm_min / 60.0), 0.0, 0.0,
1.0);

   glRotatef(-90, 1.0, 0.0, 0.0);
   gluCylinder(Cylinder, 0.45, 0, 4, 16, 16);
   glPopMatrix();
   GLfloat     ambient1[] = { 0,0,1,1 };
   GLfloat specular1[] = { 1,1,1,1 };
   GLfloat diffuse1[] = { 0,0,1,1 };
   matprop(ambient1, diffuse1, specular1, mat_shininess);
   // minuter kata
   glPushMatrix();
   glColor3f(1.0, 0.5, 1.0);
   glTranslatef(0, 0, 1.25);
   glRotatef(-(360 / 60) * newtime->tm_min, 0.0, 0.0, 1.0);

   glRotatef(-90, 1.0, 0.0, 0.0);
   gluCylinder(Cylinder, 0.4, 0, 6, 16, 16);
   glPopMatrix();

   GLfloat     ambient2[] = { 1,0,0,1 };
   GLfloat specular2[] = { 1,1,1,1 };
   GLfloat diffuse2[] = { 1,0,0,1 };
   matprop(ambient2, diffuse2, specular2, mat_shininess);
   // seconder kata
   glPushMatrix();
   glTranslatef(0, 0, 1);
   glRotatef(-(360 / 60) * newtime->tm_sec, 0.0, 0.0, 1.0);
   glRotatef(-90, 1.0, 0.0, 0.0);
   gluCylinder(Cylinder, 0.3, 0, 6, 16, 16);
   glPopMatrix();



   GLfloat     ambient3[] = { 1,1,1,1 };
   GLfloat specular3[] = { 1,1,1,1 };
   GLfloat diffuse3[] = { 0,0,0,1 };
   matprop(ambient3, diffuse3, specular3, mat_shininess);
```

```
    for (hour_ticks = 0; hour_ticks < 12; hour_ticks++)
    {
          glPushMatrix();// Draw next arm axis.
          glTranslatef(0.0, 0.0, 1);
          glRotatef((360 / 12) * hour_ticks, 0.0, 0.0, 1.0);
          glTranslatef(6.0, 0.0, 0.0);
          glutSolidCube(.8);
          glPopMatrix();
    }

    for (sec_ticks = 0; sec_ticks < 60; sec_ticks++)
    {
          glPushMatrix();
          glTranslatef(0.0, 0.0, 1.1);
          glRotatef((360 / 60) * sec_ticks, 0.0, 0.0, 1.0);
          glTranslatef(6.0, 0.0, 0.0);
          glutSolidCube(0.25);
          glPopMatrix();
    }


    glPopMatrix();

}
void window(void)
{
    int i;
    GLfloat lightIntensity[] = { .5,.9,.9,1 };
    GLfloat light_position[] = { -20,4,-60,0 };
    glLightfv(GL_LIGHT1, GL_POSITION, light_position);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, lightIntensity);

    glEnable(GL_LIGHT1);

    glPushMatrix();
    glTranslated(3.185, 1, 3.95);
    //left edge of window
    glPushMatrix();
    glTranslated(.02, 1, .02);
    glScaled(.04, 2.2, .04);
    glutSolidCube(1);
    glPopMatrix();
    //right edge
    glPushMatrix();
```

```
glTranslated(1.6 + .02, 1, 0.02);
glScaled(.04, 2.2, .04);
glutSolidCube(1);
glPopMatrix();
//top edge
glPushMatrix();
glTranslated(.9, 2 + .02, 0.02);
glScaled(1.8, .04, .04);
glutSolidCube(1);
glPopMatrix();
//bottom edge
glPushMatrix();
glTranslated(.8, .02, 0.02);
glScaled(1.8, .04, .04);
glutSolidCube(1);
glPopMatrix();

for (i = 1;i <= 3;i++)
{

      glPushMatrix();
      glTranslated(.4 * i, 0, 0);

      glRotated(-90, 1, 0, 0);
      gluCylinder(Cylinder, .01, .01, 2, 32, 32);
      glPopMatrix();
}

for (i = 1;i <= 3;i++)
{
      glPushMatrix();
      glTranslated(.1 + .4 * i, 0, 0);

      glRotated(-90, 1, 0, 0);
      gluCylinder(Cylinder, .01, .01, 2, 32, 32);
      glPopMatrix();
}

for (i = 1;i <= 4;i++)
{
      glPushMatrix();
      glTranslated(0, .4 * i, 0);

      glRotated(90, 0, 1, 0);
```

```
            gluCylinder(Cylinder, .03, .03, 1.6, 32, 32);
            glPopMatrix();
      }

      glPopMatrix();


   }
   void gate(void)
   {
      int i;
      GLfloat ambient1[] = { 1,.5,1,1 };
      GLfloat specular1[] = { 1,1,1,1 };
      GLfloat diffuse1[] = { .6,.2,.8,1 };
      GLfloat mat_shininess[] = { 120 };

      matprop(ambient1, diffuse1, specular1, mat_shininess);
      glPushMatrix();
      // main gate
      if (mgo == 1)
            glTranslated(1.5, 0, 0);
      glTranslated(-1.3, 0, 6);
      //top frame gate
      glPushMatrix();
      glTranslated(0, 1.5, 0);
      glScaled(1.7, .04, .04);
      glutSolidCube(1);
      glPopMatrix();
      //bottom fram gate
      glPushMatrix();
      glTranslated(0, .05, 0);
      glScaled(1.7, .04, .04);
      glutSolidCube(1);
      glPopMatrix();
      //left frame gate
      glPushMatrix();
      glTranslated(-.8, .75, 0);
      glScaled(.04, 1.5, .04);
      glutSolidCube(1);
      glPopMatrix();
      //right frame gate
      glPushMatrix();
      glTranslated(.8, .75, 0);
      glScaled(.04, 1.5, .04);
```

```
    glutSolidCube(1);
    glPopMatrix();
    //horizantal pipes gate
    for (i = 1;i <= 3;i++)
    {
          glPushMatrix();
          glTranslated(-.85, .4 * i, 0);
          glRotated(90, 0, 1, 0);
          gluCylinder(Cylinder, .02, .02, 1.7, 32, 32);
          glPopMatrix();
    }
    //vertical strips gate
    for (i = 1;i <= 5;i++)
    {
          glPushMatrix();
          glTranslated(-.9 + .3 * i, .75, 0);
          glScaled(.2, 1.5, .02);
          glutSolidCube(1);
          glPopMatrix();
    }

    glPopMatrix();
}

void house(void)
{
    GLfloat mat_ambient[] = { 1,0,0,1 };
    GLfloat mat_specular[] = { 1,1,1,1 };
    GLfloat mat_diffuse[] = { 1,0,.7,1 };
    GLfloat mat_shininess[] = { 50 };

    matprop(mat_ambient, mat_diffuse, mat_specular, mat_shininess);


    GLfloat lightIntensity4[] = { .7,.7,.7,.7 };
    GLfloat light_position4[] = { 3,1,.5,1 };
    glLightfv(GL_LIGHT6, GL_POSITION, light_position4);
    glLightfv(GL_LIGHT6, GL_DIFFUSE, lightIntensity4);
    glEnable(GL_LIGHT6);


    glPushMatrix();
    glTranslated(0, .15, 0);
    //roof
```

```
glPushMatrix();
glTranslated(-.02 * 4, 3.9, -.01 * 4 - .25);
glScaled(1.5 + .05, 1.5, 1.1);
wall(0.08);
glPopMatrix();


GLfloat ambient2[] = { 1,0,0,1 };
GLfloat specular2[] = { 1,1,1,1 };
GLfloat diffuse2[] = { 0,0,0,1 };
GLfloat shininess[] = { 50 };
matprop(ambient2, diffuse2, specular2, shininess);

//floor
glPushMatrix();
glTranslated(-.02 * 3, -0.05, -.01 * 4);
glScaled(1.5 + .01, 1.5, 1);
wall(0.08);
glPopMatrix();


GLfloat ambient1[] = { 1,0,0,1 };
GLfloat specular1[] = { 1,1,1,1 };
GLfloat diffuse1[] = { 1.000, 0.271, 0.000,1 };
GLfloat shininess1[] = { 50 };
matprop(ambient1, diffuse1, specular1, shininess1);

//left wall
glPushMatrix();
glRotated(90.0, 0, 0, 1);
wall(0.08);
glPopMatrix();
//right wall
glPushMatrix();
glTranslated(6, 0, 0);
glRotated(90.0, 0, 0, 1);
wall(0.08);
glPopMatrix();
//back wall
glPushMatrix();
glTranslated(-.08, 0, 0);
glScaled(1.5 + .02, 1, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
```

```
glPopMatrix();
//room vertical wall
glPushMatrix();
glTranslated(4, 0, 0);
glScaled(1, 1, .5);
glRotated(90.0, 0, 0, 1);
wall(0.08);
glPopMatrix();
//room horizantal wall
glPushMatrix();
glTranslated(4.4, 0, 2);
glScaled(.4, 1, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();
//wall above the room door
glPushMatrix();
glTranslated(4, 3, 2);
glScaled(.11, .25, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();
//left room horizantal wall
glPushMatrix();
glTranslated(0, 0, 2);
glScaled(.4, 1, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();
//lroom vertical wall
glPushMatrix();
glTranslated(1.6, 0, 0);
glScaled(1, 1, .35);
glRotated(90.0, 0, 0, 1);
wall(0.08);
glPopMatrix();
//entrance room right wall
glPushMatrix();
glTranslated(1.6, 0, 2.59);
glScaled(1, 1, .35);
glRotated(90.0, 0, 0, 1);
wall(0.08);
glPopMatrix();
//wall above main door
```

```
glPushMatrix();
glTranslated(-0.02, 3, 4);
glScaled(.13, .23, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();
//wall right to the main door
glPushMatrix();
glTranslated(.48, 0, 4);
glScaled(.68, 1, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();
//wall right to the window
glPushMatrix();
glTranslated(4.8, 0, 4);
glScaled(.3, 1, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();
//wall below the window
glPushMatrix();
glTranslated(3.2, 0, 4);
glScaled(.4, .25, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();
//wall above the window
glPushMatrix();
glTranslated(3.2, 3.03, 4);
glScaled(.4, .23, 1);
glRotated(-90.0, 1, 0, 0);
wall(0.08);
glPopMatrix();

fan();
myclock();
terece();
window();
solar();

GLfloat     ambient[] = { 1,0.5,.5,1 };
GLfloat specular[] = { 1,1,1,1 };
GLfloat diffuse[] = { 0.502, 0.502, 0.000,1 };
```

```
    matprop(ambient, diffuse, specular, mat_shininess);
    //main door
    glPushMatrix();
    glTranslated(0, 0, 4);
    glRotated(maino, 0, 1, 0);
    glTranslated(0, 0, -4);
    glPushMatrix();
    glTranslated(0, 0, 4);
    glScaled(.12, .75, 1);
    glRotated(-90.0, 1, 0, 0);
    wall(0.04);
    glPopMatrix();

    glPushMatrix();
    glTranslated(0, 0, 4);
    glScaled(.5, 1, .2);
    glRotated(-90, 1, 0, 0);
    gluCylinder(Cylinder, 0.05, 0.05, 3, 16, 16);
    glPopMatrix();
    glPopMatrix();
    //inside room door
    glPushMatrix();
    glTranslated(4, 0, (2 - .025));
    glRotated(romo, 0, 1, 0);
    glTranslated(-4, 0, -(2 - .025));
    glPushMatrix();
    glTranslated(4, 0, 2);
    glScaled(.099, .75, 1);
    glRotated(-90.0, 1, 0, 0);
    wall(0.01);
    glPopMatrix();

    glPushMatrix();
    glTranslated(4.01, 0, 2 - .025);
    glScaled(.5, 1, .6);
    glRotated(-90, 1, 0, 0);
    gluCylinder(Cylinder, 0.05, 0.05, 3, 16, 16);
    glPopMatrix();

    glPopMatrix();
    glPopMatrix();
    glFlush();
}
```

```
void display(void)
{
   time(&ltime); // Get time
   newtime = localtime(&ltime); // Convert to local time
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   gluLookAt(view[0], view[1], view[2], look[0], look[1], look[2], 0.0, 1.0, 0.0);
   earth();
   compound();
   house();
   glFlush();
   glutSwapBuffers();
   glutPostRedisplay();
}

void Keyboard(unsigned char key, int x, int y)
{
   switch (key)
   {
         //go inside
   case '1':
         view[2] -= .1;
         glutPostRedisplay();
         break;
         //go outside
   case '2':
         view[2] += .1;
         glutPostRedisplay();
         break;

         //to run and stop the fan
   case 'S':
   case 's':
         flag *= -1;
         glutPostRedisplay();
         break;
         //to open and close the main door
   case 'p':
   case 'P':
         if (maino == 0)
               maino = 85;
         else
```

```
            maino = 0;
      break;
      //to open and close inside room door
case 'O':
case 'o':
      if (romo == 0)
            romo = 75;
      else
            romo = 0;
      break;
      //to open and close main gate
case 'g':
case 'G':
      if (mgo == 0)
            mgo = 1;
      else
            mgo = 0;
      break;

      //inside view
case 'i':
case 'I':
      view[0] = 2.8;
      view[1] = 2;
      view[2] = 4.8;
      look[0] = 2.8;
      look[1] = 2;
      look[2] = 1;
      break;
      //top view
case 'T':
case 't':
      view[0] = 6;
      view[1] = 12;
      view[2] = 10;
      look[0] = 2;
      look[1] = 4;
      look[2] = 2;
      break;
      //front view
case 'f':
case 'F':
      view[0] = 2;
      view[1] = 2;
```

```
            view[2] = 12.9;
            look[0] = 2;
            look[1] = 2;
            look[2] = 3;
            break;
            //back view
      case 'b':
      case 'B':
            view[0] = 1;
            view[1] = 6;
            view[2] = -7;
            look[0] = 2;
            look[1] = 4;
            look[2] = 2;
            break;

      }
   }
   int main(int argc, char** argv)
   {
      printf("**<<Press G for Gate on & off>>**\n");
      printf("**<<Press P for Main door & O for insider door on & off>>**\n");
      printf("**<<Press I for inside view & T for top view >>**\n");
      printf("**<<Press B for back view & F for front view>>**\n");
      printf("**<<Press S for Fan on & off>>**\n");
      printf("**<<Press 1 for go inside slow & 2 for back outside>>**\n");
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
      glutInitWindowSize(1200, 600);
      glutInitWindowPosition(100, 100);
      glutCreateWindow("Sweet Home");
      myinit();
      glutDisplayFunc(display);
      glutKeyboardFunc(Keyboard);
      glEnable(GL_LIGHTING);
      glEnable(GL_LIGHT0);
      glShadeModel(GL_SMOOTH);//smooth shaded
      glEnable(GL_DEPTH_TEST);//to remove hidden surface
      glEnable(GL_NORMALIZE);//to make normal vector to unit vector
      glClearColor(0, .20, .88, 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 an 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	Press 1	Slowly zoom-in and enter House	Slowly zoomed-in and entered House	Pass
2	Press 2	Slowly zoom-out and exit House	Slowly Zoomed-out and exit House	Pass
3	Press G	The front gate opens	The front gate opened	Pass
4	Press P	Main door opens	Main door opened	Pass
5	Press T	Display top view of House	Displayed top view	Pass
6	Press B	Display back view of House	Displayed back view	Pass
7	Press F	Display front view of House	Displayed front view	Pass
8	Press I	Display Inside of House	Displayed inside of House	Pass
9	Press S	Fan in room spins	Fan spun	Pass
10	Press O	Opens bedroom door	Opened bedroom door	Pass

CHAPTER 6

SNAPSHOTS

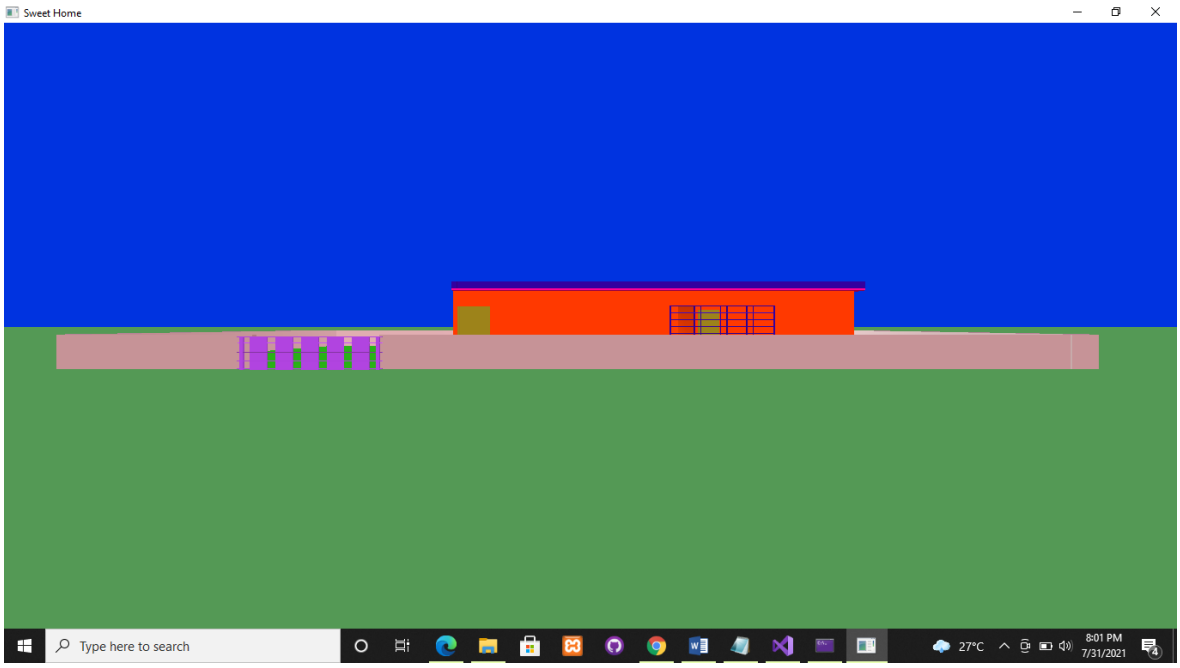


Figure 6.1: Front-view of House (On Pressing F)

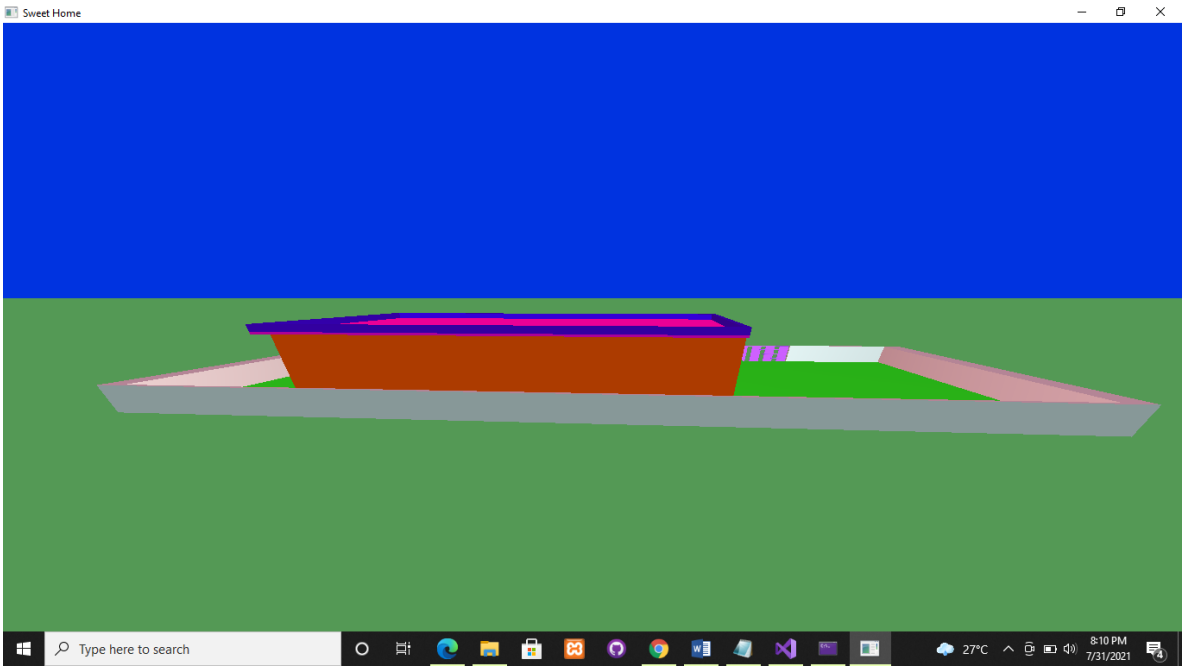


Figure 6.2: Back-view of House (On Pressing B)

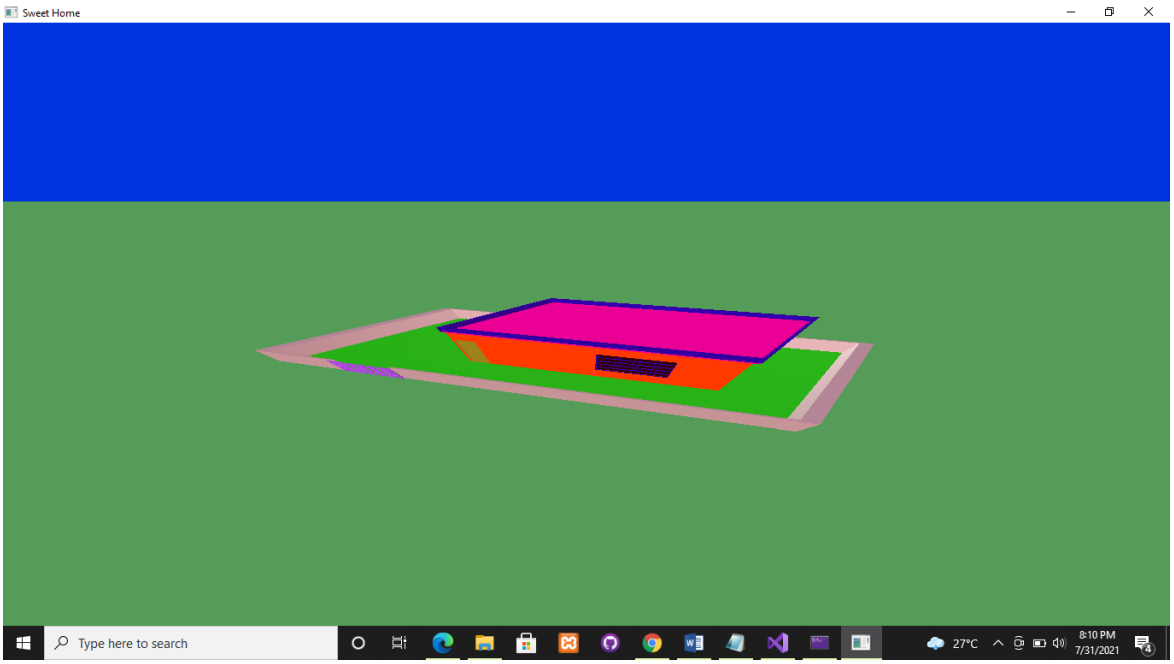


Figure 6.3: Top-view of House (On Pressing T)

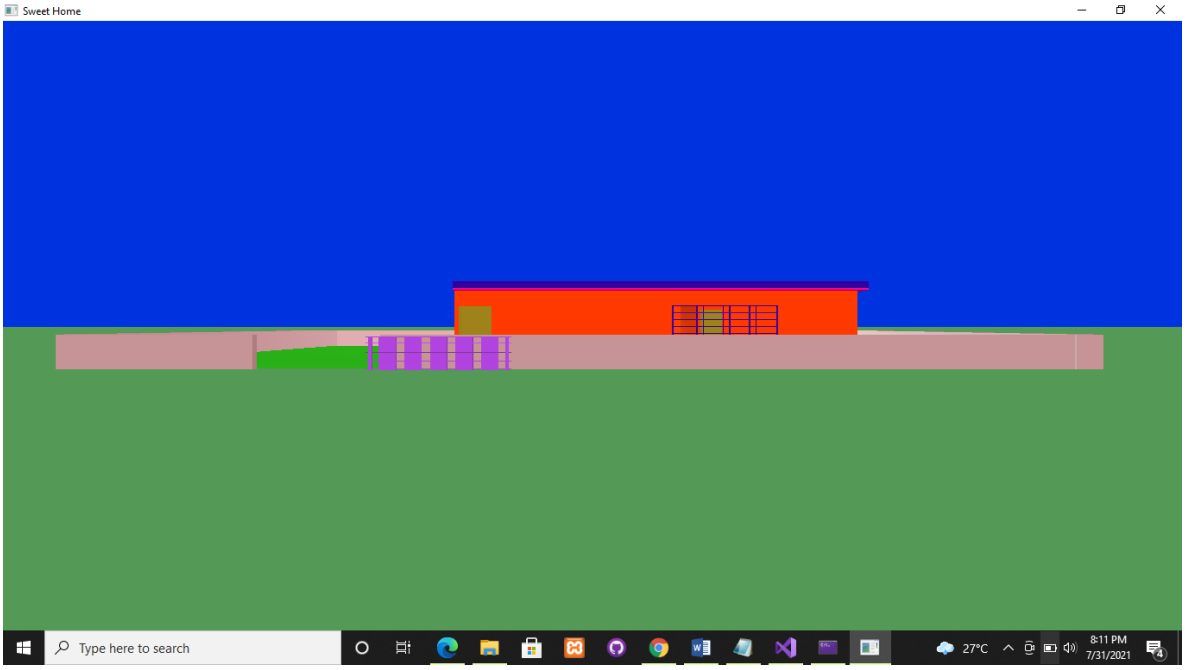


Figure 6.4: Main Gate Opens (On Pressing G)

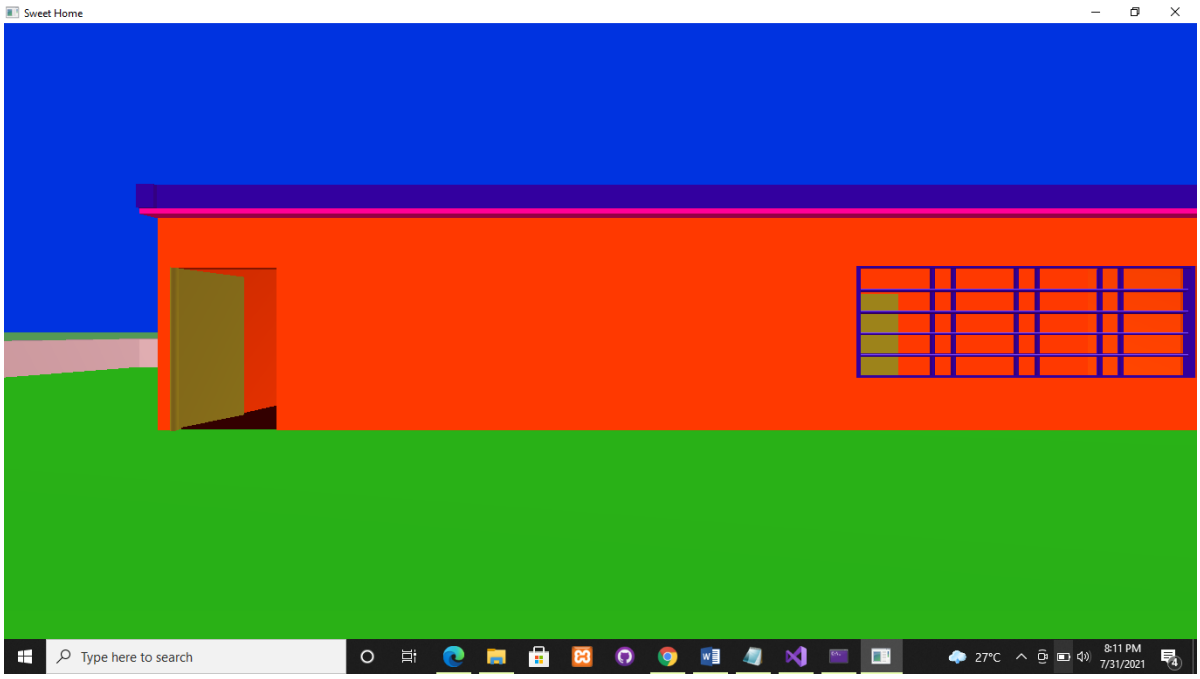


Figure 6.5: Front door Opens (On Pressing P)

Figure 6.6: Interior of House (On Pressing I)

Figure 6.7: Bedroom door Opens (On Pressing O)

Figure 6.8: Clock Showing current time & Fan Spinning (On Pressing S)

CONCLUSION

The 3D House has been tested under Windows 10 and has been found to provide ease of use and manipulation to the user. The 3D house created for the Windows 10 operating system can be used to draw lines, boxes, circles, ellipses, and polygons. It has a very simple and aesthetic user interface. We found designing and developing this 3D House as a very interesting and learning experience. It helped us to learn about computer graphics, design of Graphical User Interfaces, interface to the user, user interaction handling and screen management. The graphics editor provides all and more than the features that have been detailed in the university syllabus.

The user-friendly interface allows the user to interact with it very effectively. So, I conclude on note that this project has given me a great exposure to the OpenGL and computer graphics. This is very reliable graphics package supporting various primitive objects like polygon, line loops, etc. Also, color selection, menu and mouse-based interface are included. Transformations like translation, rotation, scaling is also provided.

BIBLIOGRAPHY

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

[3]. https://open.gl/

[4]. https://en.wikipedia.org/wiki/FreeGLUT

[5]. https://www.stackoverflow.com