VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum: 590018

A Mini Project Report (18CSL67)
on

3D HOUSE

A mini project report submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering
in
Computer Science & Engineering

Submitted by

KEERTHANA L(1AY18CS050)
NANDINI G(1AY18CS070)

Under the guidance of
Mrs VARALAKSHMI B D

Department of Computer Science & Engineering

Acharya Institute of Technology
Department of Computer Science & Engineering
Soladevanahalli, Bangalore-560107

ACHARYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belgaum)
Soladevanahalli, Bangalore – 560 107

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

Certified that the Computer Graphics Mini Project entitled **"3D HOUSE "** is a bonafide work carried out by **Ms.KEERTHANA L (1AY18CS050) & Ms.NANDINI G(1AY18CS070)** in partial fulfillment for the award of degree of **Bachelor of Engineering in Computer Science & Engineering of the Visvesvaraya Technological University**, Belgaum during the academic year **2020-2021.** It is certified that all corrections/ suggestions indicated for internal assessments have been incorporated in the Report deposited in the departmental library. The Mini project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the **Bachelor of Engineering Degree**.

Signature of Guides **Signature of H.O.D**

Name of the Examiners **Signature with date**

1.

2.

ACKNOWLEDGEMENT

We express our gratitude to our institution and management for providing us with good infrastructure, laboratory facilities and inspiring staff, and whose gratitude was of immense help in completion of this mini project successfully.

We express our sincere gratitude to our principal, **Dr. M R Prakash** for providing us the required environment and for his valuable suggestions.

Our sincere thanks to **Dr. Prashanth C M,** Head of the Department. Computer Science and Engineering, Acharya Institute of Technology for his valuable support and for rendering us the resources for this mini project.

We heartily thank **Prof. Varalakshmi B D, Prof. Vani K S and Prof. Swathi Mohan** Assistant Professors, Department of Computer Science and Engineering, Acharya Institute of Technology who guided us with valuable suggestions in completing this mini project at every stage.

Our gratitude thanks should be rendered to many people who helped us in all possible ways.

KEERTHANA L(1AY18CS050)
NANDINI G(1AY18CS070)

ABSTRACT

A Computer Graphics project based on the concept of a “3D House”. This project implements the view of a 3D House both inside and outside of the house. The house is surrounded by a compound wall. The API’s that are used in implementing these components are glutSolidCube(), glutSolidTeapot(), glutSoildCone()… etc An animation has been implemented which shows the rotation of the fan inside the house. Lighting has been implemented by the inbuilt OPENGL lighting functions. Menu’s have been provided to modify the various features such as changing the background, lighting etc.This project implements the orthographic view. Options have been provided in the menu to switch between the views.It also implements Day and Night changes in the options.

The following concepts are implemented:

1)Rotation

2)Translation

3)Scaling

4)Lighting

5)Mouse and Keyboard interactions.

CONTENTS

List of Figures

Chapter 1

INTRODUCTION

1.1 Computer Graphics

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. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive. The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-screen.

Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. Th advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

1.2 OpenGL

OpenGL Interface:

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

1. Main GL: Library has names that begin with the letter gl and are stored in a library usually referred to as GL.
2. OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.
3. OpenGL Utility Toolkit (GLUT): This provides the minimum functionality that should be accepted in any modern windowing system.

OpenGL Overview:

• OpenGL(Open Graphics Library) is the interface between a graphic program and graphics hardware. It is streamlined. In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.

• OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc.

• It is system-independent: It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.

• It is a state machine. At any moment during the execution of a program there is a current model transformation.

• It is a rendering pipeline. The rendering pipeline consists of the following steps:

o Defines objects mathematically.

o Arranges objects in space relative to a viewpoint.

o Calculates the color of the objects.

o Rasterizes the objects.

Chapter 2

SYSTEM REQUIREMENTS SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

- Programming language – C/C++ using OpenGL
- Operating system – Linux operating system
- Compiler – C Compiler
- Graphics library – GL/glut.h
- OpenGL 2.0

2.2 HARDWARE REQUIREMENTS

- Dual Core Processor
- 2GB RAM
- 40GB Hard disk
- Mouse and other pointing devices
- Keyboard

2.2 FUNCTIONAL REQUIREMENTS

OpenGL APIs:

If we want to have a control on the flow of program and if we want to interact with the window system then we use OpenGL API'S. Vertices are represented in the same manner internally, whether they are specified as two-dimensional or three- dimensional entities, everything that we do are here will be equally valid in three dimensions. Although OpenGL is easy to learn, compared with other APIs, it is nevertheless powerful. It supports the simple three dimensional programs and also supports the advanced rendering techniques.

GL/glut.h:

We use a readily available library called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system. The application program uses only GLUT functions and can be recompiled with the GLUT library for other window system. OpenGL makes a heavy use of macros to increase code readability and avoid the use of magic numbers. In most implementation, one of the include lines.

Chapter 3
ABOUT THE PROJECT

3.1 INTRODUCTION

A Computer Graphics project based on the concept of a "3D House". This project implements the view of a 3D House both inside and outside of the house. There are 5 components that are placed inside the house they are fan,clock,chairs,a dining table and a teapot placed on the table. The house is surrounded by a compound wall. An animation has been implemented which shows the rotation of the fan inside the house and also speed of the fan can be increased or decreased by options given in the menu. Lighting has been implemented by the inbuilt OPENGL lighting functions. Menu's have been provided to modify the various features such as changing the background, lighting etc.

This project implements the orthographic view. Options have been provided in the menu to switch between the views ie. Front view, Top view, Back view.It also implements Day and Night changes in the provided menu options.

3.2 OBJECTIVES:

The aim of this project is to develop a graphics package which supports basic operations which include building a 3D HOUSE using Open GL. The package must also has a user-friendly interface. The objective of developing this model was to design and apply the skills we learnt in class.

3.3 BUILT-IN FUNCTIONS:

- The **glColor3f (float, float, float) :**
  This function will set the current drawing color
- **gluOrtho2D (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top):** which defines a two dimensional viewing rectangle in the plane z=0.
- **glClear( ):**
  Takes a single argument that is the bitwise OR of several values indicating which buffer is to be cleared.

- **glClearColor ():**

  Specifies the red, green, blue, and alpha values used by **glClear** to clear the color buffers.

- **GlLoadIdentity( ):**

  the current matrix with the identity matrix.

- **glMatrixMode(mode):**

  Sets the current matrix mode, mode can be GL_MODELVIEW, GL_PROJECTION or GL_TEXTURE.

- **Void glutInit (int *argc, char**argv):**

  Initializes GLUT, the arguments from main are passed in and can be used by the application.

- **Void glutInitDisplayMode (unsigned int mode):**

  Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model and buffering.

- **Void glutInitWindowSize (int width, int height):**

  Specifies the initial position of the topleft corner of the window in pixels.

- **Int glutCreateWindow (char *title):**

  A window on the display. The string title can be used to label the window. The return value provides references to the window that can be used when there are multiple windows.

- **Void glutMouseFunc(void *f(int button, int state, int x, int y):**

  Register the mouse callback function f. The callback function returns the button, the state of button after the event and the position of the mouse relative to the top-left corner of the window.

- **Void glutKeyboardFunc(void(*func) (void)):**

  This function is called every time when you press enter key to resume the game or when you press ‘b’ or ‘B’ key to go back to the initial screen or when you press esc key to exit from the application.

- **Void glutDisplayFunc (void (*func) (void)):**

  Register the display function func that is executed when the window needs to be redrawn.

- **Void glutSpecialFunc(void(*func)(void)):**

  This function is called when you press the special keys in the keyboard like arrow keys, function keys etc. In our program, the func is invoked when the up arrow or down arrow key is pressed for selecting the options in the main menu and when the left or right arrow key is pressed for moving the object(car) accordingly.

- **glut PostReDisplay ( ) :**

  which requests that the display callback be executed after the current callback returns.

- **Void MouseFunc (void (*func) void)):**

  This function is invoked when mouse keys are pressed. This function is used as an alternative to the previous function i.e., it is used to move the object(car) to right or left in our program by clicking left and right button respectively.

- **Void glutMainLoop () :**

  Cause the program to enter an event-processing loop. It should be the last statement in main function.

3.4 USER DEFINED FUNCTIONS:

- **void house(void):**

  This function displays the objects that are placed inside house for example,clock,fan,walls of the rooms,etc.

- **void window(void);**

  This function displays the window of the house.

- **void gate(void);**

  This function displays the main gate of the house . It can be opened and closed provided by the option in the menu. Also the key (“g”) can be used to Open/Close the gate.

- **void myclock():**

  This function displays the current time with hour and minute hands inside the house.

- **void earth(void):**

  This function displays the ground which displays on the window screen.

- **void compound(void):**

  This function displays the compounds that are surrounded the house

- **void fan(void):**

  This function displays the ON/OFF the fan and also the speed can be increased or decreased that is placed in the ceiling of the room.

- **void steps(void):**

  This function displays the steps to reach to the top of the house and is viewed in the orthographic view.

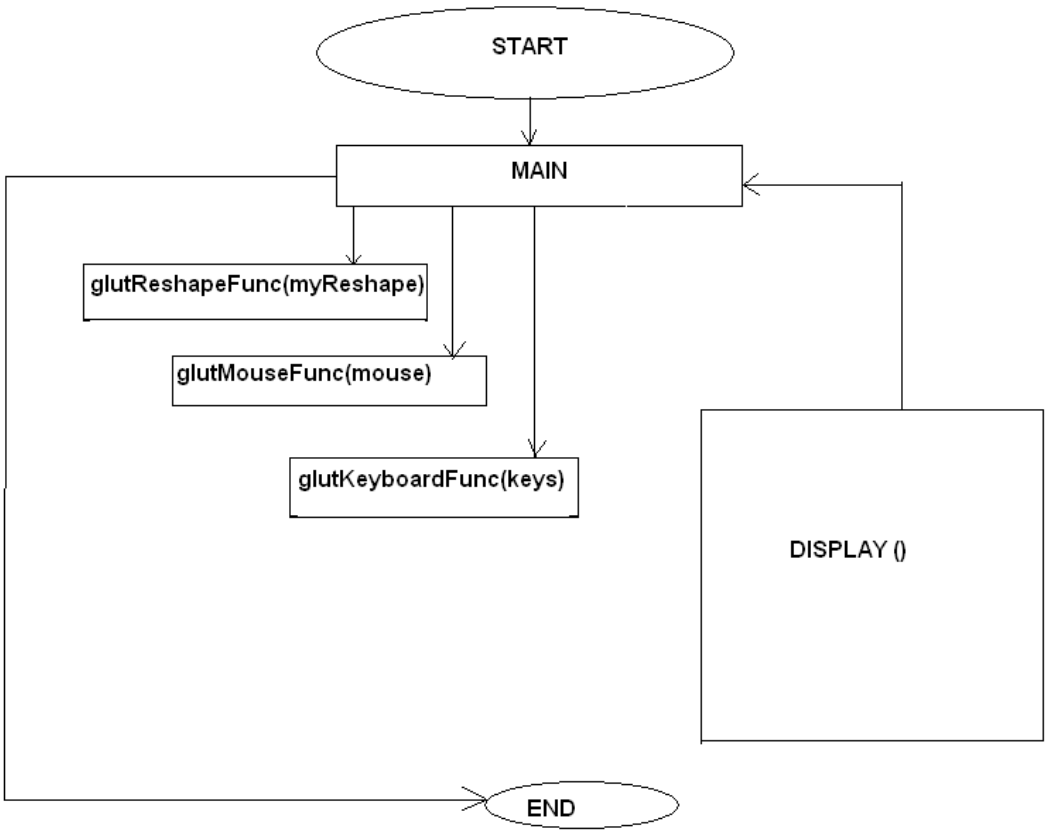
- **void sgate(void);**

  This function displays the sub gate of the house . It can be opened and closed provided by the option in the menu. Also the key ("h") can be used to Open/Close the sub gate.

- **void solar(void):**

  This function displays the solar rooftop photovoltaic (PV) at the top of house.

3.5 DATA FLOW DIAGRAM



3.5 SOURCE CODE / PSEUDO CODE:

```
#include<GL/glut.h>
#include <time.h>
double w=1280,h=720;
double view[3]={2,2,12.9};
double look[3]={2,2,2};
int flag=-1;
void steps(void);
void window(void);
void sgate(void);
void gate(void);
void light(void);
int lightflag = 0;
double angle=0,speed=5,maino=0,romo=0,tro=0,mgo=0,sgo=0;
GLUquadricObj *Cylinder;          //declarating quadric objects
GLUquadricObj *Disk;
struct tm *newtime;
time_t ltime;
GLfloat angle1;
void myinit(void)                 //initialisation
{
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-1.0,1.0,-1*w/h,1*w/h,1,200.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
 Cylinder = gluNewQuadric();   //defining new quadric object
 gluQuadricDrawStyle( Cylinder, GLU_FILL); //to set drawing style
 gluQuadricNormals( Cylinder,GLU_SMOOTH);   //to set automatic normals
```

```
 Disk = gluNewQuadric();
 gluQuadricDrawStyle( Disk, GLU_FILL);
 gluQuadricNormals( Disk, GLU_SMOOTH);
 GLfloat gam[]={0.2,.2,.2,1};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,gam);
}
void earth(void)
{
GLfloat ambient[]={1,0,0,1};
GLfloat specular[]={0,1,1,1};
GLfloat diffuse[]={.5,.5,.5,1};
GLfloat shininess[]={50};
matprop(ambient,diffuse,specular,shininess);          //to create earth
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)
{
GLfloat ambient[]={1,0,0,1};
GLfloat specular[]={0,1,1,1};
GLfloat diffuse[]={.7,1,.7,1};
GLfloat shininess[]={50};
```

```
matprop(ambient,diffuse,specular,shininess);
GLfloat lightIntensity[]={.7,.7,.7,1};
GLfloat light_position[]={2,6,1.5,0};
glLightfv(GL_LIGHT0,GL_POSITION,light_position);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightIntensity);
glPushMatrix();
glPushMatrix();              //left wall of compound
glTranslated(-4,0,-1-.04);
glRotated(90.0,0,0,1);
wall2(0.08);
glPopMatrix();
glPushMatrix();              //right wall of compound
glTranslated(8,0,-1-.02);
glRotated(90.0,0,0,1);
wall2(0.08);
glPopMatrix();
glPushMatrix();           //back wall of compound
glTranslated(2,.8,-1);
glRotated(-90,1,0,0);
glScaled(12,.02*4,1.6);
glutSolidCube(1.0);
glPopMatrix();
glPushMatrix();
glTranslated(-3,.8,6-.08);
glRotated(-90,1,0,0);         //front left wall of compound
glScaled(2,.02*4,1.6);
glutSolidCube(1.0);
glPopMatrix();
//front middle wall of compound
glPushMatrix();
```

```
glTranslated(2.5,.8,6-.08);
glRotated(-90,1,0,0);
glScaled(6,.02*4,1.6);
glutSolidCube(1.0);
glPopMatrix();
//front right wall of compound
glPushMatrix();
glTranslated(7,.8,6-.08);
glRotated(-90,1,0,0);
glScaled(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();
 sgate();
 glFlush();
}
void Keyboard(unsigned char key,int x,int y)
{
switch(key)
```

```
{
//to move the camera along -ve x axis
case '4':
view[0]-=.1;
glutPostRedisplay();
break;
//to move the camera along +ve x axis
case '6':
view[0]+=.1;
glutPostRedisplay();
break;
//to move the camera along +ve y axis
case '7':
view[1]+=.1;
glutPostRedisplay();
break;
//to move the camera along -ve y axis
case '1':
if(view[1]>1.9)
view[1]-=.1;
glutPostRedisplay();
break;
//to move the camera along -ve z axis
case '8':view[2]-=.1;
glutPostRedisplay();
break;
//to move the camera along +ve z axis
case '2':
view[2]+=.1;
glutPostRedisplay();
```

```
break;
//to run and stop the fan
case 'S':
case 's':
flag*=-1;
glutPostRedisplay();
break;
//to move the look position along +ve x axis
case 'r':
case 'R':
look[0]+=.1;
break;
//to move the look position along -ve x axis
case 'l':
case 'L':
look[0]-=.1;
break;
//to move the look position along +ve y axis
case 'U':
case 'u':
look[1]+=.1;
break;
//to move the look position along -ve y axis
case 'D':
case 'd':
look[1]-=.1;
break;
//to move the look position along +ve z axis
case 'f':
case 'F':
```

```
look[2]+=.1;
break;//to move the look position along -ve z axis
case 'B':
case 'b':
look[2]-=.1;
break;
//to open and close the main door
case 'q':
case 'Q':
if(maino==0)
 maino=85;
else
maino=0;
break;
//to open and close the below room door
case 'O':
case 'o':
if(romo==0)
 romo=75;
else
romo=0;
break;
//to open and close the main gate
case 'g':
case 'G':
if(mgo==0)
 mgo=1;
else
mgo=0;
break;
```

```
//to open and close the sub gate
case 'h':
case 'H':if(sgo==0)
 sgo=50;
else
sgo=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 'j':
case 'J':
```

```
view[0]=2;
view[1]=2;
view[2]=12.9;
look[0]=3;
look[1]=2;
look[2]=3;
break;
//back view
case 'k':
case 'K':
view[0]=1;
view[1]=6;view[2]=-7;
look[0]=2;
look[1]=4;
look[2]=2;
break;
}
}
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[]={.8,1,.8,1};
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 diningtable()
{
GLfloat ambient[]={1,0,0,1};
GLfloat specular[]={0,1,1,1};
GLfloat diffuse[]={.5,.5,.5,1};
GLfloat shininess[]={50};
matprop(ambient,diffuse,specular,shininess);
glPushMatrix();
glTranslated(3,0,1);
glScaled(1.5,1.5,1.5);
table(.3,.5,.025,.4,.005);

//back left chair
glPushMatrix();
glTranslated(-.1,0,-.1);
chair(.15,.15,.02,.3,.005);
glPopMatrix();
//back right chair
glPushMatrix();
glTranslated(.1,0,-.1);glRotated(180,0,1,0);
chair(.15,.15,.02,.3,.005);
glTranslated(0.1,0.5,0.0);
glutSolidTeapot(0.06);
glPopMatrix();
glPopMatrix();
}
```

```
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);//base
glPushMatrix();
glTranslated(4,4,3);
glPushMatrix();
glTranslated(0.4,.4,0);
glScaled(1,.8,1);
glutSolidCube(1);
glPopMatrix();
GLfloat ambient2[]={.7,.7,.7,1};
GLfloat specular2[]={1,1,1,1};
GLfloat diffuse2[]={1,1,1,1};
 matprop(ambient2,diffuse2,specular2,mat_shininess);
glPushMatrix();
glTranslated(0,.8,0);
glPushMatrix();
glTranslated(.6,.6,0);
gluCylinder(Cylinder,.1,.1,.4,32,32);
glPopMatrix();
GLfloat ambient3[]={1,0,.2,1};
```

```
GLfloat specular3[]={1,1,1,1};
GLfloat diffuse3[]={1,0,.5,1};
GLfloat mat_shininess3[]={50};
 matprop(ambient3,diffuse3,specular3,mat_shininess3);
glPushMatrix();
glTranslated(.6,.6,0);
gluDisk(Disk,0,.1,32,32);
glPopMatrix();
glPushMatrix();
glTranslated(.6,.6,0.4);
gluDisk(Disk,0,.1,32,32);
glPopMatrix();
GLfloat ambient4[]={0,0,0,1};
GLfloat specular4[]={1,1,1,1};
GLfloat diffuse4[]={0,0,0,1};
GLfloat mat_shininess4[]={50};
matprop(ambient4,diffuse4,specular4,mat_shininess4);
glPushMatrix();
glTranslated(.5,.3,.05);
sleg(.6,.01);
glPopMatrix();
glPushMatrix();
glTranslated(.7,.3,.05);
sleg(.6,.01);
glPopMatrix();
glPushMatrix();
glTranslated(.5,.3,.35);
sleg(.6,.01);
glPopMatrix();
glPushMatrix();
```

```
glTranslated(.7,.3,.35);
sleg(.6,.01);
glPopMatrix();
glPushMatrix();
glRotated(45,0,0,1);
glTranslated(.3,.015,.2);
glScaled(.6,.03,.4);
glutSolidCube(1);
glPopMatrix();
glPushMatrix();
glTranslated(.4,.21,0);
sleg(.425,.01);
glPopMatrix();
glPushMatrix();
glTranslated(.4,.21,.4);
sleg(.425,.01);
glPopMatrix();
glPushMatrix();glTranslated(.4,.4,0);
glRotated(30,0,0,1);
glRotated(90,0,1,0);
gluCylinder(Cylinder,.01,.01,.2,32,32);
glPopMatrix();
glPopMatrix();
glPopMatrix();
}
void gate(void)
{
int i=1;
GLfloat ambient1[]={1,.5,1,1};
GLfloat specular1[]={1,1,1,1};
```

```
GLfloat diffuse1[]={.5,.5,.5,1};
GLfloat mat_shininess[]={120};
 matprop(ambient1,diffuse1,specular1,mat_shininess);
glPushMatrix();
//if flag mgo=1 the open the main gate
if(mgo==1)
glTranslated(1.5,0,0);
glTranslated(-1.3,0,6);
//top frame of the main gate
glPushMatrix();
glTranslated(0,1.5,0);
glScaled(1.7,.04,.04);
glutSolidCube(1);
glPopMatrix();
//bottom frame of main gate
glPushMatrix();
glTranslated(0,.05,0);
glScaled(1.7,.04,.04);
glutSolidCube(1);
glPopMatrix();
//left frame of the main gate
glPushMatrix();
glTranslated(-.8,.75,0);
glScaled(.04,1.5,.04);
glutSolidCube(1);
glPopMatrix();
//right frame of the main gate
glPushMatrix();
glTranslated(.8,.75,0);
glScaled(.04,1.5,.04);
```

```
glutSolidCube(1);
glPopMatrix();

                    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[]={1,1,.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[]={.7,1,0.8,1};
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[]={1,1,.7,1};
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,.27,1);
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,.25,1);
glRotated(-90.0,1,0,0);
wall(0.08);
```

```
glPopMatrix();
terece();
steps();
window();
fan();
diningtable();
myclock();
solar();
GLfloat ambient[]={1,0.5,.5,1};
GLfloat specular[]={1,1,1,1};
GLfloat diffuse[]={1,.5,.5,1};
 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();
//bolow 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)
{
if (!lightflag)
{
      glClearColor(0,0.3,0.8,0);
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();
glEnable(GL_LIGHTING);
}
else {
glClearColor(0,0.0,0.0,0);
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 menu()
{
int sub_menu1=glutCreateMenu(fan_menu);
glutAddMenuEntry("on/off fan(s)",1);
glutAddMenuEntry("speed up(up arrow)",2);
glutAddMenuEntry("speed down(down arrow)",3);
int sub_menu2=glutCreateMenu(door_menu);glutAddMenuEntry("main door(q)",1);
glutAddMenuEntry("ground floor room door(o)",2);
```

```
int sub_menu3=glutCreateMenu(gate_menu);
glutAddMenuEntry("main gate(g)",1);
glutAddMenuEntry("sub gate(h)",2);
int sub_menu4=glutCreateMenu(house_view);
glutAddMenuEntry("front view(j)",3);
glutAddMenuEntry("top view(t)",2);
glutAddMenuEntry("inside view(i)",1);
glutAddMenuEntry("back view(k)",4);
int sub_menu5=glutCreateMenu(lights_menu);
glutAddMenuEntry("day",1);
glutAddMenuEntry("night",2);
glutCreateMenu(main_menu);
glutAddMenuEntry("quit",1);
glutAddSubMenu("fan menu",sub_menu1);
glutAddSubMenu("open/close door",sub_menu2);
glutAddSubMenu("open/close gate",sub_menu3);
glutAddSubMenu("house view",sub_menu4);
glutAddSubMenu("day and night",sub_menu5);
glutAttachMenu(GLUT_RIGHT_BUTTON);
}
int main(int argc,char**argv)
{
glutInit(&argc,argv);                    //to initialize the glut library
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(w,h);
glutInitWindowPosition(0,0);
glutCreateWindow("er");
myinit();
glutDisplayFunc(display);
glutKeyboardFunc(Keyboard);
```

```
glutSpecialFunc(mySpecialKeyFunc);
menu();
glutFullScreen();                          //to see o/p in full screen on monitor
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(0.0,0.3,0.8,1);
glViewport(0,0,w,h);
glutMainLoop();
  return 0;
}
```

Chapter 4

RESULT

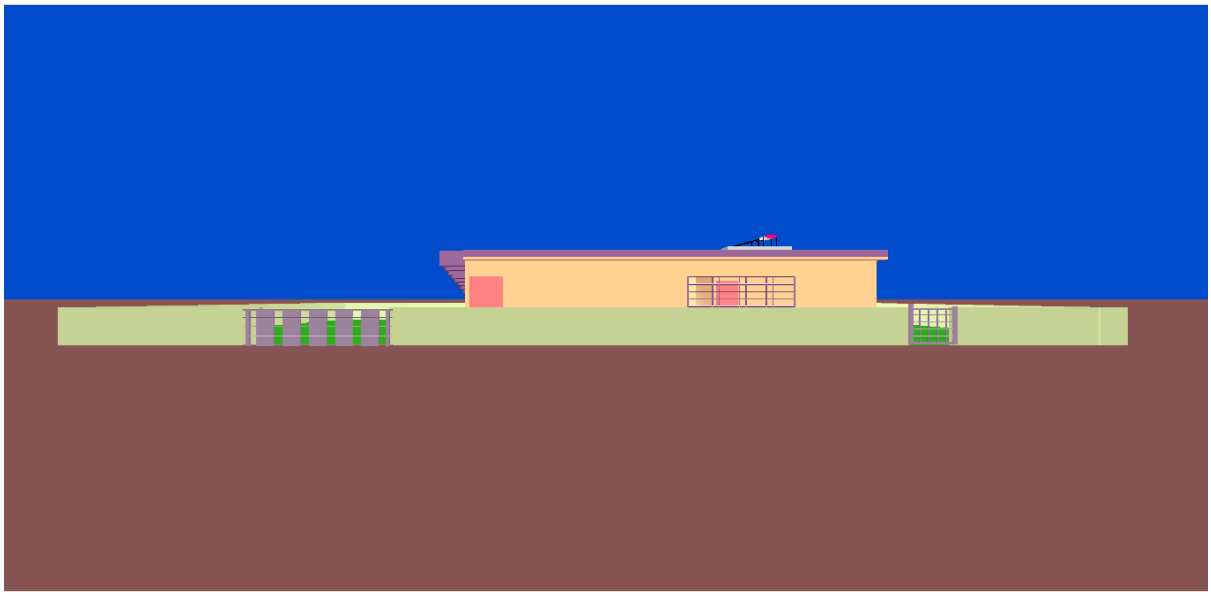


Fig 4.1: Final Output.

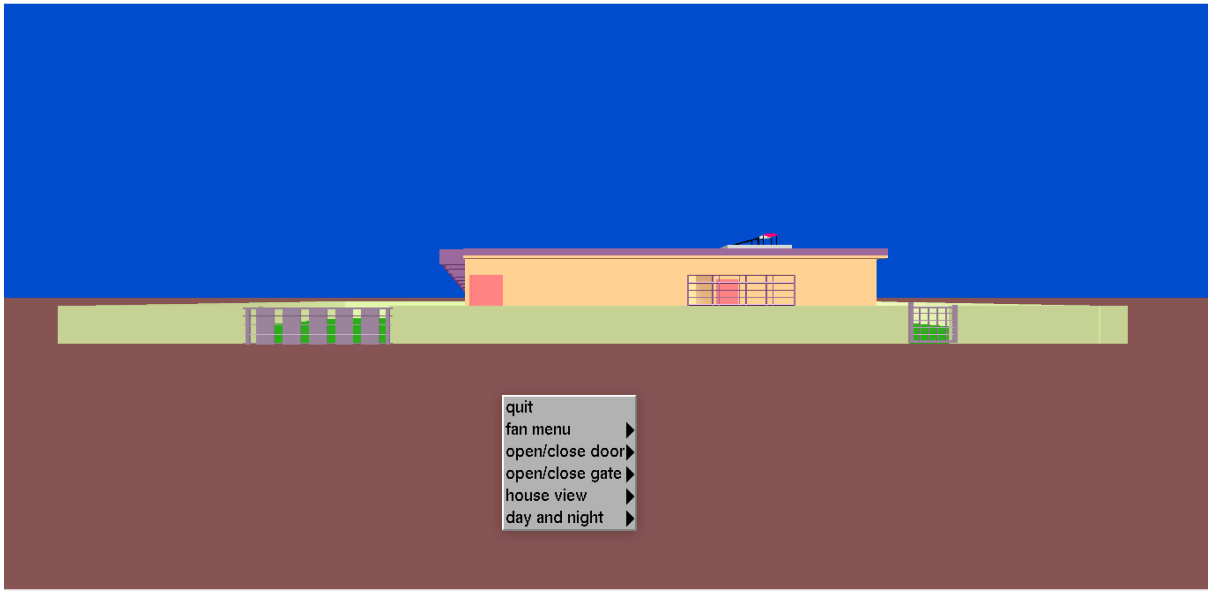


Fig 4.2: Main Menu with options.

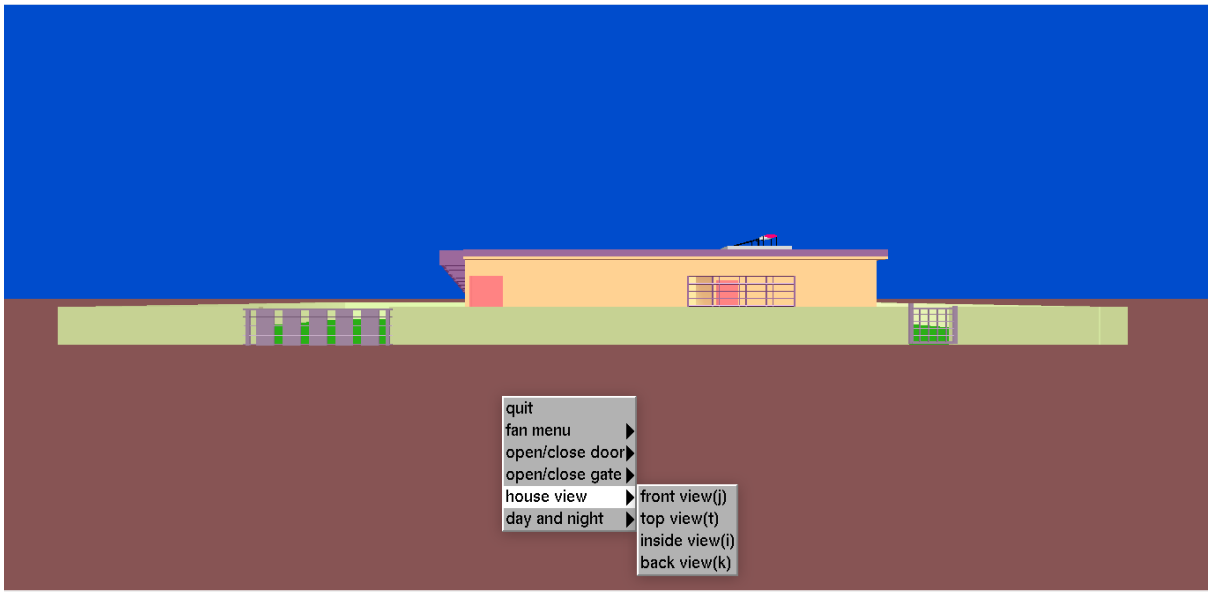


Fig 4.3: Front View of House

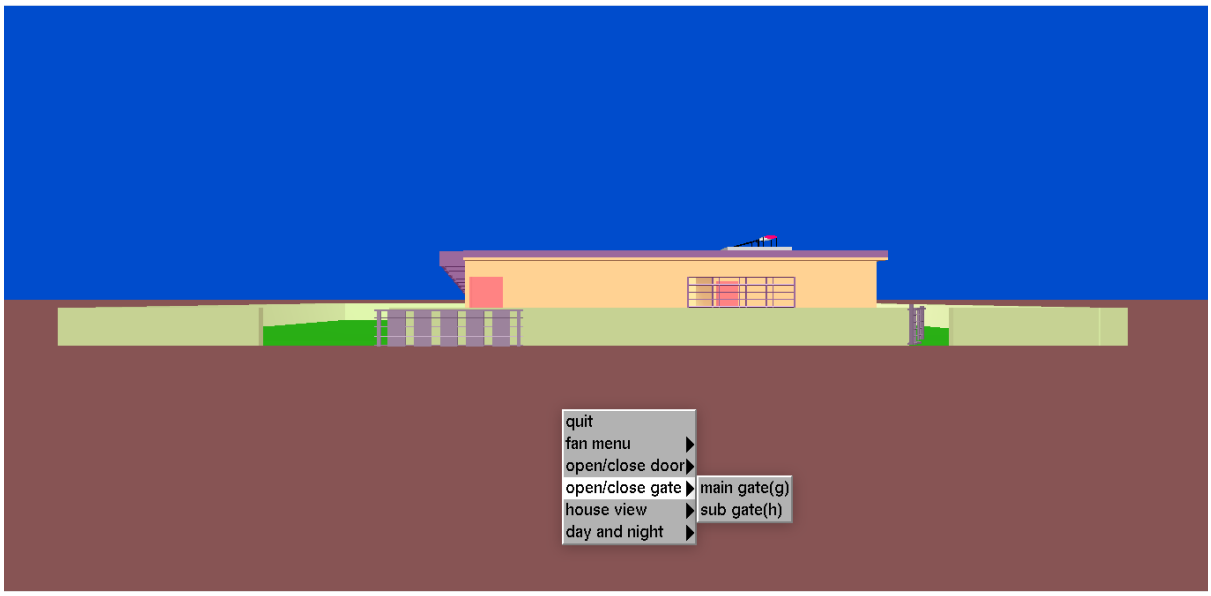


Fig 4.4: Main gate and Sub gate are opened.

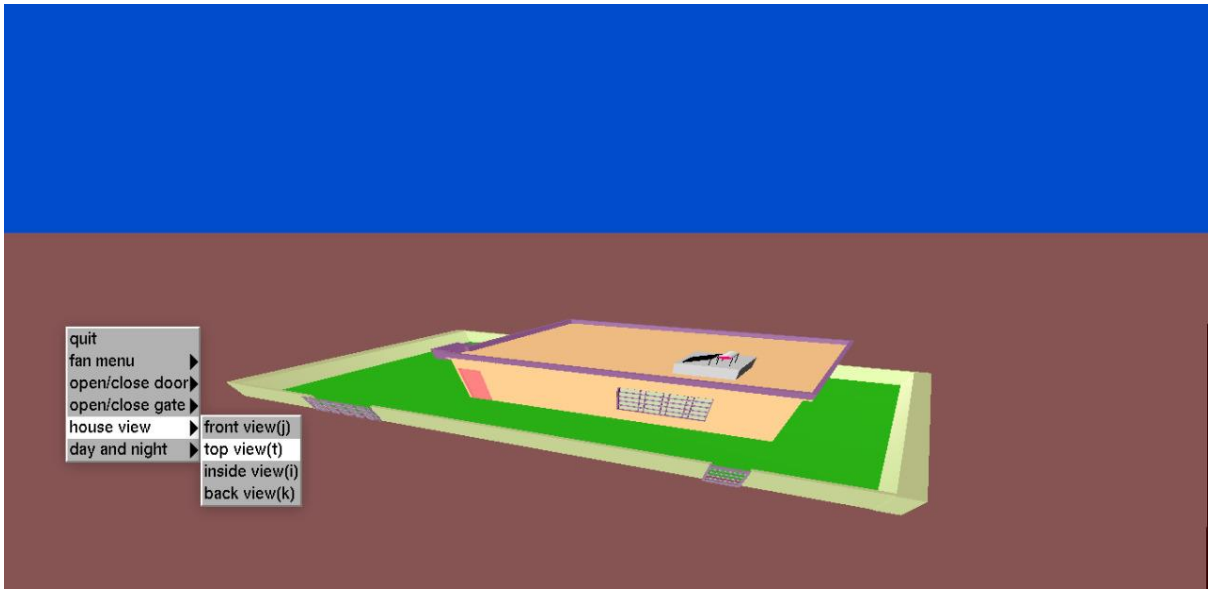


Fig 4.5: Top View of House

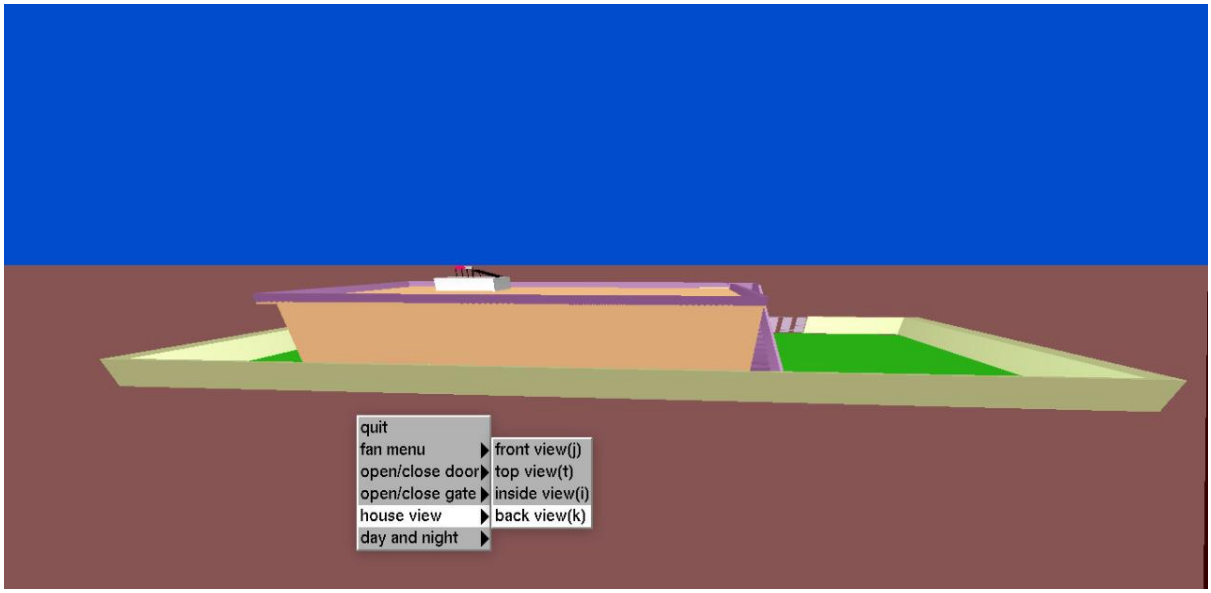


Fig 4.6: Back View of House.

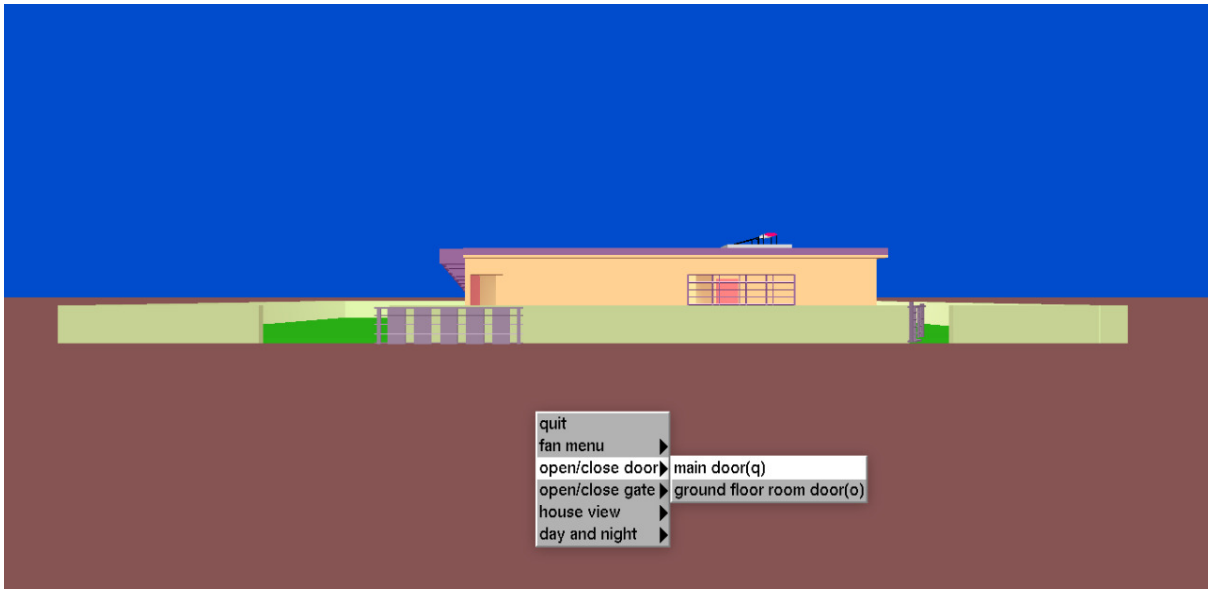


Fig 4.7: Main door and Gates are opened.

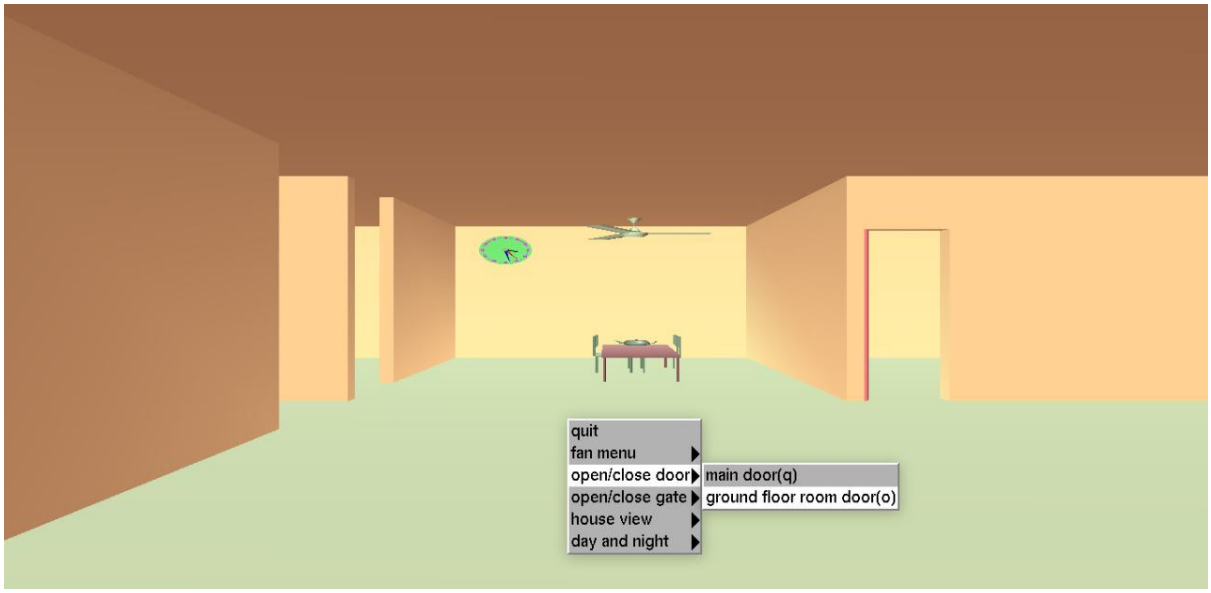


Fig 4.8: Inside View of the House.

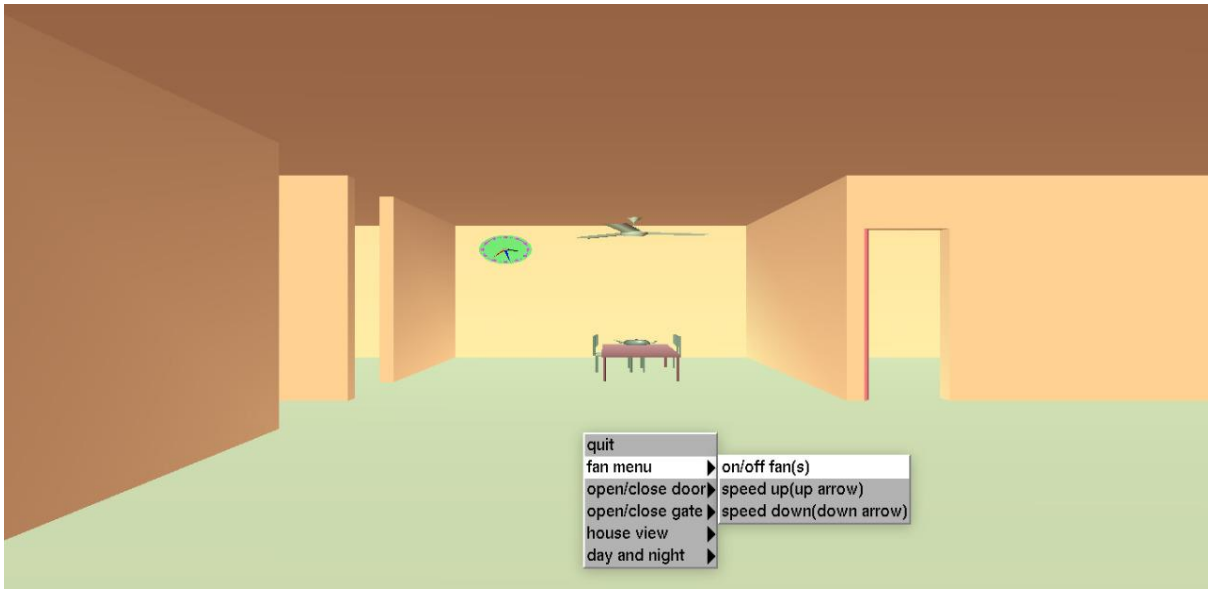


Fig 4.9: Fan switch Options.

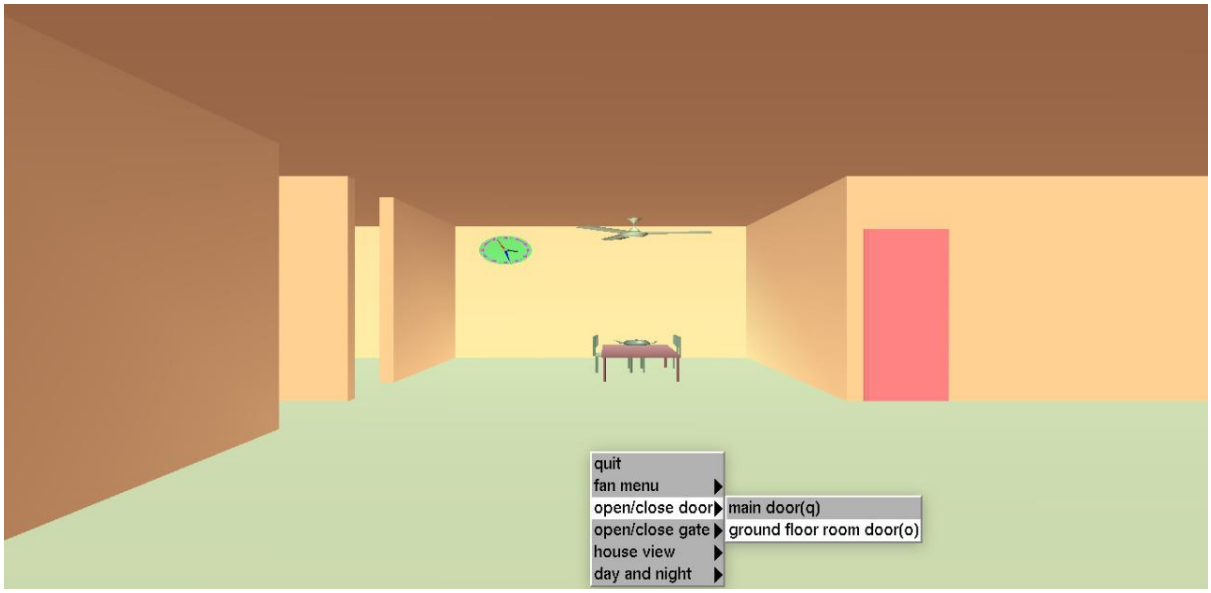


Fig 4.10: Room door.

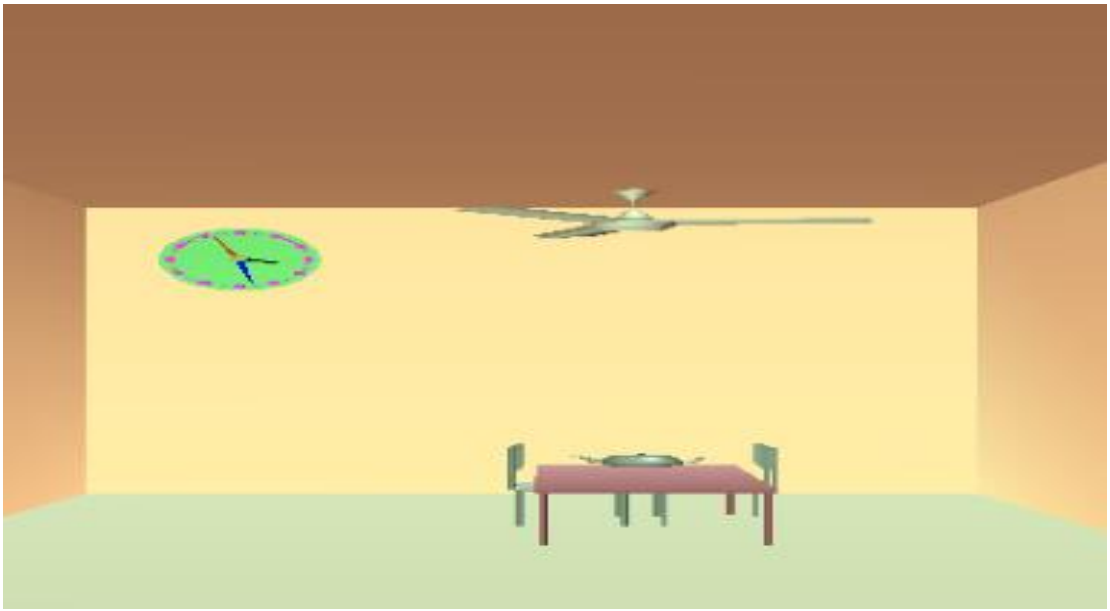


Fig 4.11: Objects present inside the House.

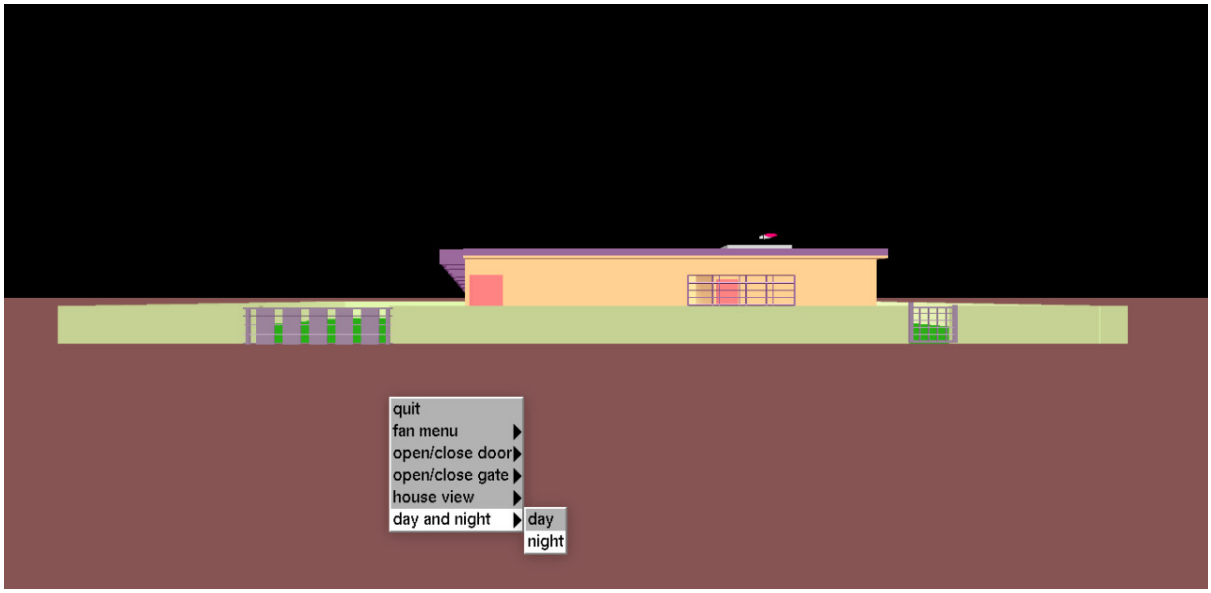


Fig 4.12: Night View.

Chapter 5

CONCLUSIONS AND FUTURE WORK

The 3D House has been implemented under Linux Operating System and has been found to provide ease of use and manipulation to the user. The 3D house created for the Linux 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

FUTURE WORK :

These are the features that are planned to be supported in the future

* Support for multiple canvases

* Support for pattern filling

* Support for 3d transformations

* Support for transparency of layers

REFERENCES

- ✓ Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, Pearson Education,2011
- ✓ Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5 th edition. Pearson Education, 2008
- ✓ James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: pearson education
- ✓ https://www.opengl.org/
- ✓ https://learnopengl.com/Getting-started/OpenGL
- ✓ https://en.wikipedia.org/wiki/Computer_graphics