VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAVI, KARNATAKA -590 018

A Mini-Project Report on

"3-D ROOM"

Submitted in partial fulfillment for the Computer Graphics Laboratory with Mini-Project (18CSL67) course of Sixth Semester of Bachelor of Engineering in Computer Science & Engineering during the academic year 2021-22.

By

Team Code: **CGP2022-C16**

Mohammed Roushan **4MH19CS126**

Prajwal Y P **4MH19CS071**

: Under the Guidance of :

Dr. Lokesh M R

Professor

Department of CS&E

MIT Mysore

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, S.R. Patna Taluk, Mandya Dist-571477.

Accredited By:

2021-22

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

~~ CERTIFICATE ~~

*Certified that the mini-project work entitled "**3-D ROOM**" is a bonafide work carried out by [**Mohammed Roushan**] (4MH19CS126) & [**Prajwal YP**] (4MH19CS071) for the Computer Graphics Laboratory with Mini-Project (18CSL67) of Sixth Semester in Computer Science & Engineering under Visvesvaraya Technological University, Belgavi during academic year 2021-22.*

It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The report has been approved as it satisfies the course requirements.

_____	_____
Signature of Guide	Signature of the HOD
Dr. Lokesh M R	**Dr. Shivamurthy R C**
Professor	Professor & HOD
Dept. of CS&E	Dept. of CS&E
MIT Mysore	MIT Mysore

External viva

Name of the Examiners	**Signature with date**

1)……

2) ……

~~~~ ACKNOWLEDGEMENT ~~~~

It is the time to acknowledge all those who have extended their guidance, inspiration, and their wholehearted co-operation all along our project work.

We are also grateful to **Dr. B G Naresh Kumar**, principal, MIT Mysore and **Dr. Shivamurthy R C**, HOD, CS&E, MIT Mysore for having provided us academic environment which nurtured our practical skills contributing to the success of our project.

We wish to place a deep sense of gratitude to all Teaching and Non-Teaching staffs of Computer Science and Engineering Department for whole-hearted guidance and constant support without which Endeavour would not have been possible.

Our gratitude will not be complete without thanking our parents and our friends, who have been a constant source of support and aspirations

Mohammed Roushan

Prajwal Y P

~~~ ABSTRACT ~~~

- A Computer Graphics project based on the concept of a “3D Room”. This project implements the view of a 3D Environment both inside and outside of the Room.

- There are 6 components that are placed inside the room they are a Sliding Door, Fan, Table, Teapot, Decorations, and Cup placed on the table.The outside of the house is a Empty Space. The room is surrounded by a compound wall.

- The APIs that are used in implementing these components are glutSolidCube(), glutSolidTeapot(), glutSoildCone()… etc An animation has been implemented which shows the rotation of the fan as well as the decorations inside the room .

- Keyboard Keys have been provided to modify the various features such as opening the door,regulating the fan speed and controlling the view point of camera etc.

.

- This project implements both the Orthographic and Perspective views. We have tried our best to implement best logic to program this 3-dimensional Room Space using GLUT library function.

~~~~~ CONTENTS ~~~~~

CHAPTER - 1

INTRODUCTION

1.1 Aim

The main aim of the project is to illustrate the concepts and usage of pre-built functions like Rotating, translating and scaling in OpenGL, and we have used input like keyboard to interact with program. It is focused specifically on helping the Civil engineers to visualize the interior or exterior perspective of any building that is under construction and get proper requirement.

1.2 Overview

This project is about modelling the "3-D Wall" in 3-Dimentional environment Space. In a empty space a 3-D room is created which consists of a sliding Doors. When entering inside the room we will be able to visualize some of the household themes such as a Table (with yellow legs and purple top), A teapot and mug on top of the table (with yellow and blue color respectively), a fan(red in color) attached to roof with rotational functionality, and finally a rotating decahedron is used as decorative objects which is placed at each corners of the room.

1.3 Outcome

Computer graphics are graphics created using computers and, more generally, there presentation and manipulation of image data by a computer.Using this special ability of our developed computer graphics project now Civil Engineers are capable helping people to visualize their required house design, which in turn leads to customer satisfaction. Also this will help in Building good futuristic building as we can visualize its interior design in 3-D in prior.

CHAPTER – 2

DESIGN AND IMPLEMENTATION

2.1 Algorithm

Step 1: START.

Step 2: Using the function glutSoildCube(), construct the walls in a free space,translate it and scale it as per the requirements.

Step 3: Again by using the function glutSolidCube(), construct the table inside a room ,translate it and scale it as per the requirements on the floor.

Step 4: Using the function glutSolidTeapot(), construct the teapot on top of the table and scale it to the best fit on the table.

Step 5: Again by using the function glutSolidCube(), construct a mug,translate it and scale it as per the requirements to the best fit on the table.

Step 6: Again by using the function glutSolidCube(), construct a fan,translate it and scale it as per the requirements.

Step 7: Now provide user input in order to control the functions such as rotate,zoom and many more with the help of keyboard.

7.1 -> to see the front view -> d;

7.2 -> to rotate left -> l;

7.3 -> to rotate right -> r;

7.4 -> to rotate top -> t;

7.5 -> to rotate bottom -> b;

7.6 -> to open the door -> o;

7.7 -> to close the door -> c;

7.8 -> to start/speed-up the fan -> S;

7.9 -> to stop/speed-down the fan -> s;

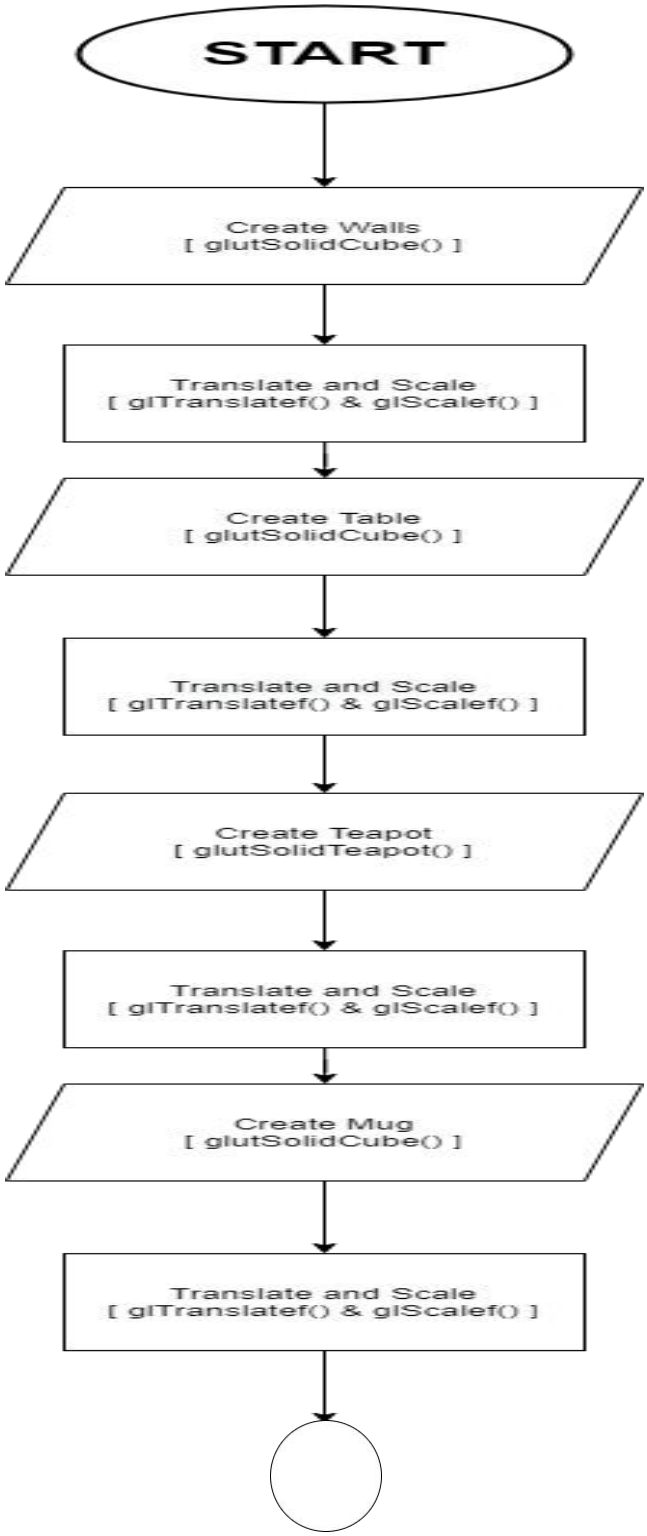
7.10 -> to zoom-in -> Z;

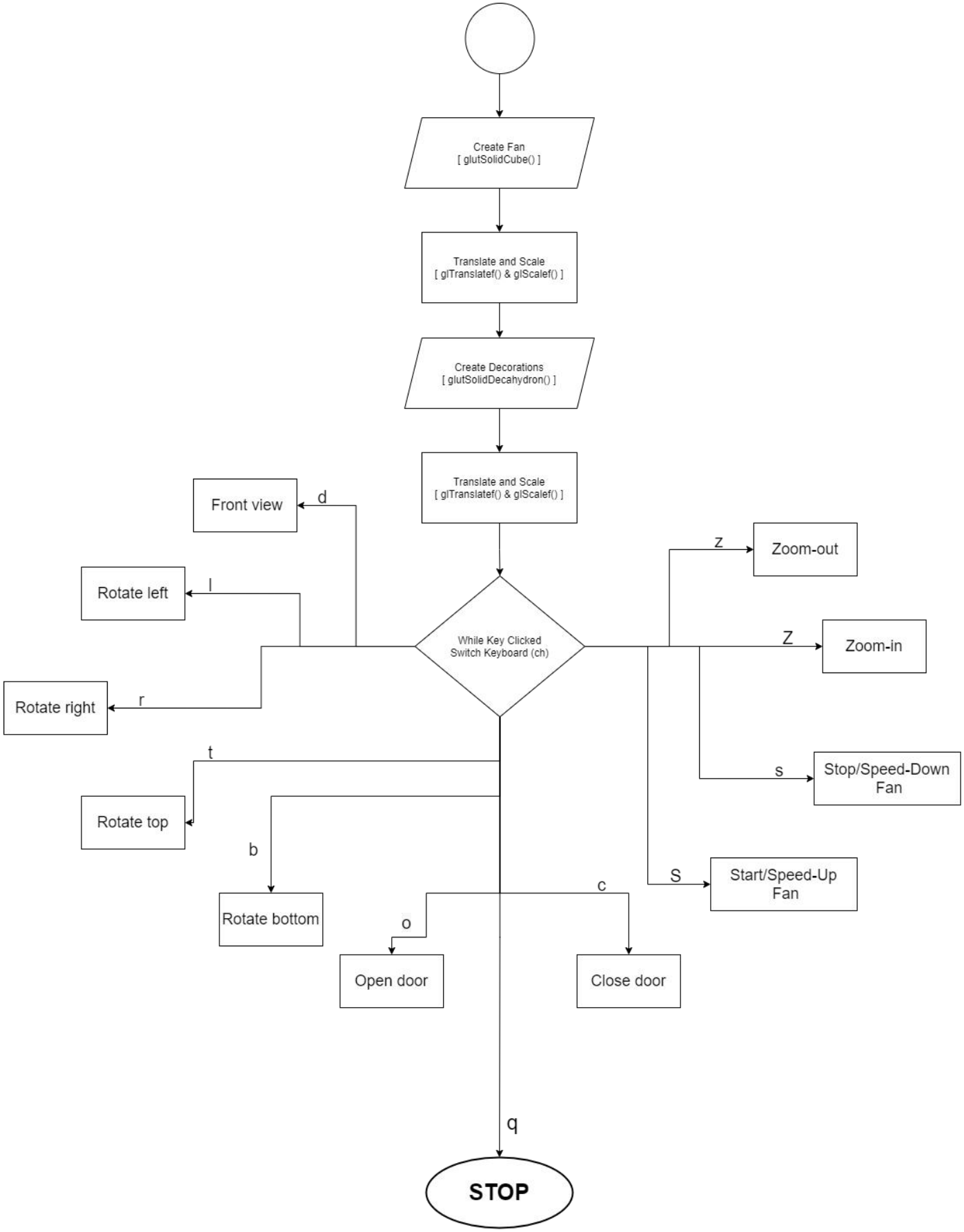
7.11 -> to zoom-out -> z;

7.12 -> to quit -> q;

Step 8: STOP.

2.2 Flowchart




Create Fan
[glutSolidCube()]
Translate and Scale
[glTranslatef() & glScalef()]
Create Decorations
[glutSolidDecahydron()]
Translate and Scale
[glTranslatef() & glScalef()]
While Key Clicked
Switch Keyboard (ch)
d
Front view
l
Rotate left
r
Rotate right
t
Rotate top
b
Rotate bottom
o
Open door
c
Close door
z
Zoom-out
Z
Zoom-in
s
Stop/Speed-Down Fan
S
Start/Speed-Up Fan
q
STOP

2.3 OpenGL API's Used with Description

main(): Execution of any program always starts with main function irrespective of where it is written in a program.

glutSwapBuffers(void): Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer. The contents of the back buffer then become undefined. The update typically takes place during the vertical retrace of the monito r, rather than immediately after glutSwapBuffers is called.

glutPostRedisplay(): marks the plane of current window as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display callback opportunity generates only a single redisplay callback.

glutInit(&argc,char ** argv): glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized.

glutInitDisplayMode(unsigned int mode): The initial display mode is used when creating toplevel windows, sub windows, and overlays to determine the OpenGL display mode for the to-becreated window or overlay. GLUT_RGB specifies the structure of each pixel. GLUT_DEPTH is a buffer to hold the depth of each pixel.

glutCreateWindow(char *title): The glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name.

glutReshapeFunc(): glutReshapeFunc sets the reshape callback for the current window. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay for the window is established. The width and height parameters of the callback specify the new window size in pixels. Before the callback, the current window is set to the window that has been reshaped.

glutMainLoop(): glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

glFlush(): Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. glFlush empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

glClear(GLbitfield): glClear sets the bit plane area of the window to values previously selected by glClearColor, glClearIndex, glClearDepth, glClearStencil, and glClearAccum. Multiple color buffers can be cleared simultaneously by selecting more than one buffer at a time using glDrawBuffer.

glVertex2f(x,y): This function is used to draw a vertex at location x,y.

glClearColor(R,G,B,A): glClearColor specifies the red, green, blue, and alpha values used by glClear to clear the color buffers. Values specified by glClearColor are clamped to the range 0 to 1 .

glMaterialfv(GLenum,GLenum,GLfloat *): glMaterial assigns values to material parameters. There are two matched sets of material parameters. One, the front- facing set, is used to shade points, lines, bitmaps, and all polygons (when two-sided lighting is disabled), or just front-facing polygons (when two-sided lighting is enabled).

glutSolidCube(size): glutSolidCube and glutWireCube render a solid or wireframe cube respectively. The cube is centered at the modeling coordinate's origin with sides of length size.

glutDisplayFunc(void (*func)()): glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and (if no overlay display callback is registered) the layer in use is set to the normal plane. The display callback is called with no parameters.

glutKeyboardFunc(void (*func)(unsigned char key, int x, int y)): glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character.

glMatrixMode(GLenum) mode: Specifies which matrix stack is the target for subsequent matrix operations. Values accepted are: GL_MODELVIEW, GL_PROJECTION.The initial value is GL_MODELVIEW. Additionally, if the ARB tension extension is supported, GL_COLOR is also accepted.

gluPerspective(fovy, aspect, near, far): gluPerspective specifies a viewing frustum into the world coordinate system. In general, the aspect ratio in gluPerspective should match the aspect ratio of the associated viewport. For example, aspect = 2.0means the viewer's angle of view is twice as wide in x as it is in y.

glOrtho(left, right, top, bottom, Znear, Zfar): glOrtho describes a transformation that produces a parallel projection. The current matrix is multiplied by this matrix and the result replaces the current matrix.

User - defined functions :

room() : This API used to construct a 3-Dimentional room which consists of 4 walls. Each wall is put in separate matrix so that it will not have other objects side effects.It uses glutSolidCube() function to build wall by scaling down x-axis and z-axis to look like a wall and ratation to place walls at top, bottom, left and right. The front door is of two solids that slide sideways to open the room.

teapot() : This API used to construct a 3-Dimentional Teapot which is kept on the table. It is put in separate matrix so that it will not have other objects side effects. It uses glutSolidTeapot() function to build teapot by scaling down x-axis, y-axis and z-axis to look like a realistic fitting Object inside the room.

table() : This API used to construct a 3-Dimentional table which consists of 4 legs and a top. Each leg and top are put in separate matrix so that they will not have other objects side effects.Each matrix uses glutSolidCube() function to build table by scaling down x-axis and z-axis to look like a leg, scaling down y axis to look like a top and rotation as well as translation to place legs at top, bottom, left and right.

cup() : This API used to construct a 3-Dimentional cup which consists of cover and holder. Each cover and holder parts are put in separate matrix so that they will not have other objects side effects. Each matrix uses glutSolidCube() function to build cup by scaling down x-axis, y-axis and z-axis where ever necessary to look like a cup, and rotation as well as translation to place the pieces of the cup-cover and cup-holder at their appropriate places.

decoration() : This API used to construct a 3-Dimentional Rotating Dodecahedron which which looks attractive to users. Each Dodecahedron are put in separate matrix so that they will not have other objects side effects. Each matrix uses glutSolidDodecahedron() function to build by scaling down x-axis, y-axis and z-axis where ever necessary to fit into the appropriate places of the room .

fan() : This API used to construct a 3-Dimentional room which consists of 4 3-parts. Each part is put in separate matrix. Some uses glutSolidCube() and one glutSolidDodecahedron() function to build fan by scaling down x-axis and z-axis to look realistic and rotation applied when ever the key ‘S’ or ‘s’ is pressed.

2.4 Source Code

```
#include<GL/glu.h>

#include<GL/glut.h>

#include<windows.h>


GLfloat T=0;

GLfloat Pos[4]={0,0,2,1};

GLfloat Col[4]={1,1,1,1};

GLfloat Col1[4]={1,0,0,1};

GLfloat Col2[4]={1,1,0,1};

GLfloat Col3[4]={1,0,1,1};

GLfloat Col4[4]={1,.09,0,1};

GLfloat Col5[4]={0,0,1,1};

GLfloat Col6[4]={0,1,0,1};

GLfloat Col7[4]={.241,.235,.156,1};

float Z=5,S=0,s=0,D=0;

int ld=1,rd=-1,x=0,y=0,z=0;


void Spin()

{

   if(s==360)

      s=1;

   S+=s;

   D+=0.3;

   glutPostRedisplay();

}
```

```
void key(unsigned char ch,int x1,int y1)
{
   switch(ch)
   {
      case 'o': ld=3;rd=-3; break;
      case 'c': ld=1;rd=-1; break;
      case 'd': x=0;y=0;z=0;T = 0; break;
      case 'r': x=0;y=1;z=0;T -= 1; break;
      case 'l': x=0;y=1;z=0;T += 1; break;
      case 't': x=1;y=0;z=0;T += 1; break;
      case 'b': x=1;y=0;z=0;T -= 1; break;
      case 'z': Z+=.5; break;
      case 'Z': Z-=.5; break;
      case 'S':
         if(s<0.8)
            s+=.2;
         break;
      case 's':
         if(s>0)
            s-=.1;
         break;
   }
   glutPostRedisplay();
}

void cup()
{
```

```
glPushAttrib(GL_ALL_ATTRIB_BITS);
    glTranslatef(.28,.2,0);
    glScalef(0.9,0.9,0.9);
    glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,Col5);
    glLightfv(GL_LIGHT0,GL_AMBIENT_AND_DIFFUSE,Col5);
      glPushMatrix();
      glTranslatef(0.13,0,0);
      glScalef(0.05,0.3,0.3);
      glutSolidCube(1);
      glPopMatrix();

      glPushMatrix();
      glTranslatef(-0.13,0,0);
      glScalef(0.05,0.3,0.3);
      glutSolidCube(1);
      glPopMatrix();

      glPushMatrix();
      glTranslatef(0,-0.13,0);
      glScalef(0.3,0.05,0.3);
      glutSolidCube(1);
      glPopMatrix();

      glPushMatrix();
      glTranslatef(0,0,-0.13);
      glScalef(0.3,0.3,0.05);
      glutSolidCube(1);
```

```
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0,0,0.13);
    glScalef(0.3,0.3,0.05);
    glutSolidCube(1);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.33,0,0);
    glScalef(0.02,0.2,0.2);
    glutSolidCube(1);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.24,0.1,0);
    glScalef(0.2,0.02,0.2);
    glutSolidCube(1);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.24,-0.1,0);
    glScalef(0.2,0.02,0.2);
    glutSolidCube(1);
    glPopMatrix();

  glPopAttrib();
```

```
}

void table()
{
   glPushAttrib(GL_ALL_ATTRIB_BITS);
      glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,Col2);
      glTranslatef(0,-.6,0);

         glPushMatrix();
         glTranslatef(-.45,-.48,-.4);
         glScalef(1,10,1);
         glutSolidCube(.1);
         glPopMatrix();

         glPushMatrix();
         glTranslatef(.45,-.48,-.4);
         glScalef(1,10,1);
         glutSolidCube(.1);
         glPopMatrix();

         glPushMatrix();
         glTranslatef(-.45,-.48,.4);
         glScalef(1,10,1);
         glutSolidCube(.1);
         glPopMatrix();

         glPushMatrix();
```

```
        glTranslatef(.45,-.48,.4);
        glScalef(1,10,1);
        glutSolidCube(.1);
        glPopMatrix();

        glPushMatrix();
        glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,Col3);
        glTranslatef(0,0,0);
        glScalef(10,1,10);
        glutSolidCube(.1);
        glPopMatrix();

    glPopAttrib();
}

void teapot()
{
  glPushAttrib(GL_ALL_ATTRIB_BITS);
    glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,Col2);
    glLightfv(GL_LIGHT0,GL_DIFFUSE,Col2);
    glPushMatrix();
    glTranslatef(-.2,-.4,0);
    glScalef(.9,.9,.9);
    glutSolidTeapot(.2);
    glPopMatrix();
    glPopAttrib();
}
```

```
void room()
{
   glRotatef(T,x,y,z);
      glPushMatrix();
      glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,Col1);
      glTranslatef(0,1.6,0);
      glScalef(4,0.2,4);
      glutSolidCube(1);
      glPopMatrix();

      glPushMatrix();
      glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,Col6);
      glTranslatef(0,-1.5,0);
      glScalef(4,0.2,4);
      glutSolidCube(1);
      glPopMatrix();

      glPushMatrix();
      glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,Col5);
      glTranslatef(2,0,0);
      glScalef(0.2,3,4);
      glutSolidCube(1);
      glPopMatrix();

      glPushMatrix();
      glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,Col3);
```

```
glTranslatef(-2,0,0);
glScalef(0.2,3,4);
glutSolidCube(1);
glPopMatrix();

glPushAttrib(GL_ALL_ATTRIB_BITS);
glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,Col4);
   glPushMatrix();
   glTranslatef(0,0,-2);
   glScalef(4,3,0.2);
   glutSolidCube(1);
   glPopMatrix();
glPopAttrib();

glPushMatrix();
glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,Col7);
glTranslatef(ld,0,2);
glScalef(2,3,0.2);
glutSolidCube(1);
glPopMatrix();

glPushMatrix();
glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,Col7);
glTranslatef(rd,0,2);
glScalef(2,3,0.2);
glutSolidCube(1);
glPopMatrix();
```

```
}

void fan()
{
   glPushAttrib(GL_ALL_ATTRIB_BITS);
   glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,Col1);
   glTranslatef(-.1,1.2,0);
   glRotatef(S,0,1,0);
      glPushMatrix();
         glTranslatef(0,0,0);
         glScalef(1,.1,.1);
         glutSolidCube(1);
      glPopMatrix();

      glPushMatrix();
         glTranslatef(0,0,0);
         //glScalef(.5,.1,.1);
         glutSolidSphere(.1,100,100);
      glPopMatrix();

      glPushMatrix();
         glTranslatef(0,0,0);
         glScalef(.1,.1,1);
         glutSolidCube(1);
      glPopMatrix();

      glPushMatrix();
```

```
        glTranslatef(0,0.2,0);
        glScalef(.1,0.5,.1);
        glutSolidCube(1);
    glPopMatrix();

  glPopAttrib();
}

void decoration()
{
  glPushAttrib(GL_ALL_ATTRIB_BITS);
  glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,Col6);
  glTranslatef(-0.3,0.47,0);

    glPushMatrix();
    glTranslatef(1.4,1,-1.5);
    glScalef(0.16,0.16,0.16);
    glRotatef(D,1,1,1);
      glutSolidDodecahedron();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-1.4,1,-1.5);
    glScalef(0.16,0.16,0.16);
    glRotatef(D,1,1,1);
      glutSolidDodecahedron();
    glPopMatrix();
```

```
glPushMatrix();
glTranslatef(1.4,-1,-1.5);
glScalef(0.16,0.16,0.16);
glRotatef(D,1,1,1);
   glutSolidDodecahedron();
glPopMatrix();

glPushMatrix();
glTranslatef(-1.4,-1,-1.5);
glScalef(0.16,0.16,0.16);
glRotatef(D,1,1,1);
   glutSolidDodecahedron();
glPopMatrix();

glPushMatrix();
glTranslatef(1.4,1,1.5);
glScalef(0.16,0.16,0.16);
glRotatef(D,1,1,1);
   glutSolidDodecahedron();
glPopMatrix();

glPushMatrix();
glTranslatef(-1.4,1,1.5);
glScalef(0.16,0.16,0.16);
glRotatef(D,1,1,1);
   glutSolidDodecahedron();
```

```
        glPopMatrix();

        glPushMatrix();
        glTranslatef(1.4,-1,1.5);
        glScalef(0.16,0.16,0.16);
        glRotatef(D,1,1,1);
            glutSolidDodecahedron();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-1.4,-1,1.5);
        glScalef(0.16,0.16,0.16);
        glRotatef(D,1,1,1);
            glutSolidDodecahedron();
        glPopMatrix();

    glPopAttrib();
}

void fandeco()
{
    glPushAttrib(GL_ALL_ATTRIB_BITS);
    glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,Col2);
        glPushMatrix();
            glScalef(0.07,0.07,0.07);
            glutSolidDodecahedron();
        glPopMatrix();
```

```
   glPopAttrib();
}

void Draw()
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();

   glLightfv(GL_LIGHT0,GL_POSITION,Pos);
   glLightfv(GL_LIGHT0,GL_DIFFUSE,Col);
   gluLookAt(0,0,Z,0,0,0,0,1,0);

   glPushAttrib(GL_ALL_ATTRIB_BITS);
   glScalef(0.8,0.8,0.8);
      room();
      teapot();
      table();
      cup();
      decoration();
      fan();
      fandeco();
   glPopAttrib();

   glutSwapBuffers();
}
```

```
void myInit()
{
   glEnable(GL_DEPTH_TEST);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glFrustum(-1,1,-1,1,1,15);
   glMatrixMode(GL_MODELVIEW);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
}

int main(int c,char **v)
{
   glutInit(&c,v);
   glutInitWindowSize(500,500);
   glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);

   glutCreateWindow("3-D Room");
   myInit();
   glutDisplayFunc(Draw);
   glutIdleFunc(Spin);
   glutKeyboardFunc(key);


   glutMainLoop();
   return 0;
}
```

CHAPTER - 3

RESULT ANALYSIS

3.1 Snap Shots :

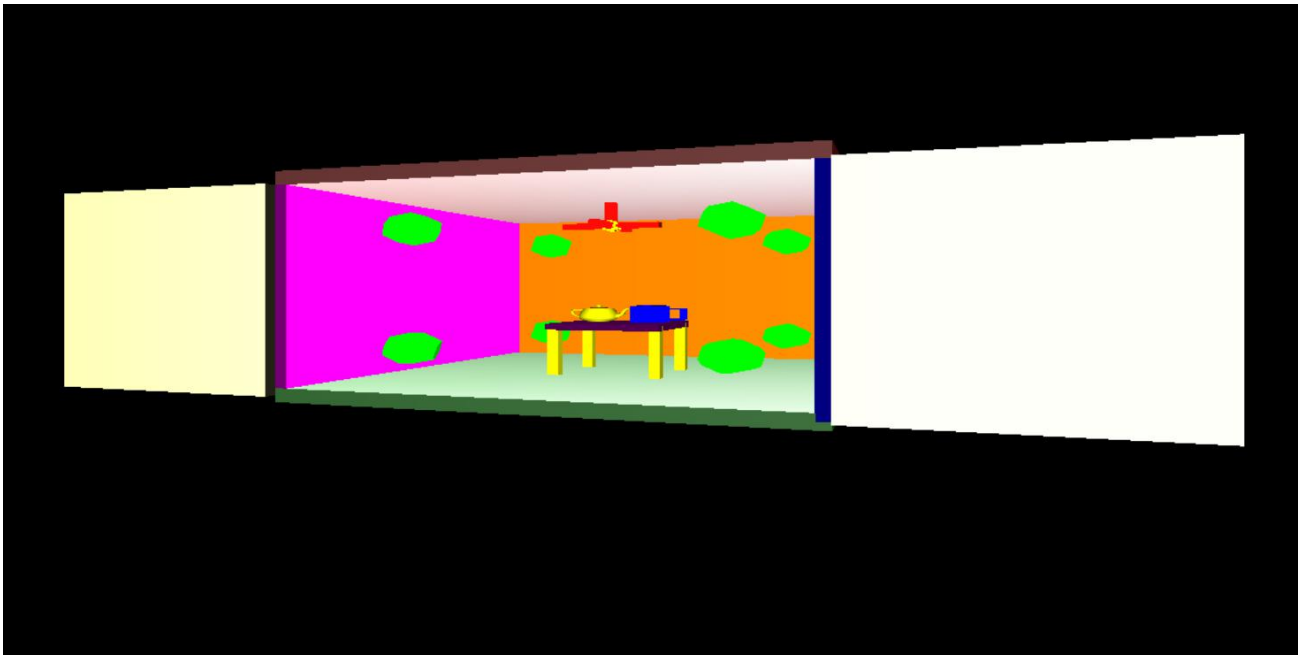


Fig-3.1.1 Side View

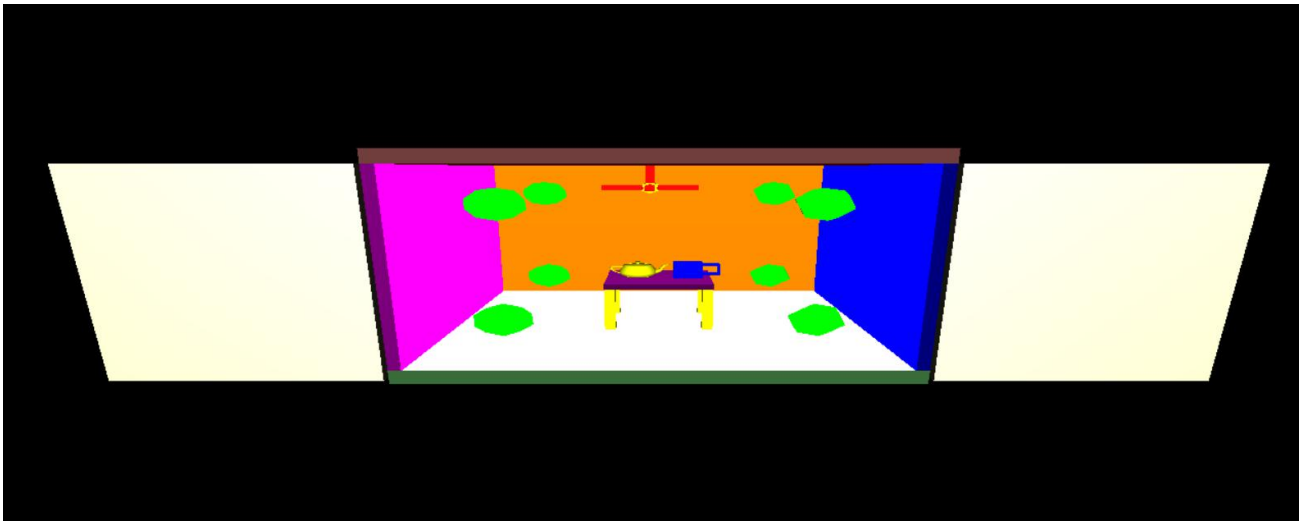


Fig-3.1.2 Incline View

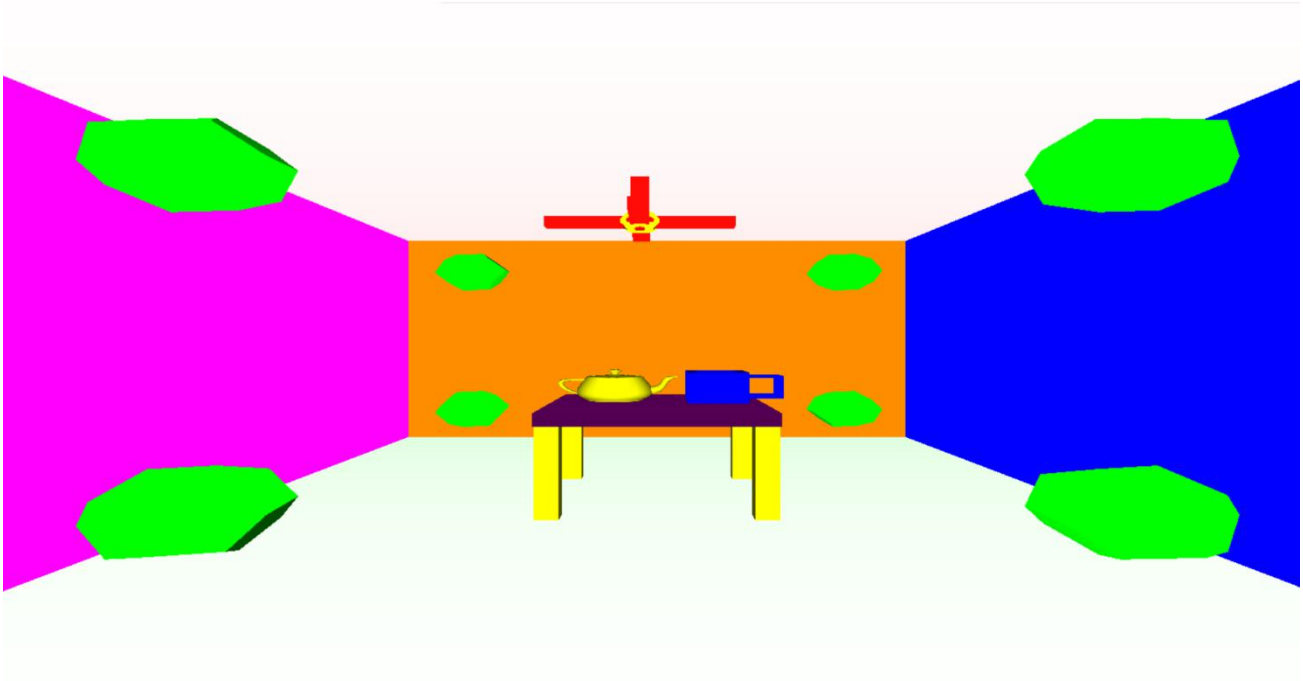


Fig-3.1.3 Inside View

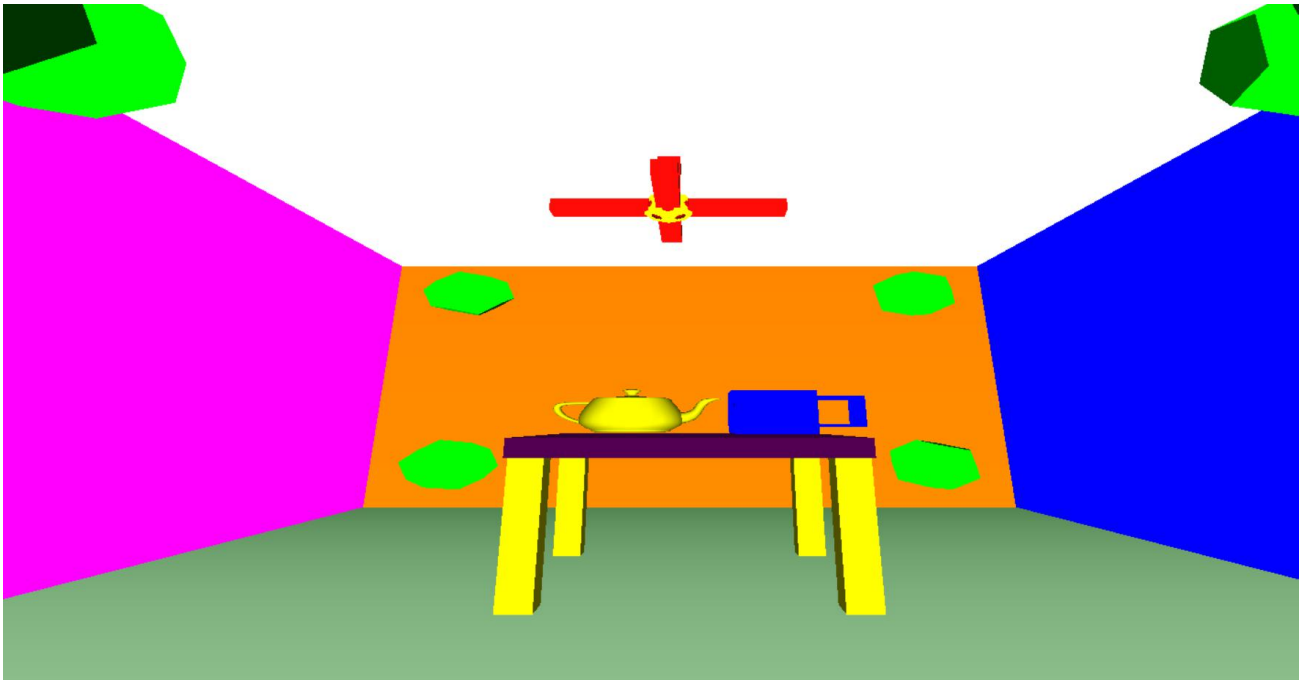


Fig-3.1.4 Decline View

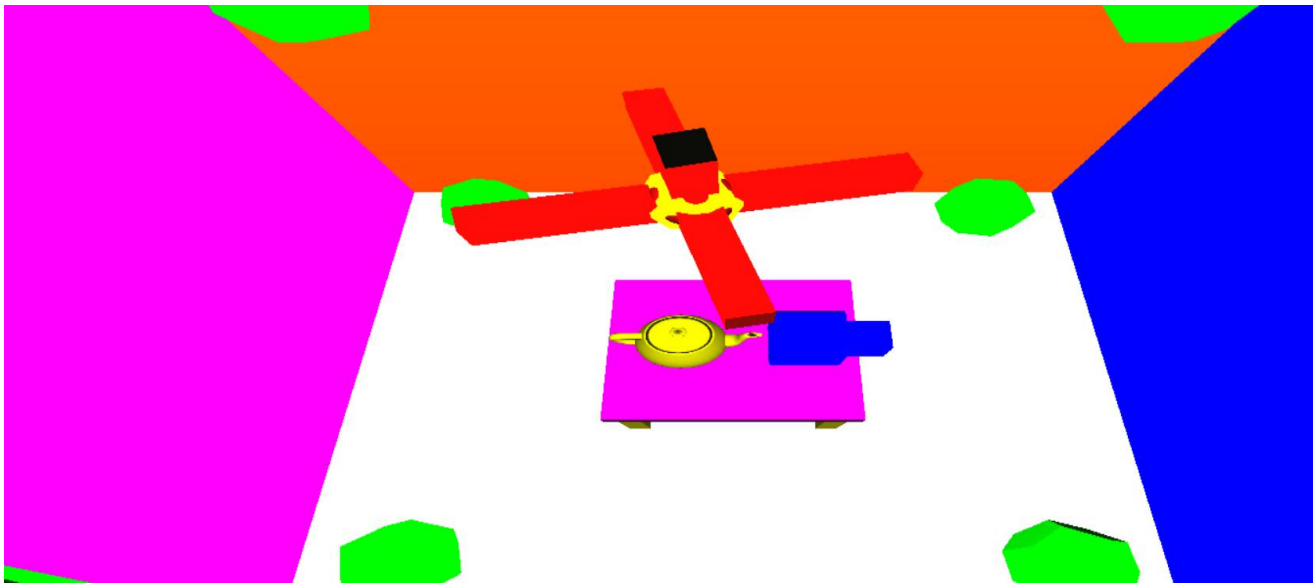


Fig-3.1.5 Top View

3.2 Discussion

An attempt has been made to develop an OpenGL graphics package, which meets the necessary requirements of the user successfully. It enables us to learn about the basic concepts in OpenGL graphics and know standard library graphics functions and also to explore some other functions.

OpenGL graphics is a huge library which consist of numerous functions. These function have been used creatively to build the programs which may be to draw figures of various. This project has given us an insight into the use of computer graphics. As we have shapes at lower level or to stimulate any real thing, animation etc at higher level. 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 game programmers.

Each of the objects in this project is brought into visibility by the help of lightning. Also our objects color is assigned with the help of glMaterialfv() function. Rotation of the fan is done with the help of the glKeyboardFunc with ‘S’ and ‘s’ keys.The decoration is done using glutSolidDecahydron() function which keeps rotating all the time. Camera movement is achieved with the help of global variables called ‘X’, ‘Y’ and ‘Z’ which are manipulated by the keys ‘l’, ’r’, ‘t’ and ‘b’ keys.

We can now converse with a certain degree of confidence about Graphics in OpenGL and finally we have successfully completed the implementation of this project using OpenGL.

CHAPTER - 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

After the completion of this project we came to know how to work with Microsoft visual studio and how we can implement a project using an open source OpenGL tool kit.

By implementing a project using OpenGL, we came to know how to use the functions like lighting, rotation, translation and scaling. With the completion of this project we have achieved a sense of happiness and we want to thank all those who helped us directly or indirectly to make this idea come true.

4.2 Future Enhancement

We are trying to visualize a clock aligned to system time, also working to extend the room and construct full home. Also we are planning on inserting some mild gardening outside the home. Also we are trying to add table lamp to make bedroom more attractive along with cot.

If possible we are trying to add seasons, by placing sun with increased brightness on pressing the key ‘n’ and moon with decreased brightness on pressing key ‘N’. And also rain by applying translation by the help of the key ‘R’ and stop it using ‘r’, and normal season by the key ‘n’.

CHAPTER - 5

BIBLIOGRAPHY

Text Books:

1 -- Edward angel: Interactive computer graphics A TOP DOWN Approach with OpenGL, 2nd edition, Addison - Wesley, 2000.

2 -- F.S. Hill, Jr.: computer graphics using OpenGL, 2nd edition, Pearson education, 2001.

Websites:

- https://www.youtube.com/c/LearningBits1024/playlists/
- https://www.youtube.com/playlist?list=PLf8EAse10pKcZ5AUN2Mmq8beu4IZ-AhRm
- https://stackoverflow.com/
- https://codeproject.com/