**Name- Parismita Devi**

**USN- 1GA18CS103**

**Project Title- 3D House**

***Execution Steps-***

1. Create an empty C++ project in Visual Studio with an item with .cpp extension.
2. Copy the code given below into the item in Visual Studio 2019.
3. Build Solution (Project→Build solution).
4. Once the project is built successfully, click on Windows Debugger and the project will start debugging.
5. 3D House project will be displayed, to know how to operate it the steps are displayed in the terminal window.
6. Or follow the following keys on your keyboard:
    - Press F: Front-view of House
    - Press T: Top-view of House
    - Press B: Back-view of House
    - Press 1: Go zoom-in and enter the House
    - Press 2: Zoom-out and exit the House
    - Press G: Open and close main gate of the House
    - Press M: Open and close main door of the House
    - Press I: Interior of the House
    - Press O: Open and close bedroom door
    - Press S: Start and Stop fan spinning

***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;
}
```