```
/**
 * Final Project Computer Graphics
 *
 * Written by Parshwa Shah (110021970)
 *
 * Version 1.0
 *
 * Draw 3D scenes like bus, trees, road, sky,
 * Aeroplane, clouds, sun and more
 *
 *  Instructions/Help to run the project
 *  Use of MOUSE
 *      I) PRESS RIGHT BUTTON FOR MENU
 *  Use of KEYBOARD in bus model mode
 *      I) X-Y-Z KEYS FOR CORRESPONDING ROTATION
 *      II) A-S-Q Bus CUSTOM SIZE SELECTION
 *      III) U-F FOR CAMERA VIEW SETTINGS
 *      IV) USE LEFT ARROW(<-) AND RIGHT ARROW(->) TO MOVE Bus
 *      V) ESCAPE TO EXIT
 *  Use of KEYBOARD in bus model mode
 *      I) Y KEY FOR Y-AXIS ROTATION
 *      II) U-F FOR CAMERA VIEW SETTINGS
 *
 */

#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <iostream>

#define ESCAPE 27
using namespace std;

void init();
void DrawDifferentScenes();
void keyboard(GLubyte key, GLint x, GLint y);
void specialFunction(int key,int keyx,int keyy);
void timer(int);
void cloudtimer(int);
void bustimer(int);

GLint window;
GLint Xsize=1000;
GLint Ysize=800;
float i,theta;
GLint nml=0,day=1;

char name3[]="110021970 Shah Parshwa 3D Animation";

GLfloat xTranslation=0.0,yTranslation=0.0,zTranslation=0.0,xw=0.0;   /*
x,y,z translation */
GLfloat xScale=1.0,yScale=1.0,zScale=1.0;

GLfloat xangle=0.0,yangle=0.0,zangle=0.0,angle=0.0;   /* axis angles */
GLubyte red=220,green=149,blue=37; // dark yellow
GLubyte rWindow=118,gWindow=215,bWindow=234;

int count=1,flg=1;
int view=0;
int flag1=1; //roads and surrounding
```

```
int aflag=1; //to switch bus driving mode
int flag2=0; //to switch fog effect
int wheelflag=1,busflag=1;
int backgroundflag=0;
GLUquadricObj *t;

float aeroplaneXinc=0,aeroplaneYinc=0,cloudXinc=0;
int aeroplaneflag = 1, cloudflag = 1;
float random1,random2;
float backr=0,backg=0.9,backb=0.9;

GLvoid Transform(GLfloat screenWidth, GLfloat screenHeight)
{
    glViewport(0, 0, screenWidth, screenHeight);             /* Set the
viewing port of the screen*/
    glMatrixMode(GL_PROJECTION);                 /* projection matrix is
selected*/
    glLoadIdentity();               /* Projection Matrix gets reseted*/
    gluPerspective(45.0,screenWidth/screenHeight,0.1,100.0);  /* Calculate
The Aspect Ratio Of The Window */
    glMatrixMode(GL_MODELVIEW);                  /* Switch back to the
model view matrix */
}

/** It will initialize all the parameters required for the window and
program to run*/
GLvoid InitGL(GLfloat Width, GLfloat Height)
{
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glLineWidth(2.0);
    Transform( Width, Height );
    t=gluNewQuadric();
    gluQuadricDrawStyle(t, GLU_FILL);

// Create light components
    GLfloat ambientLight[] = { 0.2f, 0.2f, 0.2f, 1.0f };
    GLfloat diffuseLight[] = { 0.8f, 0.8f, 0.8, 1.0f };
    GLfloat specularLight[] = { 0.5f, 0.5f, 0.5f, 1.0f };
    GLfloat position[] = { 1.5f, 1.0f, 4.0f, 1.0f };

// Assign created components to GL_LIGHT0
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);
    glLightfv(GL_LIGHT0, GL_POSITION, position);
}

/**
    My initialization method for specifying properties of window
*/
void init()
{
    glClearColor(0,0,0,0);
    glPointSize(5.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0,900.0,0.0,600.0,50.0,-50.0);
    glutPostRedisplay();        // request redisplay
}

/** This method is used to display String*/
void display_string(int x, int y, char *string, int font)
{
    int len,i;
    glColor3f(1,1,1);
    glRasterPos2f(x, y);
```

```
    len = (int) strlen(string);
    for (i = 0; i < len; i++)
    {
        if(font==1)
            glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]);
        if(font==2)
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,string[i]);
        if(font==3)
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12,string[i]);
        if(font==4)
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10,string[i]);
    }
}

void displayTextContent(void)
{
    glClearColor(0.063,0.47,0.588,1.0);
    display_string(380,560,"University of Windsor",1);
    display_string(350,520,"Computer Graphics CS 8520",1);
    display_string(300,480,name3,1);
    display_string(10,450,"Instructions/Help to run the project",2);
    display_string(40,420,"Use of MOUSE",2);
    display_string(70,380,"I) PRESS RIGHT BUTTON FOR MENU",3);
    display_string(40,340,"Use of KEYBOARD in bus MODEL mode",2);
    display_string(70,310,"I) X-Y-Z KEYS FOR CORRESPONDING ROTATION",3);
    display_string(70,280,"II) A-S-Q Bus CUSTOM SIZE SELECTION",3);
    display_string(70,250,"III) U-F FOR CAMERA VIEW SETTINGS",3);
    display_string(70,220,"IV) USE LEFT ARROW(<-) AND RIGHT ARROW(->) TO
MOVE Bus",3);
    display_string(70,190,"V) ESCAPE TO EXIT",3);
    display_string(40,160,"Use of KEYBOARD in bus DRIVING mode",2);
    display_string(70,130,"I) Y KEY FOR Y-AXIS ROTATION",3);
    display_string(70,100,"II) U-F FOR CAMERA VIEW SETTINGS",3);
    display_string(330,60,"PRESS SPACE BAR TO ENTER",1);
    glutPostRedisplay();
    glutSwapBuffers();
}

/**
This method is used to draw all the different scenes and object.
Also this method manages entire output of the project
*/
void DrawDifferentScenes()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); /** Clear The
Screen And The Depth Buffer */
    //Display the first screen
    if(view==0)
    {
        init();
        displayTextContent();
    }
    else //Display the bus screen with all other object
    {
        if(count==1)
            InitGL(Xsize,Ysize);
        if(aflag==1)/* Initialize our window. */
            glClearColor(backr,backg,backb,1);
        else
            glClearColor(0.1,0.1,0.1,0);

        if(backgroundflag == 1)/* Initialize our window. */
            glClearColor(1,1,1,0);

        glPushMatrix();
        glLoadIdentity();
```

```
        glTranslatef(-1.0,-0.5,-3.5);
        glRotatef(xangle,1.0,0.0,0.0);
        glRotatef(yangle,0.0,1.0,0.0);
        glRotatef(zangle,0.0,0.0,1.0);
        glTranslatef(xTranslation,yTranslation,zTranslation);
        glScalef(xScale,yScale,zScale);
        glEnable(GL_COLOR_MATERIAL);
        glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);

        /** This portion displays the fog effect */
        if(flag2==1)
        {
            GLfloat fogcolour[4]= {1.0,1.0,1.0,1.0};

            glFogfv(GL_FOG_COLOR,fogcolour);             /* Define the
fog colour */
            glFogf(GL_FOG_DENSITY,0.1);                  /* How dense */
            glFogi(GL_FOG_MODE,GL_EXP);                  /* exponential
decay */
            glFogf(GL_FOG_START,3.0);                   /* Where wwe start
fogging */
            glFogf(GL_FOG_END,100.0);                      /* end */
            glHint(GL_FOG_HINT, GL_FASTEST);             /* compute per
vertex */
            glEnable(GL_FOG);/* ENABLE */
        }
        //It will disable the fog effect
        if(flag2==0)
            glDisable(GL_FOG);

        if(!aflag)
        {
            glBegin(GL_POINTS);
            glColor3f(1,1,1);
            glPointSize(200.0);
            int ccount=0;
            float x=10,y=10;
            while(ccount<20)
            {
                glVertex2f(x,y);
                x+=10;
                y+=10;
                if(y>Ysize)
                    y-=10;
                if(x>Xsize)
                    x-=10;
                ccount++;
            }
            glEnd();
        }

        /** headlight*/
        glColor3f(1.0,.75,0.0);
        glPointSize(30.0);
        glBegin(GL_POINTS);
        glVertex3f(0.2,0.3,0.3);
        glVertex3f(0.2,0.3,0.5);
        glEnd();
        glPointSize(200.0);

        glBegin(GL_QUADS);                /* OBJECT MODULE*/

        /* top of cube*/
        //************************FRONT BODY of the
bus************************************
        glColor3ub(red,green,blue);
```

```
        glVertex3f( 0.2, 0.5,0.6);
        glVertex3f(0.6, 0.5,0.6);
        glVertex3f(0.6, 0.5,0.2);
        glVertex3f( 0.2,0.5,0.2);

        /* bottom of cube*/
        glVertex3f( 0.2,0.2,0.6);
        glVertex3f(0.6,0.2,0.6);
        glVertex3f(0.6,0.2,0.2);
        glVertex3f( 0.2,0.2,0.2);

        /* front of cube*/
        glVertex3f( 0.2,0.2,0.6);
        glVertex3f(0.2, 0.4,0.6);
        glVertex3f(0.2,0.4,0.2);
        glVertex3f( 0.2,0.2,0.2);
        /* back of cube.*/
        glVertex3f(0.6,0.2,0.6);
        glVertex3f(0.6,0.5,0.6);
        glVertex3f(0.6,0.5,0.2);
        glVertex3f( 0.6,0.2,0.2);

        /* left of cube*/
        glVertex3f(0.2,0.2,0.6);
        glVertex3f(0.6,0.2,0.6);
        glVertex3f(0.6,0.5,0.6);
        glVertex3f(0.2,0.5,0.6);

        /* Right of cube */
        glVertex3f(0.2,0.2,0.2);
        glVertex3f( 0.6,0.2,0.2);
        glVertex3f( 0.6,0.5,0.2);
        glVertex3f( 0.2,0.5,0.2);
//*************************************************************************
****
        glVertex3f(0.2,0.65,0.6);
        glVertex3f(0.2,0.65,0.2);
        glVertex3f(2.1,0.65,0.2);        //top cover of the bus
        glVertex3f(2.1,0.65,0.6);
//*************************back guard*****************************
        glColor3ub(red,green,blue);

        /* top of cube*/
        glVertex3f(1.8, 0.5,0.6);
        glVertex3f(1.8, 0.5,0.2);
        glVertex3f(2.1, 0.5, 0.2);
        glVertex3f(2.1,0.5,0.6);

        /* bottom of cube*/
        glVertex3f( 2.1,0.2,0.6);
        glVertex3f(2.1,0.2,0.2);
        glVertex3f(1.8,0.2,0.6);
        glVertex3f( 1.8,0.2,0.6);

        /* back of cube.*/

        glVertex3f(2.1,0.4,0.6);
        glVertex3f(2.1,0.4,0.2);
        glVertex3f(2.1,0.2,0.2);
        glVertex3f(2.1,0.2,0.6);

        /* left of cube*/
        glVertex3f(1.8,0.2,0.2);
        glVertex3f(1.8,0.5,0.2);
        glVertex3f(2.1,0.5,0.2);
        glVertex3f(2.1,0.2,0.2);
```

```
        /* Right of cube */
        glVertex3f(1.8,0.2,0.6);
        glVertex3f(1.8,0.5,0.6);
        glVertex3f(2.1,0.5,0.6);
        glVertex3f(2.1,0.2,0.6);
//******************MIDDLE BODY OF THE
BUS***********************************

        glVertex3f( 0.6, 0.5,0.6);
        glVertex3f(0.6, 0.2,0.6);
        glVertex3f(1.8, 0.2, 0.6);
        glVertex3f(1.8,0.5,0.6);

        /* bottom of cube*/
        glVertex3f( 0.6,0.2,0.6);
        glVertex3f(0.6,0.2,0.2);
        glVertex3f(1.8,0.2,0.2);
        glVertex3f( 1.8,0.2,0.6);

        /* back of cube.*/
        glVertex3f(0.6,0.5,0.2);
        glVertex3f(0.6,0.2,0.2);
        glVertex3f(1.8,0.2,0.2);
        glVertex3f(1.8,0.5,0.2);
//********************** WINDOW OF THE BUS
*********************************
        glColor3ub(rWindow,gWindow,bWindow);
        glVertex3f( 0.77, 0.63,0.2);
        glVertex3f(0.75, 0.5,0.2);        // front window
        glVertex3f(1.2, 0.5, 0.2);
        glVertex3f( 1.22,0.63,0.2);

        glVertex3f(1.27,0.63,.2);
        glVertex3f(1.25,0.5,0.2);        // back window
        glVertex3f(1.65,0.5,0.2);
        glVertex3f(1.67,0.63,0.2);

        glColor3ub(red,green,blue);
        glVertex3f(0.7,0.65,0.2);
        glVertex3f(0.7,0.5,.2);       //first separation
        glVertex3f(0.75,0.5,0.2);
        glVertex3f(0.77,0.65,0.2);

        glVertex3f(1.2,0.65,0.2);
        glVertex3f(1.2,0.5,.2);       //second separation
        glVertex3f(1.25,0.5,0.2);
        glVertex3f(1.27,0.65,0.2);

        glVertex3f(1.65,0.65,0.2);
        glVertex3f(1.65,0.5,.2);     //3d separation
        glVertex3f(1.7,0.5,0.2);
        glVertex3f(1.7,0.65,0.2);

        glVertex3f( 0.75, 0.65,0.2);
        glVertex3f(0.75, 0.63,0.2);        //line strip
        glVertex3f(1.7, 0.63, 0.2);
        glVertex3f( 1.7,0.65,0.2);

        glVertex3f( 0.75, 0.65,0.6);
        glVertex3f(0.75, 0.63,0.6);        //line strip
        glVertex3f(1.7, 0.63, 0.6);
        glVertex3f( 1.7,0.65,0.6);

        glColor3ub(rWindow,gWindow,bWindow);
        glVertex3f( 0.77, 0.63,0.6);
        glVertex3f(0.75, 0.5,0.6);        //quad front window
```

```
        glVertex3f(1.2, 0.5, 0.6);
        glVertex3f( 1.22,0.63,0.6);

        glVertex3f(1.27,0.63,.6);
        glVertex3f(1.25,0.5,0.6);        //quad back window
        glVertex3f(1.65,0.5,0.6);
        glVertex3f(1.67,0.63,0.6);

        glColor3ub(red,green,blue);
        glVertex3f(0.7,0.65,0.6);
        glVertex3f(0.7,0.5,.6);       //first separation
        glVertex3f(0.75,0.5,0.6);
        glVertex3f(0.77,0.65,0.6);

        glVertex3f(1.2,0.65,0.6);
        glVertex3f(1.2,0.5,.6);       //second separation
        glVertex3f(1.25,0.5,0.6);
        glVertex3f(1.27,0.65,0.6);

        glColor3ub(red,green,blue);
        glVertex3f(1.65,0.65,0.6);
        glVertex3f(1.65,0.5,.6);
        glVertex3f(1.7,0.5,0.6);
        glVertex3f(1.7,0.65,0.6);
        glEnd();
//*************************************************************
        glBegin(GL_QUADS);

        /* top of cube*/
        glColor3ub(rWindow,gWindow,bWindow);

        glVertex3f( 0.2, 0.4,0.6);
        glVertex3f(0.2, 0.4,0.2);        //quad front window
        glVertex3f(0.2, 0.63, 0.2);
        glVertex3f( 0.2,0.63,0.6);

        glVertex3f(2.1,0.63,.6);
        glVertex3f(2.1,0.63,0.2);        //quad back window
        glVertex3f(2.1,0.4,0.2);
        glVertex3f(2.1,0.4,0.6);

        glColor3ub(red,green,blue);
        glVertex3f(2.1,0.65,.6);
        glVertex3f(2.1,0.65,0.2);        //quad back window
        glVertex3f(2.1,0.63,0.2);
        glVertex3f(2.1,0.63,0.6);

        glVertex3f( 0.2, 0.63,0.6);
        glVertex3f(0.2, 0.63,0.2);        //quad front window
        glVertex3f(0.2, 0.65, 0.2);
        glVertex3f( 0.2,0.65,0.6);
        glEnd();
//*****************************road and surrounding
development***********************************
        if(flag1)
        {
            glBegin(GL_QUADS);
            glPushMatrix();
            glTranslatef(xw,0,0);
            xangle=5.0;
            zangle=0;

            glColor3f(0,0.9,0);
            glVertex3f(-100,0.1,-100);
            glVertex3f(-100,0.1,0);         //a green surroundings
            glVertex3f(100,0.1,0);
```

```
glVertex3f(100,0.1,-100);

glColor3f(1,1,1);
glVertex3f(-100,0.1,0);
glVertex3f(-100,0.1,0.1);          //road boundary
glVertex3f(100,0.1,0.1);
glVertex3f(100,0.1,0);

glColor3f(0.3,0.4,0.5);
glVertex3f(-100,0.1,0.1);
glVertex3f(-100,0.1,0.75);          //a long road
glVertex3f(100,0.1,0.75);
glVertex3f(100,0.1,0.1);

glColor3f(1.0,0.9,0.0);
glVertex3f(-100,0.1,0.75);        //a median
glVertex3f(-100,0.1,0.85);
glVertex3f(100,0.1,0.85);
glVertex3f(100,0.1,0.75);

glColor3f(0.3,0.4,0.5);
glVertex3f(-100,0.1,0.85);
glVertex3f(-100,0.1,1.30);          //a long road
glVertex3f(100,0.1,1.30);
glVertex3f(100,0.1,0.85);

glColor3f(1,1,1);
glVertex3f(-100,0.1,1.30);
glVertex3f(-100,0.1,1.40);          //road boundary
glVertex3f(100,0.1,1.40);
glVertex3f(100,0.1,1.30);

glColor3f(0,0.9,0);
glVertex3f(-100,0.1,1.40);
glVertex3f(-100,0.1,100);          //a green surroundings
glVertex3f(100,0.1,100);
glVertex3f(100,0.1,1.40);
glPopMatrix();
glEnd();

/** sun */
glColor3f(1.0,0.9,0.0);
glPushMatrix();
glTranslatef(-3,2.5,-8.1);
glutSolidSphere(.5,50,50);
glPopMatrix();

/** Aeroplane */
glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex3f(4+aeroplaneXinc,1.9+aeroplaneYinc,-5);
glVertex3f(4.20+aeroplaneXinc,2.08+aeroplaneYinc,-5);
glVertex3f(4.72+aeroplaneXinc,2.08+aeroplaneYinc,-5);
glVertex3f(4.84+aeroplaneXinc,2.2+aeroplaneYinc,-5);
glVertex3f(5+aeroplaneXinc,2.2+aeroplaneYinc,-5);
glVertex3f(4.68+aeroplaneXinc,1.9+aeroplaneYinc,-5);
glEnd();

glColor3f(0.8,0.8,0.8);
glBegin(GL_LINE_LOOP);
glVertex3f(4+aeroplaneXinc,1.9+aeroplaneYinc,-5);
glVertex3f(4.20+aeroplaneXinc,2.08+aeroplaneYinc,-5);
glVertex3f(4.72+aeroplaneXinc,2.08+aeroplaneYinc,-5);
glVertex3f(4.84+aeroplaneXinc,2.2+aeroplaneYinc,-5);
glVertex3f(5+aeroplaneXinc,2.2+aeroplaneYinc,-5);
glVertex3f(4.68+aeroplaneXinc,1.9+aeroplaneYinc,-5);
```

```
            glEnd();

            if(aeroplaneflag)
            {
                aeroplaneXinc = 0;
                aeroplaneYinc = 0;
                aeroplaneflag = 0;
                glutTimerFunc(10,timer,0);
            }

            if(cloudflag == 1)
            {
                glutTimerFunc(0,cloudtimer,0);
                random1 = -5 + static_cast <float> (rand()) /( static_cast
<float> (RAND_MAX/(-8+5)));
                random2 = -5 + static_cast <float> (rand()) /( static_cast
<float> (RAND_MAX/(-8+5)));
                cloudflag = 0;
            }

            /** cloud1 */
            glColor3f(1.0,1.0,1.0);
            glPushMatrix();
            glTranslatef(-8+cloudXinc,2,random1);
            glutSolidSphere(.17,50,50);
            glTranslatef(0.32,0,0);
            glutSolidSphere(.3,50,50);
            glTranslatef(0.20,0,0);
            glutSolidSphere(.3,50,50);
            glTranslatef(0.32,0,0);
            glutSolidSphere(.17,50,50);
            glPopMatrix();

            /** cloud2 */
            glColor3f(1.0,1.0,1.0);
            glPushMatrix();
            glTranslatef(-6.5+cloudXinc,2.5,random2);
            glutSolidSphere(.17,50,50);
            glTranslatef(0.32,0,0);
            glutSolidSphere(.3,50,50);
            glTranslatef(0.20,0,0);
            glutSolidSphere(.3,50,50);
            glTranslatef(0.20,0,0);
            glutSolidSphere(.3,50,50);
            glTranslatef(0.32,0,0);
            glutSolidSphere(.17,50,50);
            glPopMatrix();

            /** Mountains */
            glEnable(GL_LIGHTING);
            glEnable(GL_LIGHT0);
            glColor3ub(115,118,83);
            glPushMatrix();
            glTranslatef(xw,0,0);
            glPushMatrix();
            glTranslatef(-100,0.1,-10);
            glutSolidDodecahedron();
            for(int i=0; i<40; i++)
            {
                glTranslatef(1.5,0,0);
                glutSolidDodecahedron();

                glTranslatef(3,0,0);
                glutSolidDodecahedron();

                glTranslatef(1,0,0);
```

```
            glutSolidDodecahedron();
        }
        glPopMatrix();
        glDisable(GL_LIGHTING);
        glDisable(GL_LIGHT0);

        /** Trees */
        glPushMatrix();
        glTranslatef(-100,0.1,-1);
        for(int i=0; i<45; i++)
        {
            glTranslatef(5,0,0);
            glPushMatrix();
            glColor3f(0.9,0.2,0.0);
            glBegin(GL_POLYGON);
            glVertex3f(0,0,0);
            glVertex3f(0,0.4,0);
            glVertex3f(.12,0.4,0);
            glVertex3f(.12,0,0);
            glEnd();
            glColor3f(0.0,0.5,0.0);
            glTranslatef(0,0.3,0);
            glutSolidSphere(.16,50,50);
            glTranslatef(0.16,0,0);
            glutSolidSphere(.16,50,50);
            glTranslatef(-0.11,0.18,0);
            glutSolidSphere(.14,50,50);
            glTranslatef(0.06,0,0);
            glutSolidSphere(.14,50,50);
            glTranslatef(-0.02,0.15,0);
            glutSolidSphere(.10,50,50);
            glPopMatrix();
        }
        glPopMatrix();
        glPushMatrix();
        glTranslatef(-100,0.1,2);
        for(int i=0; i<45; i++)
        {
            glTranslatef(5,0,0);
            glPushMatrix();
            glColor3f(0.9,0.2,0.0);
            glBegin(GL_POLYGON);
            glVertex3f(0,0,0);
            glVertex3f(0,0.4,0);
            glVertex3f(.12,0.4,0);
            glVertex3f(.12,0,0);
            glEnd();
            glColor3f(0.0,0.5,0.0);
            glTranslatef(0,0.3,0);
            glutSolidSphere(.16,50,50);
            glTranslatef(0.16,0,0);
            glutSolidSphere(.16,50,50);
            glTranslatef(-0.11,0.18,0);
            glutSolidSphere(.14,50,50);
            glTranslatef(0.06,0,0);
            glutSolidSphere(.14,50,50);
            glTranslatef(-0.02,0.15,0);
            glutSolidSphere(.10,50,50);
            glPopMatrix();
        }
        glPopMatrix();
        glPopMatrix();
    }

    if(wheelflag)
    {
```

```
            glPushMatrix();
            glTranslatef(xw,0,0);
            glColor3f(0.3,0.3,0.3);
            glBegin(GL_QUADS);
            for(i=0; i<200; i+=0.2)
            {
                glVertex3f(-100+i,0,1.40);
                glVertex3f(-99.9+i,0,1.40);
                glVertex3f(-99.9+i,0.2,1.40);
                glVertex3f(-100+i,0.2,1.40);
                i+=0.5;
            }
            for(i=0; i<200; i+=0.2)
            {
                glVertex3f(-100+i,0,0);
                glVertex3f(-99.9+i,0,0);
                glVertex3f(-99.9+i,0.2,0);
                glVertex3f(-100+i,0.2,0);
                i+=0.5;
            }
            glEnd();
            glPopMatrix();

            if(busflag)
            {
                glutTimerFunc(10,bustimer,0);
                busflag=0;
            }
        }
//*************************************************************************
************************
        glBegin(GL_QUADS);                /* start drawing the cube.*/

        /* top of cube*/
        glColor3ub(rWindow,gWindow,bWindow);
        glVertex3f( 0.2, 0.5,0.6);
        glVertex3f( 0.2, 0.63,0.6);
        glVertex3f( 0.7,0.63,0.6);       //tri front window
        glVertex3f(0.7,0.5,0.6);

        glVertex3f( 0.2, 0.5,0.2);
        glVertex3f( 0.2, 0.63,0.2);
        glVertex3f( 0.7,0.63,0.2);       //tri front window
        glVertex3f(0.7,0.5,0.2);

        glVertex3f( 1.7, 0.63,0.6);
        glVertex3f( 2.1,0.63,0.6);
        glVertex3f( 2.1,0.5,0.6);       //tri back window
        glVertex3f(1.7,0.5,0.6);

        glVertex3f( 1.7, 0.63,0.2);
        glVertex3f( 2.1,0.63,0.2);
        glVertex3f( 2.1,0.5,0.2);       //tri back window
        glVertex3f(1.7,0.5,0.2);

        glColor3ub(red,green,blue);
        glVertex3f(.2,0.63,0.6);
        glVertex3f(.2,0.65,0.6);     //1st separation
        glVertex3f(.7,0.65,0.6);
        glVertex3f(.7,0.63,0.6);

        glVertex3f(.2,0.63,0.2);
        glVertex3f(.2,0.65,0.2);     //2nd separation
        glVertex3f(.7,0.65,0.2);
        glVertex3f(.7,0.63,0.2);
```

```
        glVertex3f( 1.7, 0.65,0.6);
        glVertex3f( 2.1,0.65,0.6);    //3rd separation
        glVertex3f( 2.1,0.63,0.6);
        glVertex3f(1.7,0.63,0.6);

        glVertex3f( 1.7, 0.65,0.2);
        glVertex3f( 2.1,0.65,0.2);    //3rd separation
        glVertex3f( 2.1,0.63,0.2);
        glVertex3f(1.7,0.63,0.2);
        glEnd();

        glBegin(GL_TRIANGLES);                  /* start drawing the cube.*/
        glColor3ub(rWindow,gWindow,bWindow);

        glEnd();
//************IGNITION SYSTEM**********************************
        glPushMatrix();
        glColor3f(0.7,0.7,0.7);
        glTranslatef(1.65,0.2,0.3);
        glRotatef(90.0,0,1,0);
        gluCylinder(t,0.02,0.03,.5,10,10);
        glPopMatrix();
//********************WHEEL*********************************************

        glColor3f(0.7,0.7,0.7);
        glPushMatrix();
        glBegin(GL_LINE_STRIP);
        for(theta=0; theta<360; theta=theta+30)
        {
            glVertex3f(0.6,0.2,0.62);

glVertex3f(0.6+(0.08*(cos(((theta+angle)*3.14)/180))),0.2+(0.08*(sin(((the
ta+angle)*3.14)/180))),0.62);
        }
        glEnd();

        glBegin(GL_LINE_STRIP);
        for(theta=0; theta<360; theta=theta+30)
        {
            glVertex3f(0.6,0.2,0.18);

glVertex3f(0.6+(0.08*(cos(((theta+angle)*3.14)/180))),0.2+(0.08*(sin(((the
ta+angle)*3.14)/180))),0.18);
        }
        glEnd();

        glBegin(GL_LINE_STRIP);
        for(theta=0; theta<360; theta=theta+30)
        {
            glVertex3f(1.7,0.2,0.18);

glVertex3f(1.7+(0.08*(cos(((theta+angle)*3.14)/180))),0.2+(0.08*(sin(((the
ta+angle)*3.14)/180))),0.18);
        }
        glEnd();

        glBegin(GL_LINE_STRIP);
        for(theta=0; theta<360; theta=theta+30)
        {
            glVertex3f(1.7,0.2,0.62);

glVertex3f(1.7+(0.08*(cos(((theta+angle)*3.14)/180))),0.2+(0.08*(sin(((the
ta+angle)*3.14)/180))),0.62);
        }
        glEnd();
        glTranslatef(0.6,0.2,0.6);
```

```
        glColor3f(0,0,0);
        glutSolidTorus(0.025,0.07,10,25);

        glTranslatef(0,0,-0.4);
        glutSolidTorus(0.025,0.07,10,25);

        glTranslatef(1.1,0,0);
        glutSolidTorus(0.025,0.07,10,25);

        glTranslatef(0,0,0.4);
        glutSolidTorus(0.025,0.07,10,25);
        glPopMatrix();
//*************************************************************
        glPopMatrix();
        glEnable(GL_DEPTH_TEST);
        glutPostRedisplay();
        glutSwapBuffers();
    }
}

/** Handles keyboard events*/
void keyboard(GLubyte key, GLint x, GLint y)
{
    /** This switch case helps to identify which key was pressed*/
    switch ( key )
    {
    case ESCAPE :
        printf("escape pressed. exit.\n");
        glutDestroyWindow(window);  /* Close our window */
        exit(0);
        break;

    case ' ':
        view=1;
        DrawDifferentScenes();
        break;

    case 'x':
        xangle += 5.0;
        glutPostRedisplay();
        break;

    case 'X':
        xangle -= 5.0;
        glutPostRedisplay();
        break;

    case 'y':
        yangle += 5.0;
        glutPostRedisplay();
        break;

    case 'Y':
        yangle -= 5.0;
        glutPostRedisplay();
        break;

    case 'z':
        zangle += 5.0;
        glutPostRedisplay();
        break;

    case 'Z':
        zangle -= 5.0;
        glutPostRedisplay();
        break;
```

```
    case 'u':                           /* Move down */
        if(wheelflag)
        {
            if(yTranslation <= 0.6)
                yTranslation += 0.2;                    /* Move down */
        }
        else
            yTranslation += 0.2;                    /* Move down */
        glutPostRedisplay();
        break;

    case 'U':
        if(wheelflag)
        {
            if(yTranslation > -0.4)
                yTranslation -= 0.2;                    /* Move up */
        }
        else
            yTranslation -= 0.2;                    /* Move up */
        glutPostRedisplay();
        break;

    case 'f':                           /* Move forward */
        if(wheelflag)
        {
            if(zTranslation < 5)
                zTranslation += 0.2;                    /* Move forward
*/
        }
        else
            zTranslation += 0.2;                    /* Move forward */
        glutPostRedisplay();
        break;


    case 'F':
        if(wheelflag)
        {
            if(zTranslation > -1.8)
                zTranslation -= 0.2;                    /* Move away */
        }
        else
            zTranslation -= 0.2;                    /* Move away */
        glutPostRedisplay();
        break;

    case 's':
        if(!wheelflag)
        {
            zScale+=.2;
            glutPostRedisplay();
        }
        break;

    case 'S':
        if(!wheelflag)
        {
            zScale-=0.2;
            glutPostRedisplay();

        }
        break;

    case 'a':
        if(!wheelflag)
        {
```

```
            yScale+=.2;
            glutPostRedisplay();

        }
        break;

    case 'A':
        if(!wheelflag)
        {
            yScale-=0.2;
            glutPostRedisplay();
        }
        break;

    case 'q':
        if(!wheelflag)
        {
            xScale+=.2;
            glutPostRedisplay();
        }
        break;

    case 'Q':
        if(!wheelflag)
        {
            xScale-=0.2;
            glutPostRedisplay();
        }
        break;
    default:
        break;
    }

}

/** Handles special key events*/
void specialFunction(int key,int keyx,int keyy)
{

    /** This switch case helps to identify which key was pressed*/
    switch ( key )
    {
    case GLUT_KEY_RIGHT:
        if(!wheelflag)
            xTranslation += 0.2;
        glutPostRedisplay();
        break;

    case GLUT_KEY_LEFT:
        if(!wheelflag)
            xTranslation -= 0.2;
        glutPostRedisplay();
        break;
    }
}

/** Handles the menu operation */
void myMenu(int id)
{
    if (id==1)
    {
        flag1=0;
        wheelflag=0;
        busflag=0;
        xw=50;
        rWindow=77,gWindow=77,bWindow=77;
```

```
        red=104,green=12,blue=13; //brown
        backgroundflag = 1;
        glutPostRedisplay();
    }
    if(id ==2)
    {
        flag1=1;
        flag2=0;
        wheelflag=1;
        xw=50;
        busflag=1;
        rWindow=118,gWindow=215,bWindow=234;
        backgroundflag = 0;
        glutPostRedisplay();
    }
    if(id==3)
    {

        if(flag2 == 1)
            flag2=0;
        else
            flag2=1;
        xangle += 5.0;
        glutPostRedisplay();
    }
    if(id==13)
    {
        aflag=1;
        day=1;
        glClearColor(1,1,1,1);
        glDisable(GL_FOG);
        glDisable(GL_LIGHTING);
        glDisable(GL_LIGHT0);
        backr=0,backg=0.9,backb=0.9;
        glutPostRedisplay();
    }
    if(id==14)
    {
        aflag=0;
        day=0;
        flag2=2;
        glClearColor(0.1,0.1,0.1,0);
        backr=0,backg=0,backb=0;
        GLfloat fogcolour[4]= {0.0,0.0,0.0,1.0};
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glutPostRedisplay();
    }
    if(id == 15)
    {
        aeroplaneYinc = 10;
        aeroplaneflag = 1;
    }
}

/** Handles the color menu operation when clicked */
void colorMenu(int id)
{
    if(id ==6)
        red=104,green=12,blue=13; //brown
    if(id==7)
        red=172,green=13,blue=255; //purple 3
    if (id==8)
        red=255,green=19,blue=36; //red 4
    if(id==9)
        red=0,green=32,blue=193; //blue 5
```

```
    if(id==10)
        red=255,green=70,blue=32; //orange 6
    if(id==11)
        red=255,green=42,blue=238; //pink 7
    if (id==12)
        red=220,green=149,blue=37;
    glutPostRedisplay();
}

/** Manages the aeroplane movement */
void timer(int)
{
    if( aeroplaneXinc + 4.4 > -3)
        aeroplaneXinc -= 0.02;

    if( aeroplaneYinc + 2.3 <4)
    {
        glutTimerFunc(1000/60,timer,0);
        aeroplaneYinc += 0.005;
    }
    glutPostRedisplay();
}

/** It handles the movement of clouds */
void cloudtimer(int)
{
    if(cloudXinc + -1 < 14)
    {
        cloudXinc += 0.02;
        glutTimerFunc(1000/60,cloudtimer,0);
    }
    else
    {
        cloudXinc = 0;
        cloudflag = 1;
    }
    glutPostRedisplay();
}

/** This method handles the movement of the bus */
void bustimer(int)
{
    if (25 >= xw && -25 <= xw)
    {
        angle=xw/0.0007;
        xw+=0.025;
        glutTimerFunc(1000/60,bustimer,0);
    }
    else
    {
        busflag=1;
        xw = 0;
    }
    glutPostRedisplay();
}

/** Mian function from where the execution of program will start */
int main(int argc, char** argv)
{

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA |      /* Red Green Blue and Alpha */
                        GLUT_DOUBLE|     /* double buffer */
                        GLUT_DEPTH);     /* Z buffer (depth) */

    glutInitWindowSize(Xsize,Ysize);     /* set initial window size. */
```

```
    glutInitWindowPosition(0,0);         /* upper left corner of the
screen. */
    glutCreateWindow("Final Project with 3D scenes"); /* Open a window
with a title. */

    glutDisplayFunc(DrawDifferentScenes); //Start drawing the different
scenes and objects
    glEnable(GL_DEPTH_TEST);

    glutKeyboardFunc(keyboard);
    glutSpecialFunc(specialFunction);
    int submenu=glutCreateMenu(colorMenu);
    glutAddMenuEntry("Brown", 6);
    glutAddMenuEntry("Purple",7);
    glutAddMenuEntry("Red",8);
    glutAddMenuEntry("Blue",9);
    glutAddMenuEntry("Orange",10);
    glutAddMenuEntry("Pink",11);
    glutAddMenuEntry("Dark Yellow", 12);

    glutCreateMenu(myMenu);
    glutAddMenuEntry("Bus model mode", 1);
    glutAddMenuEntry("Bus driving mode", 2);
    glutAddMenuEntry("Aeroplane",15);
    glutAddMenuEntry("fog effect",3);
    glutAddSubMenu("Bus colors",submenu);
    glutAddMenuEntry("Day mode",13);
    glutAddMenuEntry("Night mode",14);

    glutAttachMenu(GLUT_RIGHT_BUTTON);

    glutMainLoop();
}
```