Computer Graphics

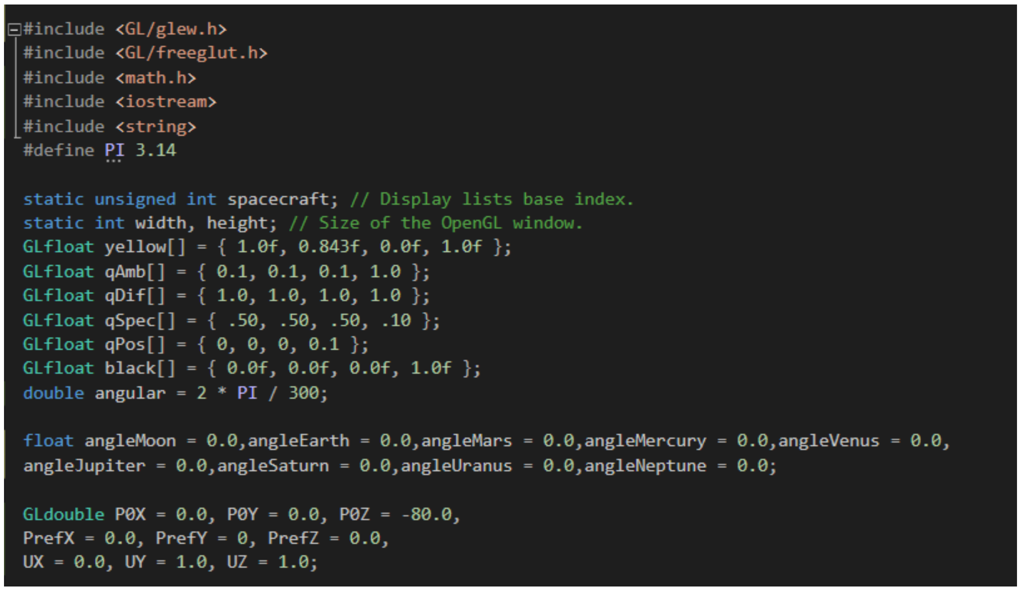
Heba-tallah Mostafa Abdel-Reheem

>>> 19016836

….. Course Project

>>Global Variables Initialization:

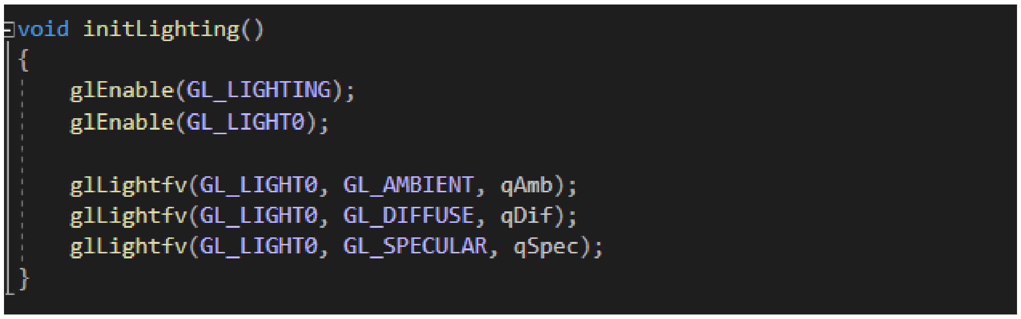
- *Initialization of colors arrays & Phong's parameters , initialization of rotation angles which are used for animation & initialization of space craft viewport parameters P_0, P_{ref} & V_{up} vector .*



```
#include <GL/glew.h>
#include <GL/freeglut.h>
#include <math.h>
#include <iostream>
#include <string>
#define PI 3.14

static unsigned int spacecraft; // Display lists base index.
static int width, height; // Size of the OpenGL window.
GLfloat yellow[] = { 1.0f, 0.843f, 0.0f, 1.0f };
GLfloat qAmb[] = { 0.1, 0.1, 0.1, 1.0 };
GLfloat qDif[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat qSpec[] = { .50, .50, .50, .10 };
GLfloat qPos[] = { 0, 0, 0, 0.1 };
GLfloat black[] = { 0.0f, 0.0f, 0.0f, 1.0f };
double angular = 2 * PI / 300;

float angleMoon = 0.0,angleEarth = 0.0,angleMars = 0.0,angleMercury = 0.0,angleVenus = 0.0,
angleJupiter = 0.0,angleSaturn = 0.0,angleUranus = 0.0,angleNeptune = 0.0;

GLdouble P0X = 0.0, P0Y = 0.0, P0Z = -80.0,
PrefX = 0.0, PrefY = 0, PrefZ = 0.0,
UX = 0.0, UY = 1.0, UZ = 1.0;
```

>>initLighting Function :

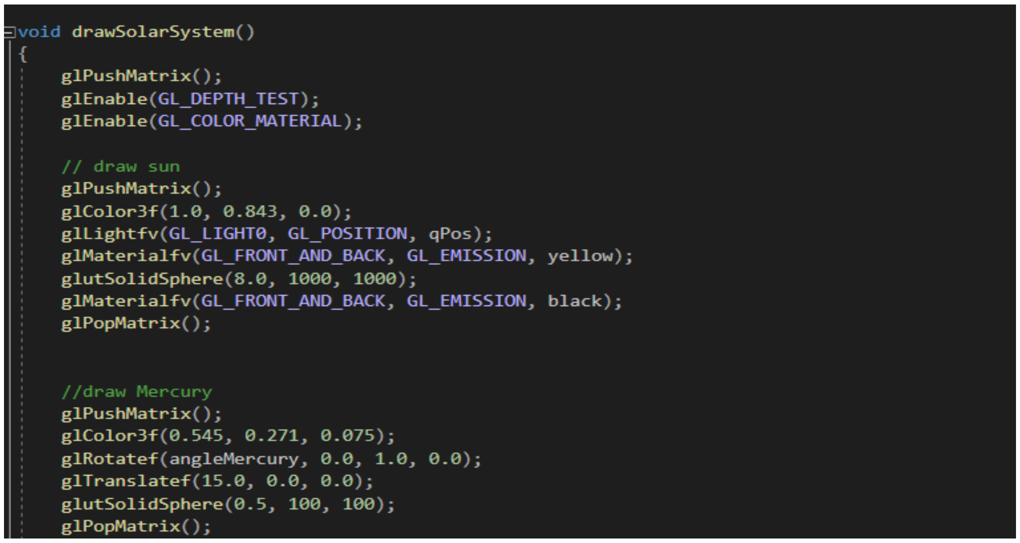
- *Enable and define the used lighting and its parameters & values.*



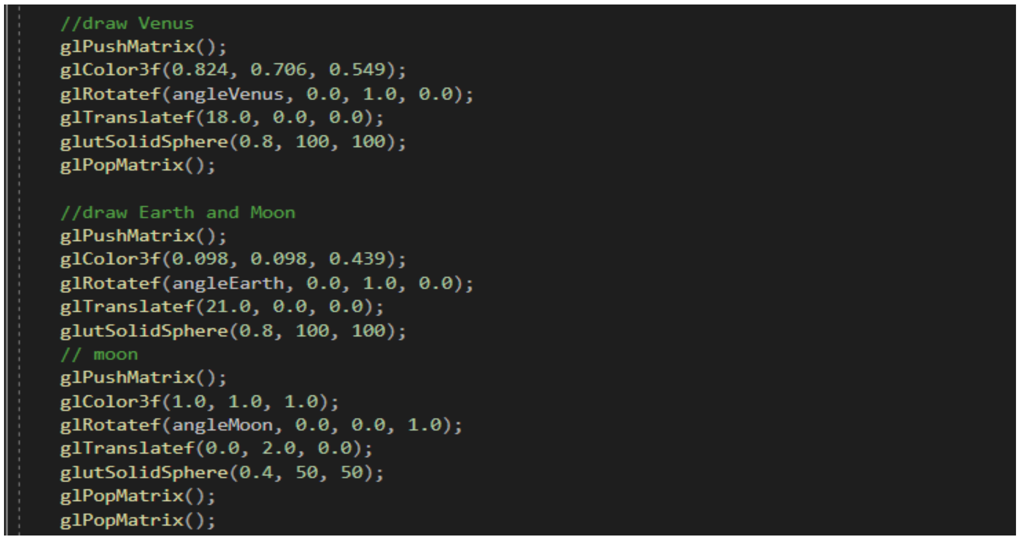
```
void initLighting()
{
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glLightfv(GL_LIGHT0, GL_AMBIENT, qAmb);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, qDif);
    glLightfv(GL_LIGHT0, GL_SPECULAR, qSpec);
}
```

>> drawSolarSystem Function :

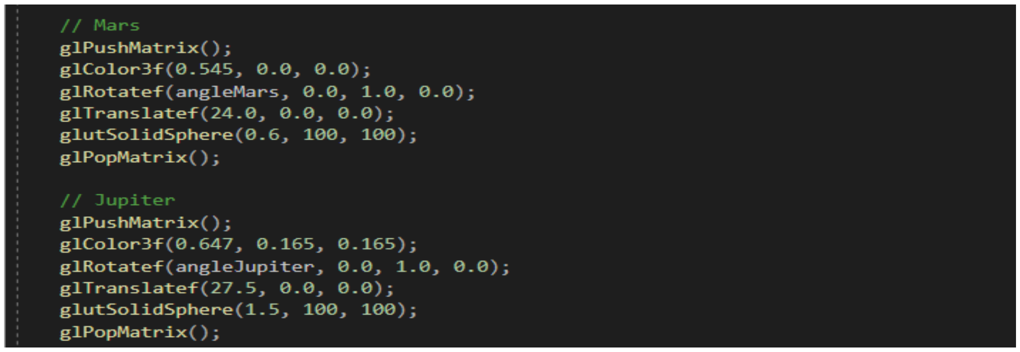
- *In this function we draw the sun and the 8 planets using the built-in function glutSolidSphere() .*



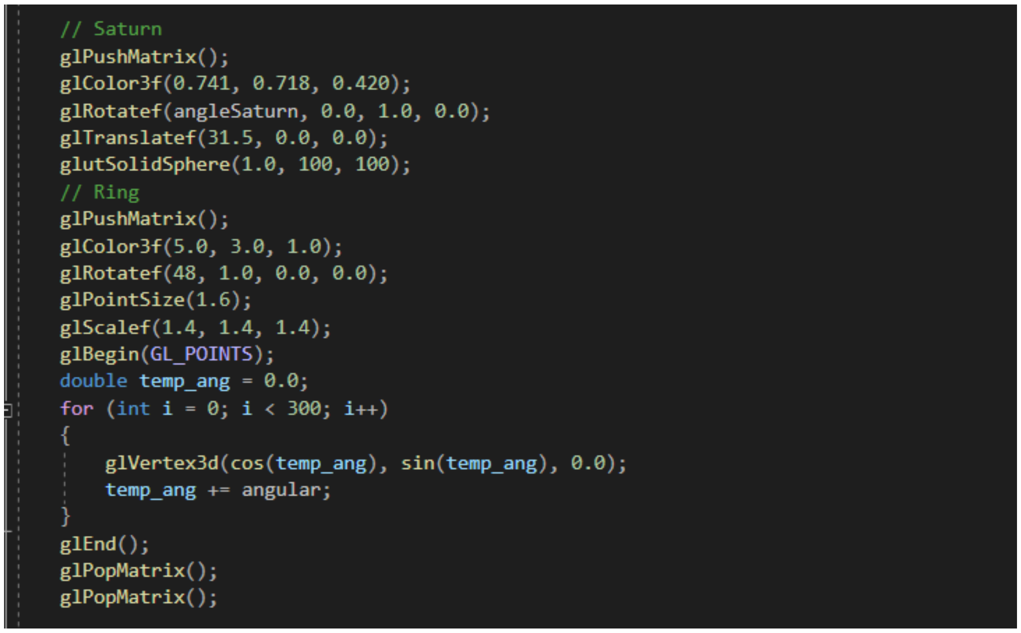
```
void drawSolarSystem()
{
    glPushMatrix();
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_COLOR_MATERIAL);

    // draw sun
    glPushMatrix();
    glColor3f(1.0, 0.843, 0.0);
    glLightfv(GL_LIGHT0, GL_POSITION, qPos);
    glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, yellow);
    glutSolidSphere(8.0, 1000, 1000);
    glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, black);
    glPopMatrix();


    //draw Mercury
    glPushMatrix();
    glColor3f(0.545, 0.271, 0.075);
    glRotatef(angleMercury, 0.0, 1.0, 0.0);
    glTranslatef(15.0, 0.0, 0.0);
    glutSolidSphere(0.5, 100, 100);
    glPopMatrix();
```



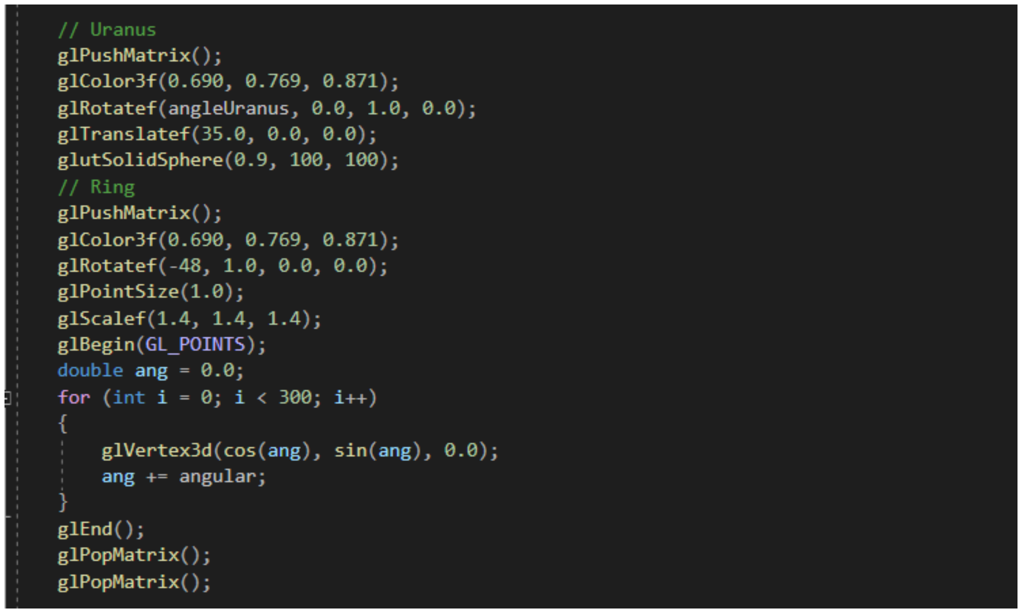
```
    //draw Venus
    glPushMatrix();
    glColor3f(0.824, 0.706, 0.549);
    glRotatef(angleVenus, 0.0, 1.0, 0.0);
    glTranslatef(18.0, 0.0, 0.0);
    glutSolidSphere(0.8, 100, 100);
    glPopMatrix();

    //draw Earth and Moon
    glPushMatrix();
    glColor3f(0.098, 0.098, 0.439);
    glRotatef(angleEarth, 0.0, 1.0, 0.0);
    glTranslatef(21.0, 0.0, 0.0);
    glutSolidSphere(0.8, 100, 100);
    // moon
    glPushMatrix();
    glColor3f(1.0, 1.0, 1.0);
    glRotatef(angleMoon, 0.0, 0.0, 1.0);
    glTranslatef(0.0, 2.0, 0.0);
    glutSolidSphere(0.4, 50, 50);
    glPopMatrix();
    glPopMatrix();
```



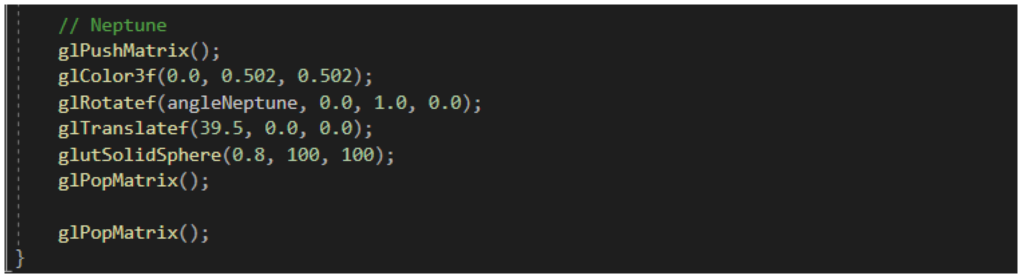
```
    // Mars
    glPushMatrix();
    glColor3f(0.545, 0.0, 0.0);
    glRotatef(angleMars, 0.0, 1.0, 0.0);
    glTranslatef(24.0, 0.0, 0.0);
    glutSolidSphere(0.6, 100, 100);
    glPopMatrix();

    // Jupiter
    glPushMatrix();
    glColor3f(0.647, 0.165, 0.165);
    glRotatef(angleJupiter, 0.0, 1.0, 0.0);
    glTranslatef(27.5, 0.0, 0.0);
    glutSolidSphere(1.5, 100, 100);
    glPopMatrix();
```



```
    // Saturn
    glPushMatrix();
    glColor3f(0.741, 0.718, 0.420);
    glRotatef(angleSaturn, 0.0, 1.0, 0.0);
    glTranslatef(31.5, 0.0, 0.0);
    glutSolidSphere(1.0, 100, 100);
    // Ring
    glPushMatrix();
    glColor3f(5.0, 3.0, 1.0);
    glRotatef(48, 1.0, 0.0, 0.0);
    glPointSize(1.6);
    glScalef(1.4, 1.4, 1.4);
    glBegin(GL_POINTS);
    double temp_ang = 0.0;
    for (int i = 0; i < 300; i++)
    {
        glVertex3d(cos(temp_ang), sin(temp_ang), 0.0);
        temp_ang += angular;
    }
    glEnd();
    glPopMatrix();
    glPopMatrix();
```



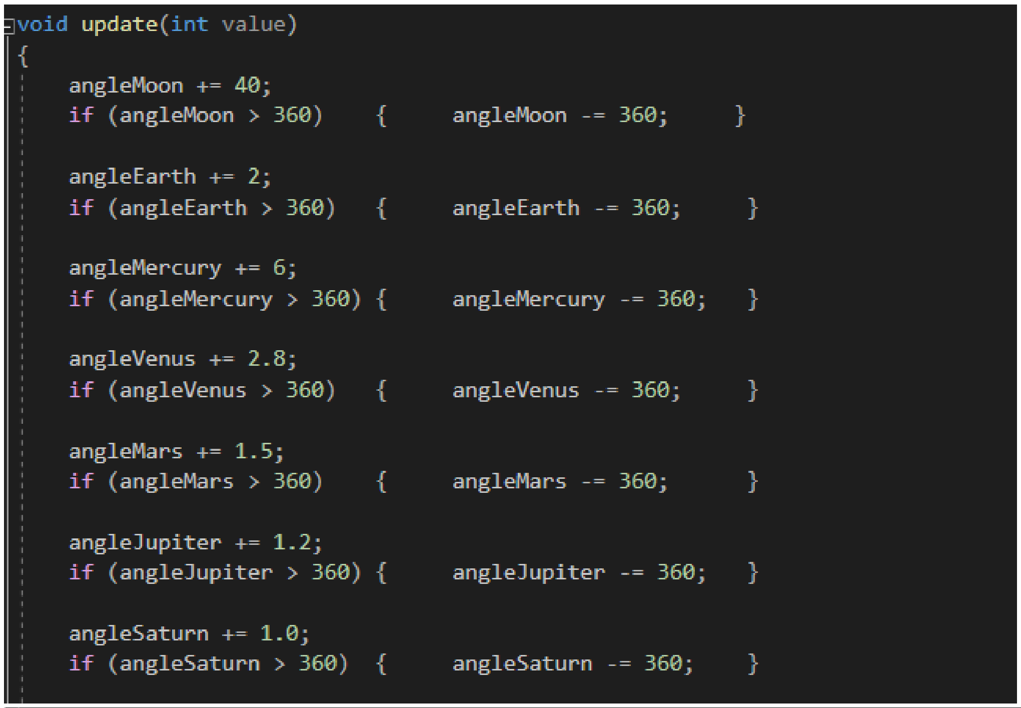
```
    // Uranus
    glPushMatrix();
    glColor3f(0.690, 0.769, 0.871);
    glRotatef(angleUranus, 0.0, 1.0, 0.0);
    glTranslatef(35.0, 0.0, 0.0);
    glutSolidSphere(0.9, 100, 100);
    // Ring
    glPushMatrix();
    glColor3f(0.690, 0.769, 0.871);
    glRotatef(-48, 1.0, 0.0, 0.0);
    glPointSize(1.0);
    glScalef(1.4, 1.4, 1.4);
    glBegin(GL_POINTS);
    double ang = 0.0;
    for (int i = 0; i < 300; i++)
    {
        glVertex3d(cos(ang), sin(ang), 0.0);
        ang += angular;
    }
    glEnd();
    glPopMatrix();
    glPopMatrix();
```



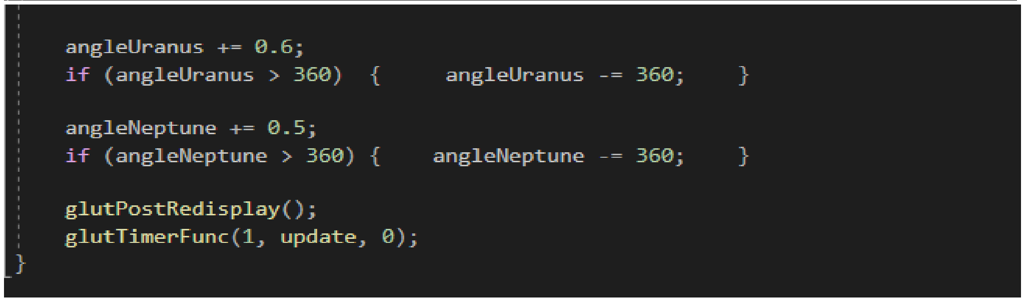
```
    // Neptune
    glPushMatrix();
    glColor3f(0.0, 0.502, 0.502);
    glRotatef(angleNeptune, 0.0, 1.0, 0.0);
    glTranslatef(39.5, 0.0, 0.0);
    glutSolidSphere(0.8, 100, 100);
    glPopMatrix();

    glPopMatrix();
}
```

>> Update Function :

- *This function is responsible for updating the rotation angles for each planet and the moon and reset it if it exceeds 360˚.*

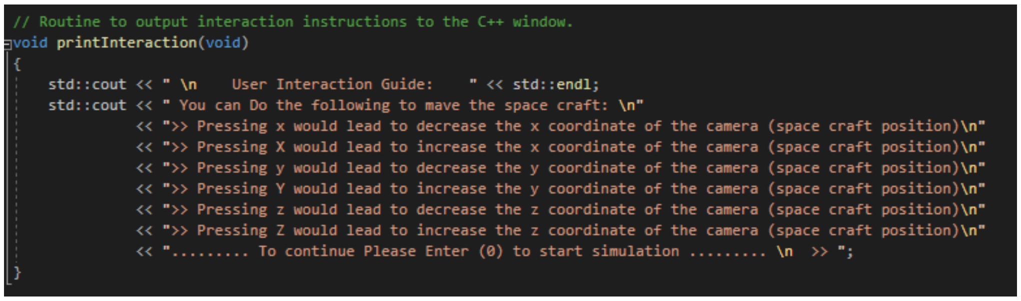


```
void update(int value)
{
    angleMoon += 40;
    if (angleMoon > 360)    {     angleMoon -= 360;     }

    angleEarth += 2;
    if (angleEarth > 360)   {     angleEarth -= 360;     }

    angleMercury += 6;
    if (angleMercury > 360) {     angleMercury -= 360;   }

    angleVenus += 2.8;
    if (angleVenus > 360)   {     angleVenus -= 360;     }

    angleMars += 1.5;
    if (angleMars > 360)    {     angleMars -= 360;      }

    angleJupiter += 1.2;
    if (angleJupiter > 360) {     angleJupiter -= 360;   }

    angleSaturn += 1.0;
    if (angleSaturn > 360)  {     angleSaturn -= 360;    }
```



```

    angleUranus += 0.6;
    if (angleUranus > 360)  {     angleUranus -= 360;    }

    angleNeptune += 0.5;
    if (angleNeptune > 360) {    angleNeptune -= 360;    }

    glutPostRedisplay();
    glutTimerFunc(1, update, 0);
}
```

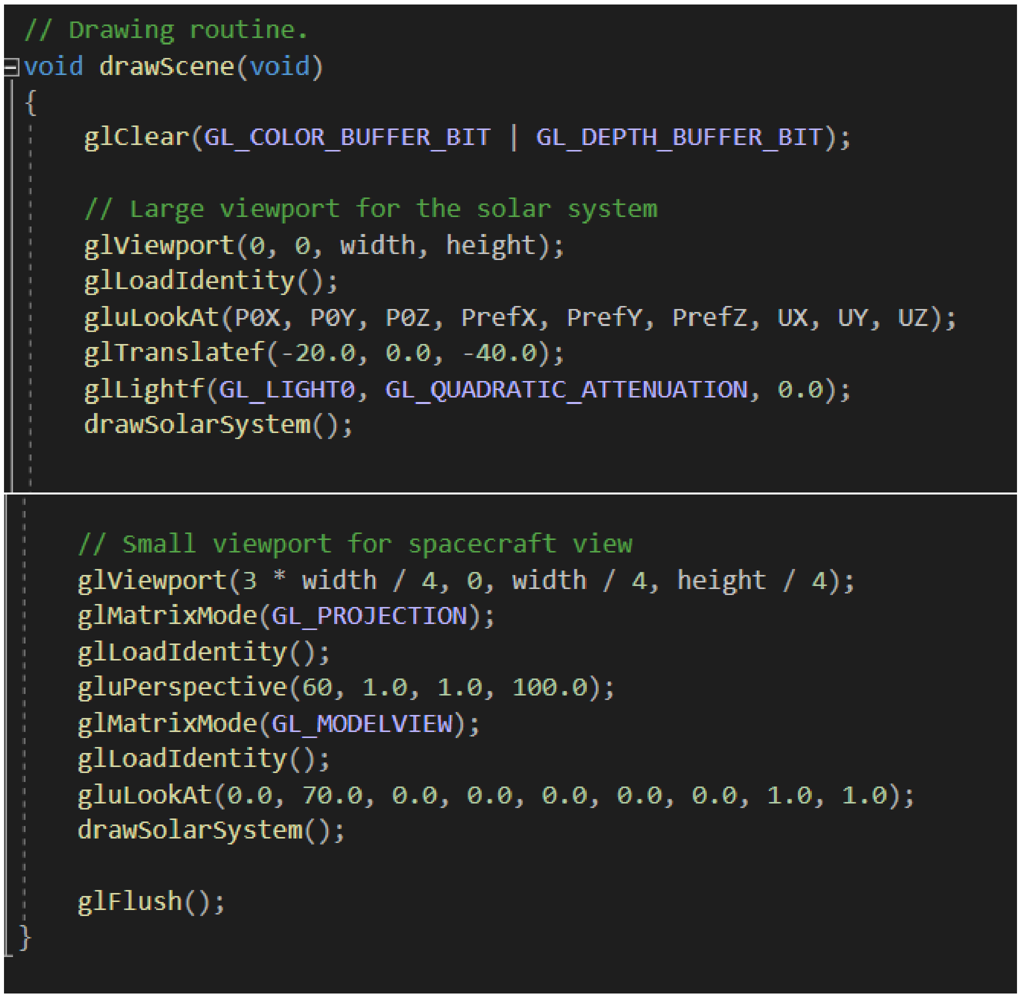
>> User Interaction Function :



```
// Routine to output interaction instructions to the C++ window.
void printInteraction(void)
{
    std::cout << " \n    User Interaction Guide:    " << std::endl;
    std::cout << " You can Do the following to mave the space craft: \n"
              << ">> Pressing x would lead to decrease the x coordinate of the camera (space craft position)\n"
              << ">> Pressing X would lead to increase the x coordinate of the camera (space craft position)\n"
              << ">> Pressing y would lead to decrease the y coordinate of the camera (space craft position)\n"
              << ">> Pressing Y would lead to increase the y coordinate of the camera (space craft position)\n"
              << ">> Pressing z would lead to decrease the z coordinate of the camera (space craft position)\n"
              << ">> Pressing Z would lead to increase the z coordinate of the camera (space craft position)\n"
              << "......... To continue Please Enter (0) to start simulation ......... \n  >> ";
}
```

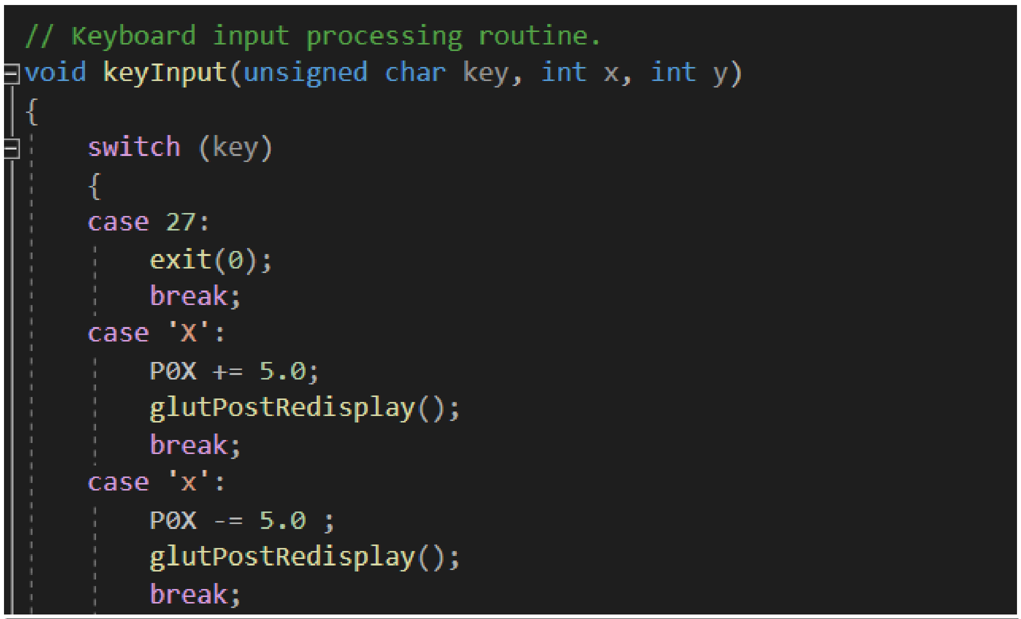
>> Draw Scene Function :

- This function is responsible for construction of the two view ports with suitable coordinates and draw the system for each view port .
- For the second view port has coordinates (3*width/4,height/4) for the bottom left corner and ¼ width and ¼ height of the window , firstly use perspective projection then lookat with the coordinates P_0 = (0.0, 70.0, 0.0) , P_{ref} = (0.0, 0.0, 0.0) , V_{up} = (0.0, 1.0, 1.0) .
- For the first view port has coordinates (0,0) for the bottom left corner and the same width and height as the window ,use look at with coordinates P_0 = (X, Y, Z) , P_{ref} = (P_x, P_y, P_z) , V_{up} = (V_x, V_y, V_z) , X ,Y and Z changes as the user may need to move the space craft .

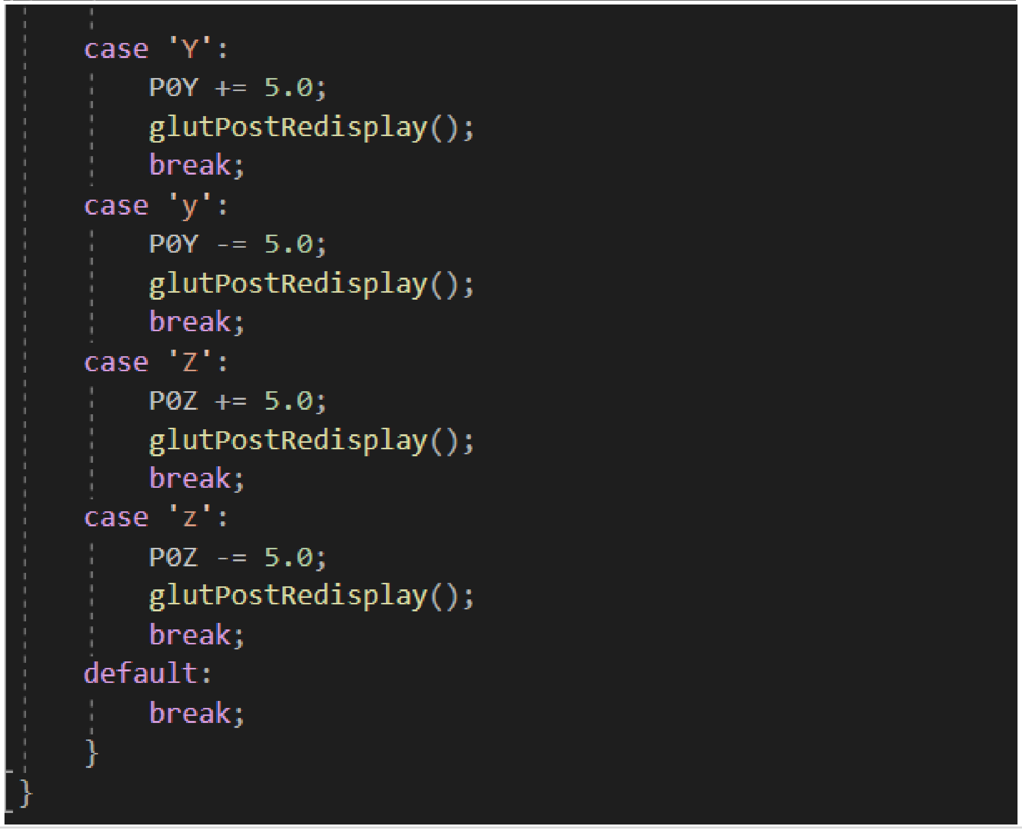

```
// Drawing routine.
void drawScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Large viewport for the solar system
    glViewport(0, 0, width, height);
    glLoadIdentity();
    gluLookAt(P0X, P0Y, P0Z, PrefX, PrefY, PrefZ, UX, UY, UZ);
    glTranslatef(-20.0, 0.0, -40.0);
    glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.0);
    drawSolarSystem();


    // Small viewport for spacecraft view
    glViewport(3 * width / 4, 0, width / 4, height / 4);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, 1.0, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0, 70.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0);
    drawSolarSystem();

    glFlush();
}
```

Key Input Function :

- *Pressing (x , X) (decrease, increase) the x coordinate of the camera .*
- *Pressing (y, Y) (decrease, increase) the y coordinate of the camera .*
- *Pressing (z , Z) (decrease, increase) the z coordinate of the camera .*

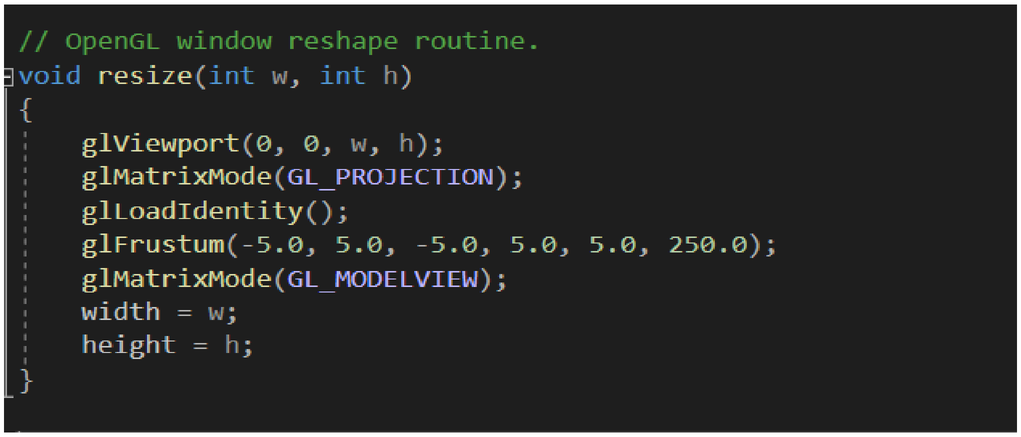


```
// Keyboard input processing routine.
void keyInput(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 27:
        exit(0);
        break;
    case 'X':
        P0X += 5.0;
        glutPostRedisplay();
        break;
    case 'x':
        P0X -= 5.0 ;
        glutPostRedisplay();
        break;
```



```
    case 'Y':
        P0Y += 5.0;
        glutPostRedisplay();
        break;
    case 'y':
        P0Y -= 5.0;
        glutPostRedisplay();
        break;
    case 'Z':
        P0Z += 5.0;
        glutPostRedisplay();
        break;
    case 'z':
        P0Z -= 5.0;
        glutPostRedisplay();
        break;
    default:
        break;
    }
}
```

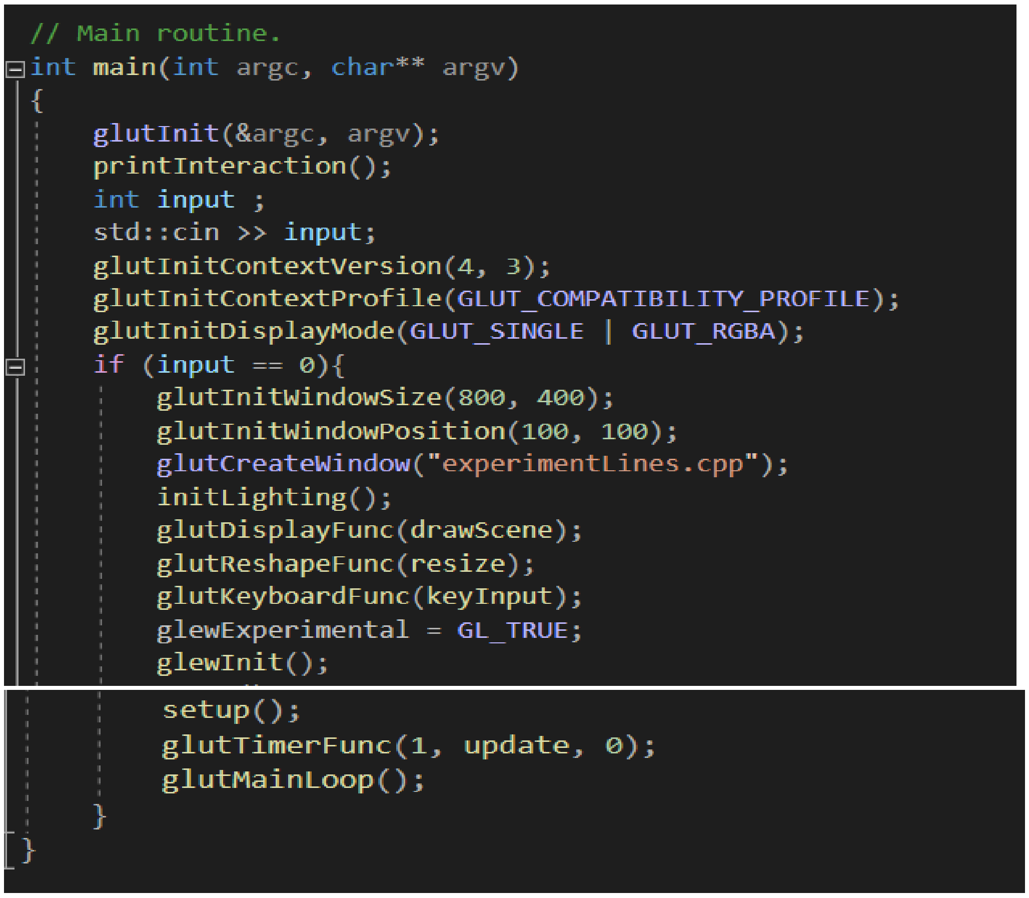
>> Resize Function :



```
// OpenGL window reshape routine.
void resize(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-5.0, 5.0, -5.0, 5.0, 5.0, 250.0);
    glMatrixMode(GL_MODELVIEW);
    width = w;
    height = h;
}
```

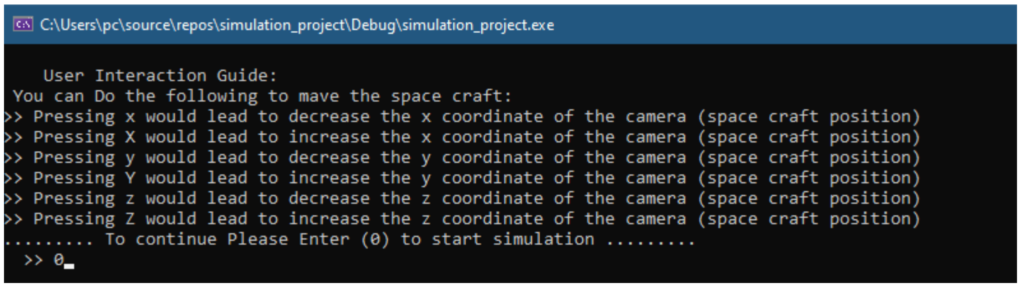
Main Function:

- *User must enter 0 so the program would start .*



```
// Main routine.
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    printInteraction();
    int input ;
    std::cin >> input;
    glutInitContextVersion(4, 3);
    glutInitContextProfile(GLUT_COMPATIBILITY_PROFILE);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
    if (input == 0){
        glutInitWindowSize(800, 400);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("experimentLines.cpp");
        initLighting();
        glutDisplayFunc(drawScene);
        glutReshapeFunc(resize);
        glutKeyboardFunc(keyInput);
        glewExperimental = GL_TRUE;
        glewInit();
        setup();
        glutTimerFunc(1, update, 0);
        glutMainLoop();
    }
}
```

Sample Runs :



C:\Users\pc\source\repos\simulation_project\Debug\simulation_project.exe

```
   User Interaction Guide:
 You can Do the following to mave the space craft:
>> Pressing x would lead to decrease the x coordinate of the camera (space craft position)
>> Pressing X would lead to increase the x coordinate of the camera (space craft position)
>> Pressing y would lead to decrease the y coordinate of the camera (space craft position)
>> Pressing Y would lead to increase the y coordinate of the camera (space craft position)
>> Pressing z would lead to decrease the z coordinate of the camera (space craft position)
>> Pressing Z would lead to increase the z coordinate of the camera (space craft position)
......... To continue Please Enter (0) to start simulation .........
  >> 0_
```