Course project

(Solar System Simulation)

Code description:

- Function to draw the orbits around the sun

```
void drawOrbit(float radius) {
    glBegin(GL_LINE_STRIP);
    for (float i = 0; i < 360.0; i += 0.1) {
        float x = radius * cos(i * 3.1415926535 / 180);
        float y = radius * sin(i * 3.1415926535 / 180);
        //float z = radius * tan(i * 3.1415926535 / 180);
        glVertex3f(x, y,1);
    }
    glEnd();
}
```

- **In function setup():**

```
void Setup()
{

    srand(time(0));
    //Parameter handling
    glShadeModel(GL_SMOOTH);
    glEnable(GL_DEPTH_TEST);

    // polygon rendering mode
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);

    // Black background
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

    //to draw spacecraft
    spacecraft = glGenLists(1);
    glNewList(spacecraft, GL_COMPILE);
    glPushMatrix();
    glRotatef(180.0, 0.0, 1.0, 0.0);
    glColor3f(1.0f, 0.41f, 0.71f);
    glutWireCone(2.0, 3.0, 10, 10);
    glPopMatrix();
    glEndList();
}
```

- **In function drawScene():**

❖ In the drawScene I put **2 viewports**, One for space-craft and the other for the whole solar system

❖ **<u>View Port 1(Space-craft view):</u>**

- First I put the view port to whole screen and put the camera with variables cameraX, cameraY and cameraZ to can change its variables in interaction with user.

```
glViewport(0, 0, Width, Height);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(cameraX, cameraY, cameraZ, PX, PY, 0.0, 0.0, 1.0, 0.0);
```

- Then I enable the lighting for pointer I declared to draw any planrt in solar system

```
if (!animate) {
    glPushMatrix();
    glDisable(GL_LIGHTING);
    glDisable(GL_LIGHT0);
    glColor3f(1, 1, 1);
    glTranslatef(-8, 50, 0.0);
    glPopMatrix();
}
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
GLUquadricObj* planet;
planet = gluNewQuadric();
gluQuadricDrawStyle(planet, GLU_FILL);
gluDeleteQuadric(planet);

glDisable(GL_LIGHTING);
```

- I made stack to draw orbits around the sun

```
glPushMatrix();
glColor3f(0.3f, 0.3f, 0.3f);
glLineWidth(0.1);
glRotatef(-90, 1.0f, 0.0f, 0.0f);
drawOrbit(20);
drawOrbit(40);
drawOrbit(60);
drawOrbit(80);
drawOrbit(100);
drawOrbit(125);
drawOrbit(165);
drawOrbit(200);
drawOrbit(225);
glPopMatrix();
```

- To draw the sun and enable lighting by configuring the lighting properties of light source 0, including its position, diffuse color, and ambient color.

```
//drawing sun
glPushMatrix();
glColor3f(0.8, 0.498039, 0.196078);
gluSphere(planet, 20, 36, 18);
glColor3f(1, 1, 0);
glEnable(GL_BLEND);
glBlendFunc(first, second);
gluSphere(planet, 30, 36, 18);
glDisable(GL_BLEND);
glPopMatrix();

glEnable(GL_LIGHTING);
GLfloat light_position[] = { 0.0, 0.0, 0.0,1 };
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
GLfloat light_diff[] = { 1.0, 1.0, 1.0, 1.0 };
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diff);
GLfloat light_amb[] = { 0.0, 0.0, 0.0, 1.0 };
glLightfv(GL_LIGHT0, GL_AMBIENT, light_amb);
```

- To draw the planets: I made the stack matrix and change the size of the planet and change the rotation_speed to match reality

```
//mercury
glPushMatrix();
glRotatef(x_rotation * 1.6, 0, 1, 0);
glTranslatef(40, 0, -10.0);
glColor3ub(204, 126, 56);
gluSphere(planet, 4, 36, 18);
glPopMatrix();
```

```
//venus
glPushMatrix();
glRotatef(x_rotation * 1.17, 0, 1, 0);
glTranslatef(60, 0, -10.0);
glColor3ub(215, 122, 98);
gluSphere(planet, 5.5, 36, 18);
glPopMatrix();
```

  - The Earth and its moon:

```
//Earth
glPushMatrix();
glRotatef(x_rotation, 0, 1, 0);
glTranslatef(80, 0, -10.0);
glColor3f(0.275f, 0.51f, 0.71f);
gluSphere(planet, 6 , 36, 18);

//moon
glRotatef(x_rotation * 70, 0.0, 1.0, 0.0);
glTranslatef(2, 0, -10.0);
glColor3f(0.75f, 0.75f, 0.75f);
gluSphere(planet, 3, 36, 18);

glPopMatrix();
```

```
//mars
glPushMatrix();
glRotatef(x_rotation * 0.8, 0, 1, 0);
glTranslatef(100, 0, -10.0);
glColor3ub(198, 62, 60);
gluSphere(planet, 4.5, 36, 18);
glPopMatrix();
```

```
//Jupiter
glPushMatrix();
glRotatef(x_rotation * 0.438, 0, 1, 0);
glTranslatef(125, 0, -10.0);
glColor3ub(231, 203, 191);
gluSphere(planet, 16, 36, 18);
glPopMatrix();
```

```
//saturn
glPushMatrix();
glRotatef(x_rotation * 0.325, 0, 1, 0);
glTranslatef(165, 0, -10.0);
glColor3f(0.8f, 0.6f, 0.5f);
gluSphere(planet, 15, 36, 18);
glColor3f(0.8f, 0.6f, 0.5f);
glLineWidth(7);
drawOrbit(18);
glPopMatrix();
```

```
//uranus
glPushMatrix();
glRotatef(x_rotation * 0.228, 0, 1, 0);
glTranslatef(200, 0, -10.0);
glColor3ub(36, 97, 253);
gluSphere(planet, 9, 36, 18);
glPopMatrix();
```

```
//Neptune
glPushMatrix();
glRotatef(x_rotation * 0.182, 0, 1, 0);
glTranslatef(225, 0, -10.0);
glColor3ub(153, 223, 254);
gluSphere(planet, 8.8, 36, 18);
glPopMatrix();
```

❖ View Port 2 (Solar system):

- I do the same as viewport1 but I made the variables constant and I draw space_craft (its color is **hot pink**)

```
//for space craft
glPushMatrix();
glTranslatef(cameraX, cameraY, offset-300);
glRotatef(angle_x, 1.0f, 0.0f, 0.0f);
glRotatef(angleY, 0.0f, 1.0f, 0.0f);
glRotatef(angleZ, 0.0f, 0.0f, 1.0f);
glCallList(spacecraft);
glPopMatrix();
```

- **resize function:**

```
void Resize(int w, int h)
{
    // define the visible area of the window ( in pixels )
    if (h == 0) h = 1;
    Height = h;
    Width = w;
    // Setup viewing volume
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluPerspective(60.0, (float)w / (float)h, 1.0, 1000.0);
    //glFrustum(-5.0, 5.0, -5.0, 5.0, 5.0, 500);
}
```

- **keyboard function:**

- to rotate around x:

```
case 'x':
    angle_x -= 10.0f;
    glutPostRedisplay();
    break;

case 'X':
    angle_x += 10.0f;
    glutPostRedisplay();
    break;
```

- to rotate around y:

```
case 'y':
    angleY -= 5.0f;
    glutPostRedisplay();
    break;

case 'Y':
    angleY += 5.0f;
    if (angleY > 360.0f) angleY -= 360.0f;
    glutPostRedisplay();
    break;
```

- to rotate around z:

```
case 'z':
    angleZ -= 10.0f;
    glutPostRedisplay();
    break;

case 'Z':
    angleZ += 10.0f;
    glutPostRedisplay();
    break;
```

- to zoom in and out

```
case 'O':
    offset += 1;
    glutPostRedisplay();
    break;
case 'o':
    offset -= 1;
    glutPostRedisplay();
    break;
```

- If the user enter space, it toggles between animate or not:

```
case ' ':
    animate = !animate;;
    break;
```

- To change the position of the camera around x axis:

```
case 'w':
    cameraX -= 0.3f;
    PX -= 0.3f;
    break;
case 'W':
    cameraX += 0.3f;
    PX += 0.3f;
    break;
```

- To change the position of the camera around y axis:

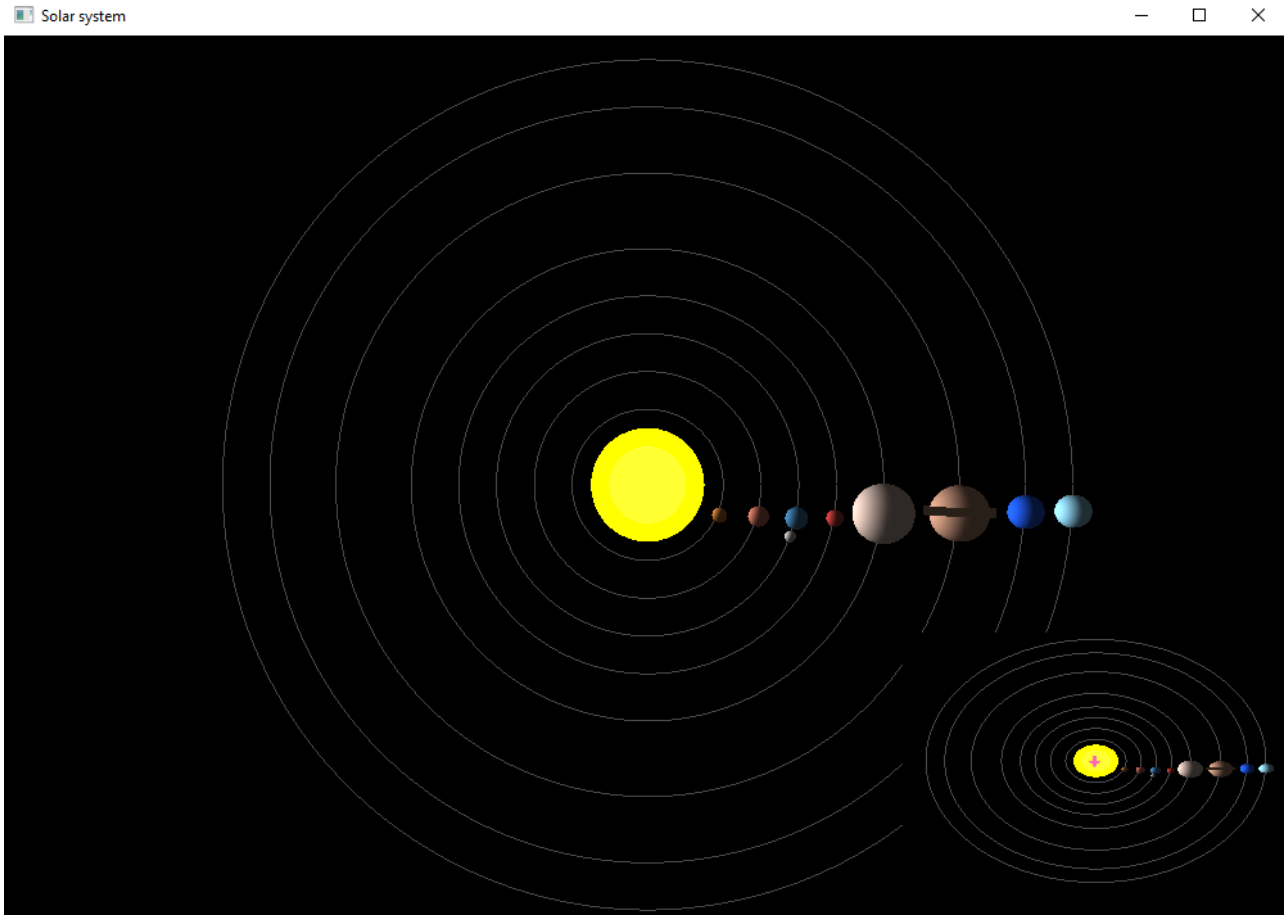
```
case 'h':
    cameraY -= 0.3f;
    PY -= 0.3f;
    break;
case 'H':
    cameraY += 0.3f;
    PY += 0.3f;
    break;
```

- **Main function:**

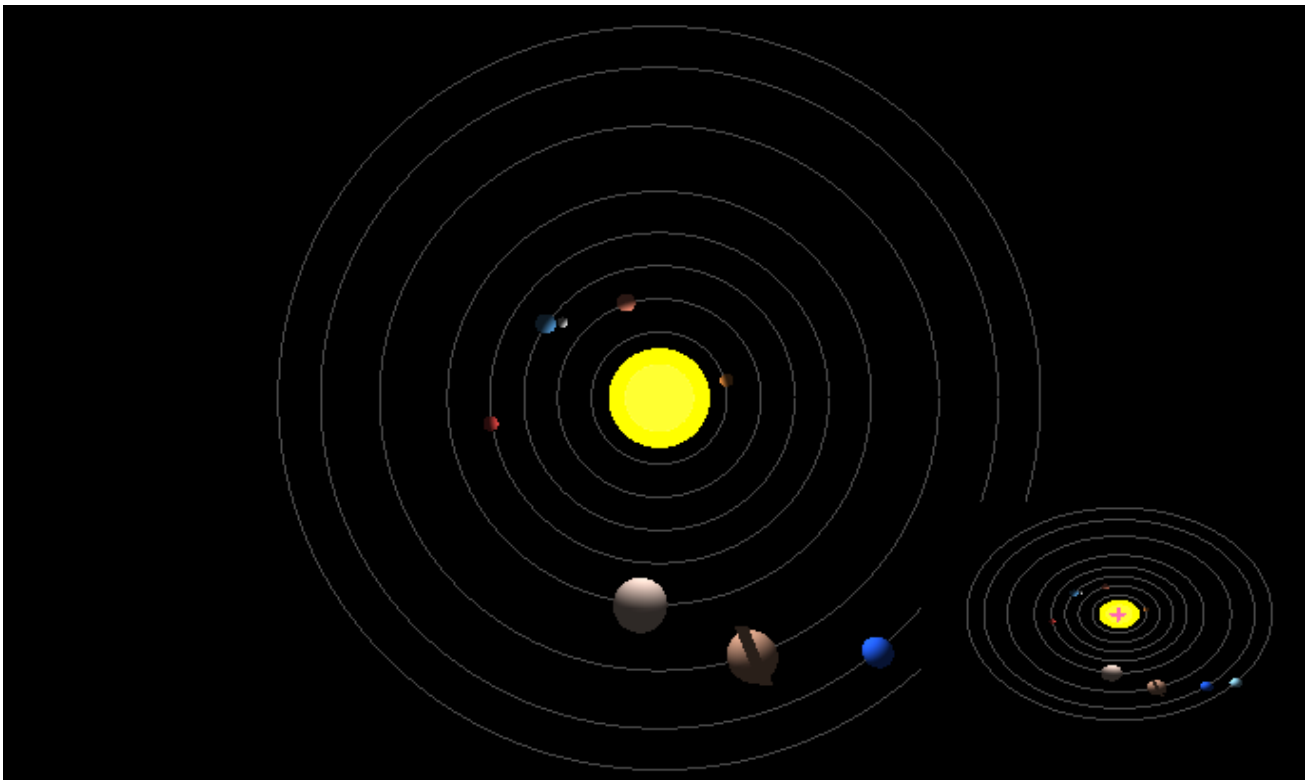
```
int main(int argc, char* argv[])
{
    // initialize GLUT library state
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_DOUBLE);

    // Define the main window size and initial position
    glutInitWindowSize(1000, 700);
    glutInitWindowPosition(50, 10);

    // Create and label the main window
    glutCreateWindow("Solar system");
    Setup();

    // The rendering function
    glutDisplayFunc(drawScene);
    glutReshapeFunc(Resize);
    glutIdleFunc(Idle);
    glutKeyboardFunc(Keyboard);

    //Enter main event handling loop
    glutMainLoop();
    return 0;
}
```

Runs:

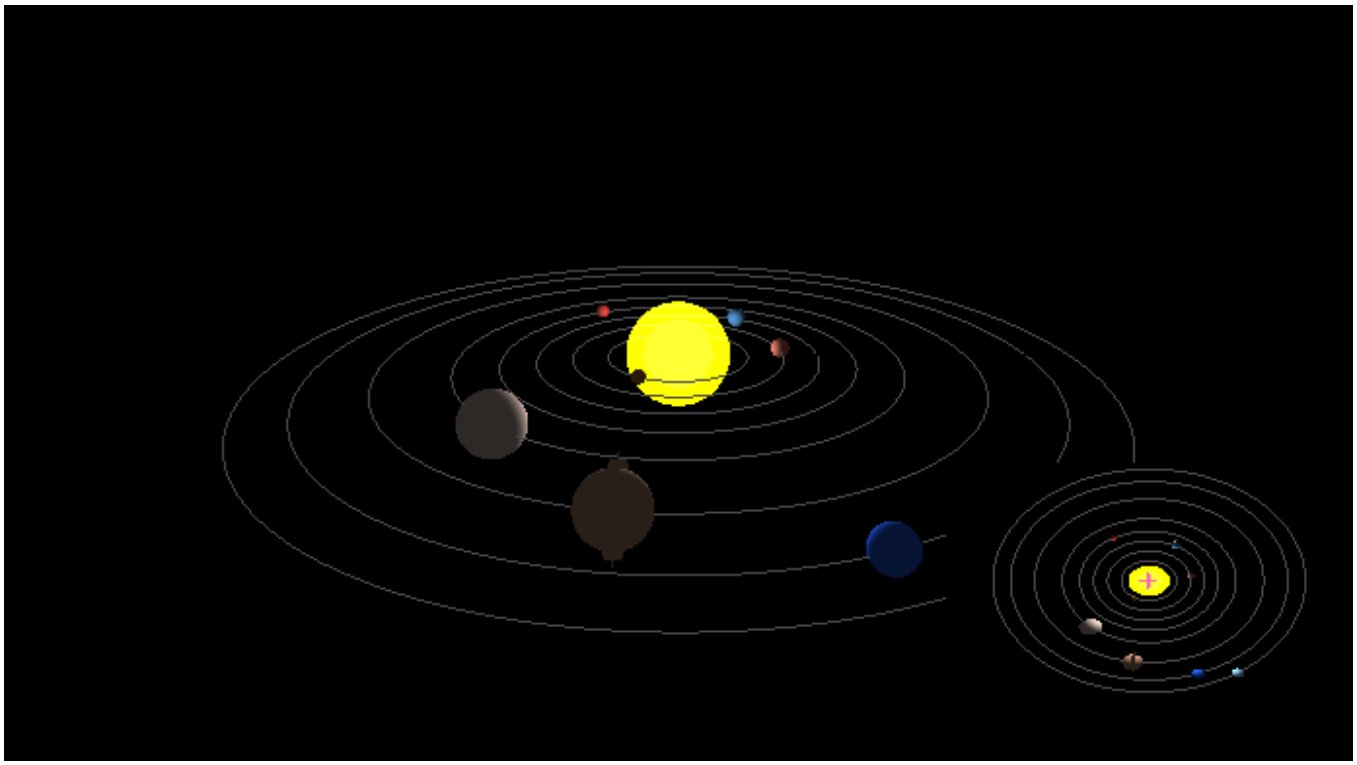
- **First screen:**



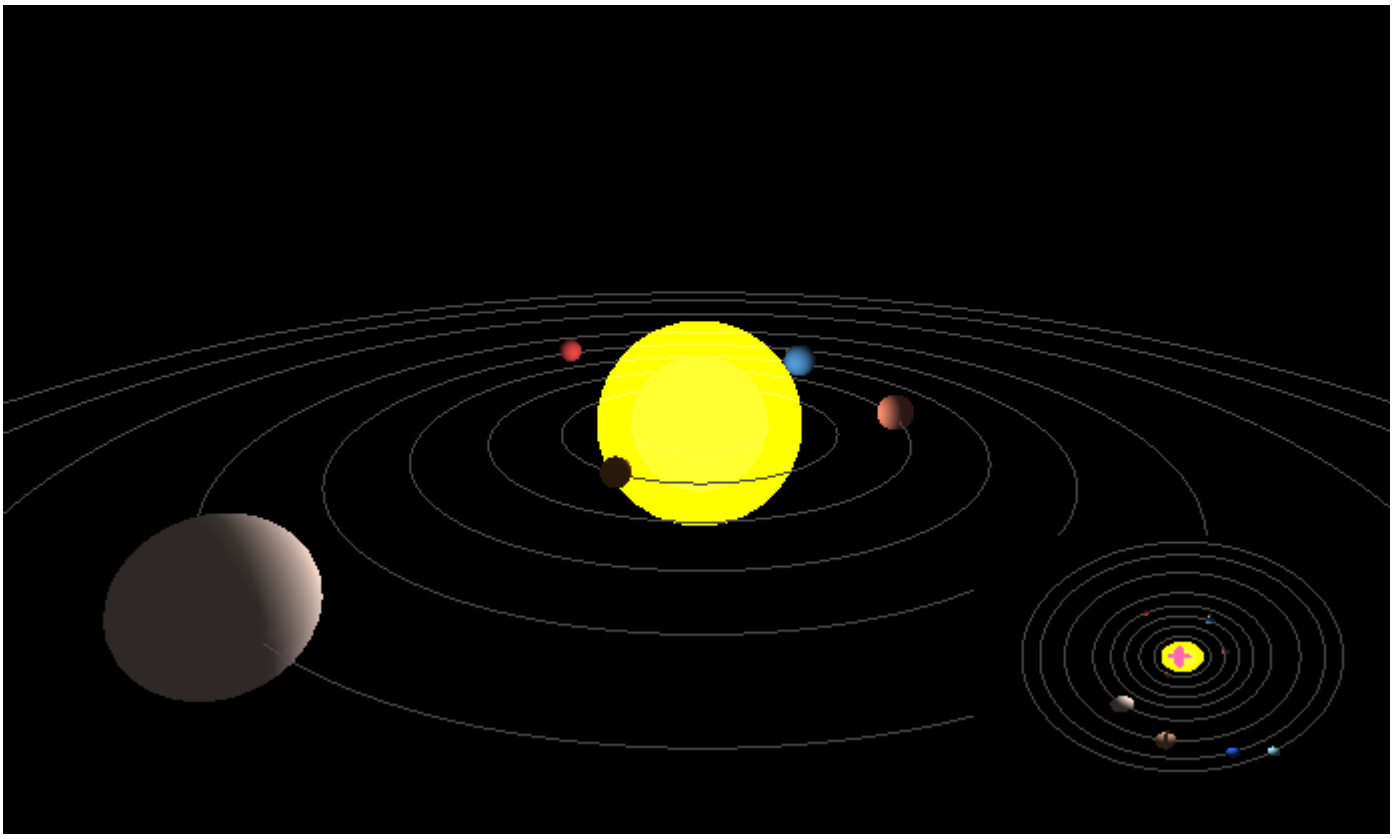
- **Enter space : it start to simulate:**



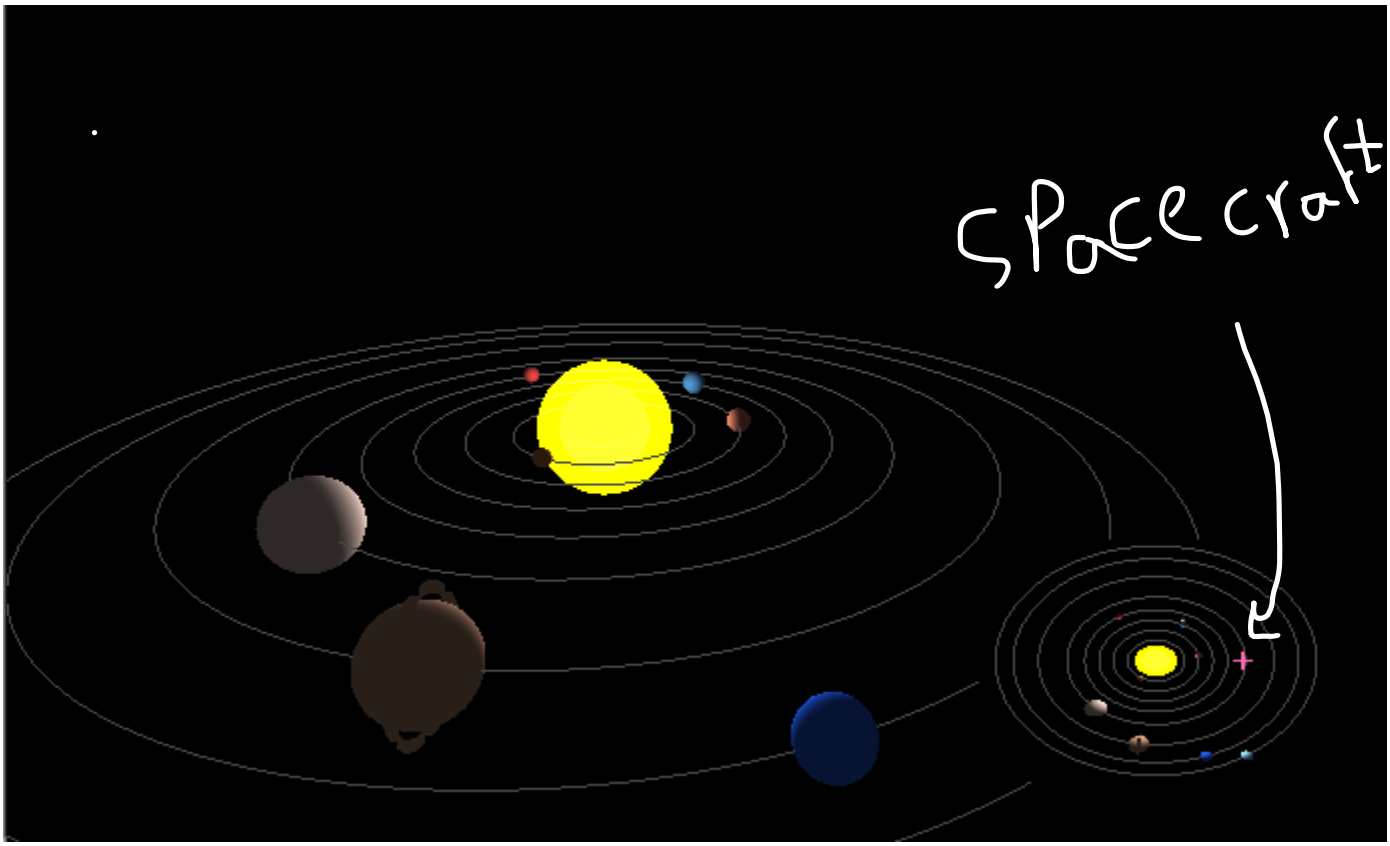
- **Enter X, x:**



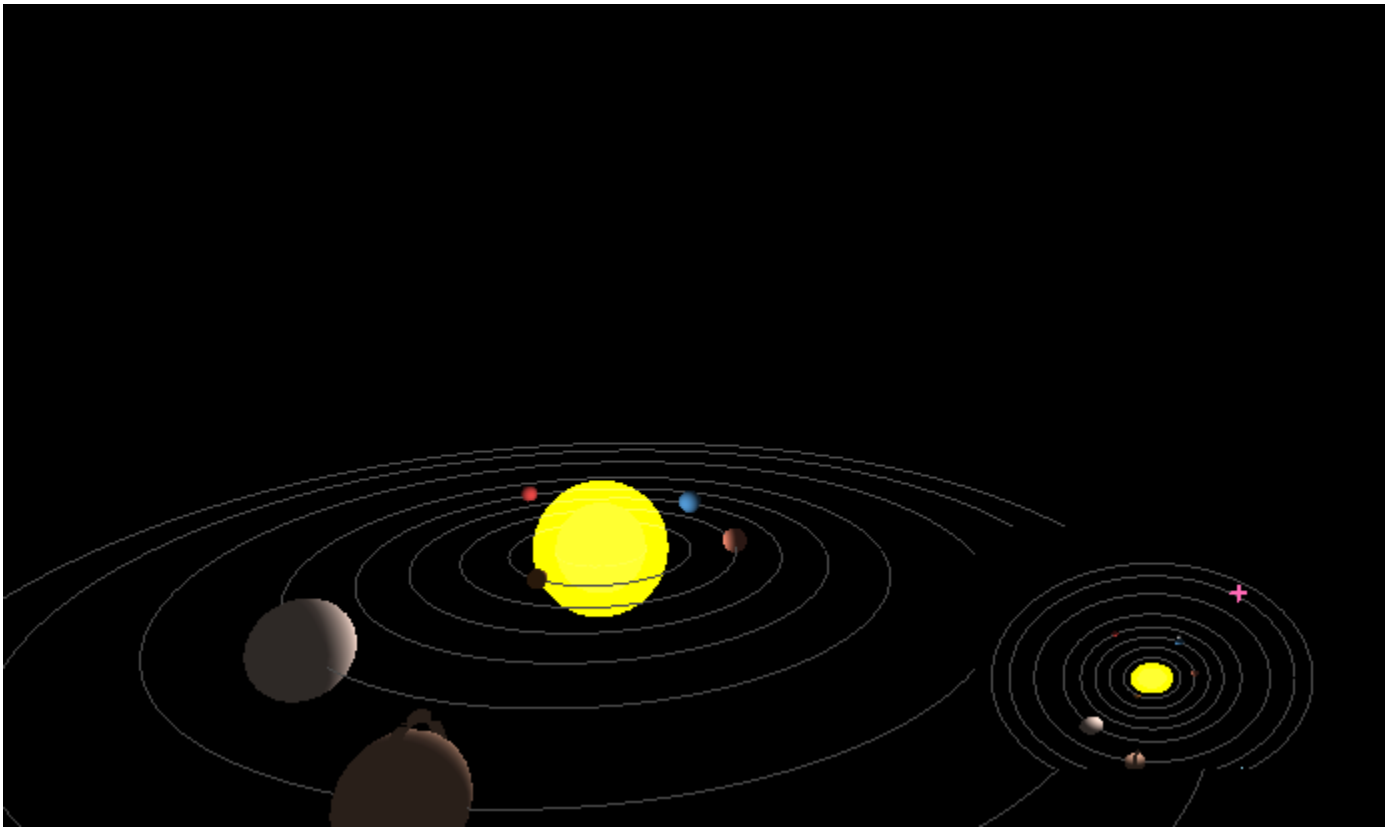
- **Enter O, o:**



- **Enter W, w:**



- **Enter H, h => note the space craft:**



- **Enter Z, z**