CS 450/550 Fall 2020

Final Project: Solar System

Tenghuan Li

liten@oregonstate.edu

Solar System

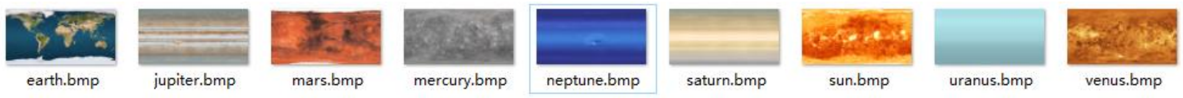
A. Proposal

1. For the Solar system, I will exaggerate the planet diameter and planet orbital radius to make a solar system. For each planet, I will make the planets object's proportion like the actual stars.
2. I will find some good textures or good coloring for the sun and all the planets. Like NASA website.
3. I will set up a suitable observation position to have a good view of all the planets in the solar system. And I will do the point-light lighting from the Sun.
4. I will try to achieve to follow Kepler's Third law of planetary motion:

   Orbital Period is proportional to OrbitalRadius3/2 = pow(OrbitalRadius, 3./2.)
5. I will realize the elliptical orbit of the planet. And pay attention on the Temporal Aliasing (The wagon-wheel-spokes-rotating-backwards effect).

B. What I actually did.

1. I find the texture of each planet and convert them from JPG to BMP. (In the solar system picture files)



2. In this project, I realized the comparison of each planet according to the proportion of the

radius of the real star, and I made the comparison with the earth as the benchmark.

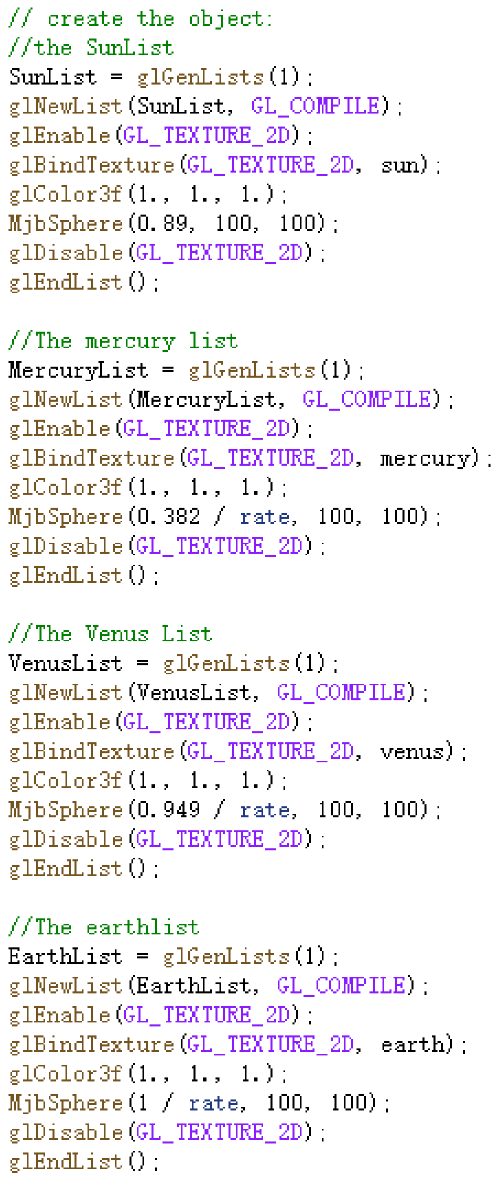
a. Radius of the planet

Mercury	2439.7km	0.382
Venus	6051.8km	0.949
Earth	6378.14km	1
Mars	3397km	0.533
Jupiter	71492km	11.209
Saturn	60268km	9.449
Uranus	25559km	4.007
Neptune	24764km	3.883

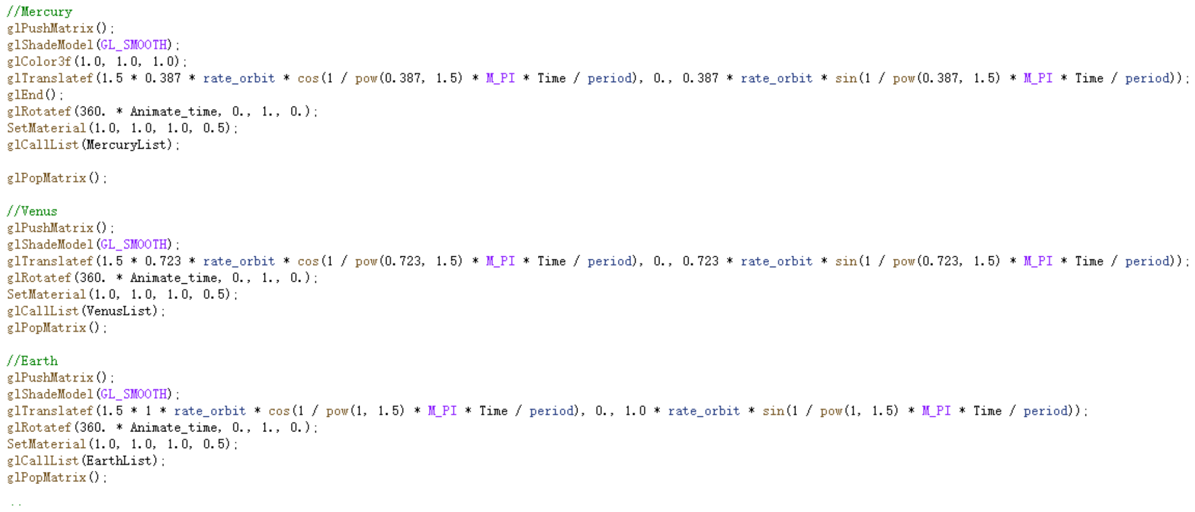
b. The radius from the sun

Mercury	57.91 (Millions of Kilo)	0.387
Venus	108.2	0.723
Earth	149.59	1
Mars	227.94	1.523
Jupiter	778.41	5.203
Saturn	1426.72	9.54
Uranus	2870.97	19.19
Neptune	4498.25	30.07

(PS: Because the sun is so big, I didn't draw it in terms of actual size, I just defined the size arbitrarily)



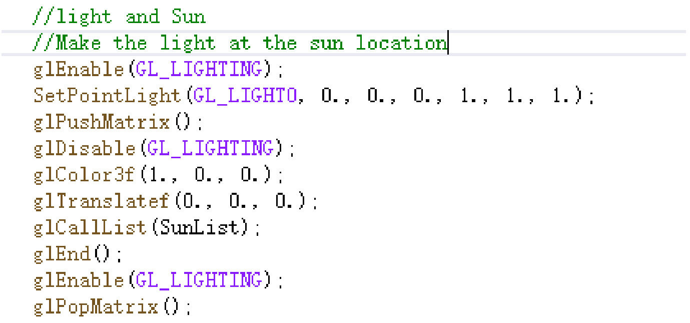
```
// create the object:
//the SunList
SunList = glGenLists(1);
glNewList(SunList, GL_COMPILE);
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, sun);
glColor3f(1., 1., 1.);
MjbSphere(0.89, 100, 100);
glDisable(GL_TEXTURE_2D);
glEndList();

//The mercury list
MercuryList = glGenLists(1);
glNewList(MercuryList, GL_COMPILE);
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, mercury);
glColor3f(1., 1., 1.);
MjbSphere(0.382 / rate, 100, 100);
glDisable(GL_TEXTURE_2D);
glEndList();

//The Venus List
VenusList = glGenLists(1);
glNewList(VenusList, GL_COMPILE);
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, venus);
glColor3f(1., 1., 1.);
MjbSphere(0.949 / rate, 100, 100);
glDisable(GL_TEXTURE_2D);
glEndList();

//The earthlist
EarthList = glGenLists(1);
glNewList(EarthList, GL_COMPILE);
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, earth);
glColor3f(1., 1., 1.);
MjbSphere(1 / rate, 100, 100);
glDisable(GL_TEXTURE_2D);
glEndList();
```



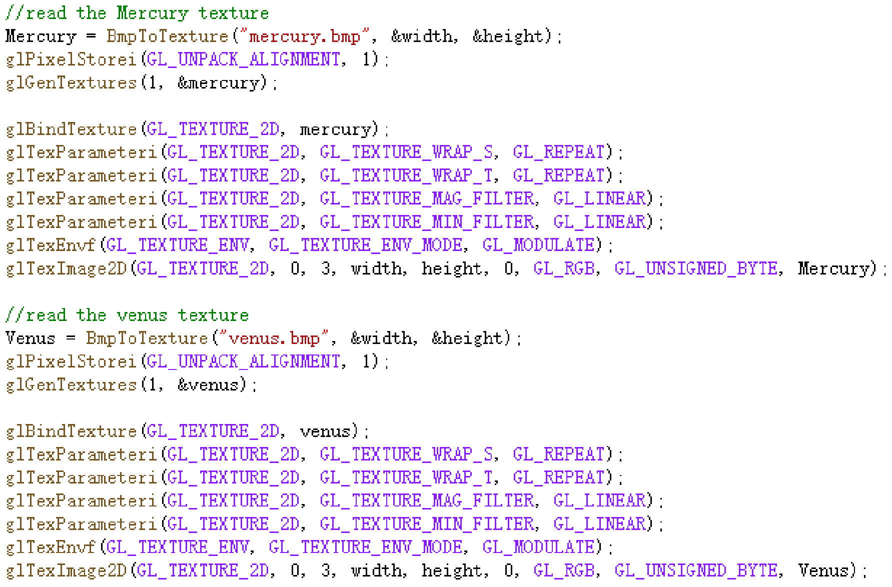
```
//Mercury
glPushMatrix();
glShadeModel(GL_SMOOTH);
glColor3f(1.0, 1.0, 1.0);
glTranslatef(1.5 * 0.387 * rate_orbit * cos(1 / pow(0.387, 1.5) * M_PI * Time / period), 0., 0.387 * rate_orbit * sin(1 / pow(0.387, 1.5) * M_PI * Time / period));
glEnd();
glRotatef(360. * Animate_time, 0., 1., 0.);
SetMaterial(1.0, 1.0, 1.0, 0.5);
glCallList(MercuryList);

glPopMatrix();

//Venus
glPushMatrix();
glShadeModel(GL_SMOOTH);
glTranslatef(1.5 * 0.723 * rate_orbit * cos(1 / pow(0.723, 1.5) * M_PI * Time / period), 0., 0.723 * rate_orbit * sin(1 / pow(0.723, 1.5) * M_PI * Time / period));
glRotatef(360. * Animate_time, 0., 1., 0.);
SetMaterial(1.0, 1.0, 1.0, 0.5);
glCallList(VenusList);
glPopMatrix();

//Earth
glPushMatrix();
glShadeModel(GL_SMOOTH);
glTranslatef(1.5 * 1 * rate_orbit * cos(1 / pow(1, 1.5) * M_PI * Time / period), 0., 1.0 * rate_orbit * sin(1 / pow(1, 1.5) * M_PI * Time / period));
glRotatef(360. * Animate_time, 0., 1., 0.);
SetMaterial(1.0, 1.0, 1.0, 0.5);
glCallList(EarthList);
glPopMatrix();
```

3. I've set the light source for the sun, and I've set each planet to GL_MODULATE so that

there's light in the direction facing the sun, other side is dark when facing away from the sun.

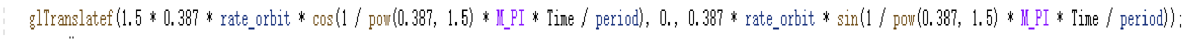


```
//light and Sun
//Make the light at the sun location
glEnable(GL_LIGHTING);
SetPointLight(GL_LIGHT0, 0., 0., 0., 1., 1., 1.);
glPushMatrix();
glDisable(GL_LIGHTING);
glColor3f(1., 0., 0.);
glTranslatef(0., 0., 0.);
glCallList(SunList);
glEnd();
glEnable(GL_LIGHTING);
glPopMatrix();
```



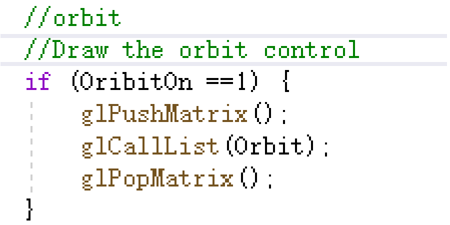
```
//read the Mercury texture
Mercury = BmpToTexture("mercury.bmp", &width, &height);
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glGenTextures(1, &mercury);

glBindTexture(GL_TEXTURE_2D, mercury);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, Mercury);

//read the venus texture
Venus = BmpToTexture("venus.bmp", &width, &height);
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glGenTextures(1, &venus);

glBindTexture(GL_TEXTURE_2D, venus);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, Venus);
```

4. I'm following Kepler's third law to realize the motion of the planet. Using OrbitalRadius3/2 = pow(OrbitalRadius, 3./2.). Each planet rotates at a certain speed, and this is done by rotation

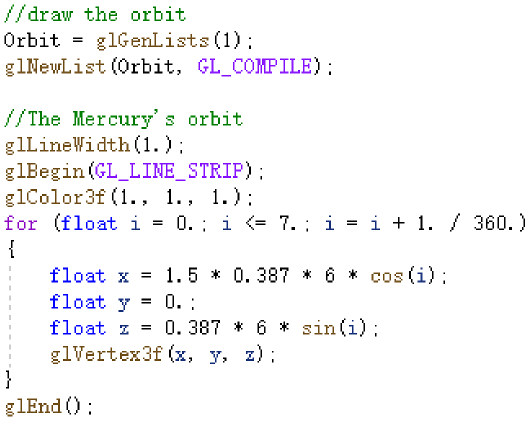


```
glTranslatef(1.5 * 0.387 * rate_orbit * cos(1 / pow(0.387, 1.5) * M_PI * Time / period), 0., 0.387 * rate_orbit * sin(1 / pow(0.387, 1.5) * M_PI * Time / period));
```

5. I set the trajectory switch to control whether or not to show the planet's trajectory. It is call “Oribit”



```
//orbit
//Draw the orbit control
if (OribitOn ==1) {
    glPushMatrix();
    glCallList(Orbit);
    glPopMatrix();
}
```



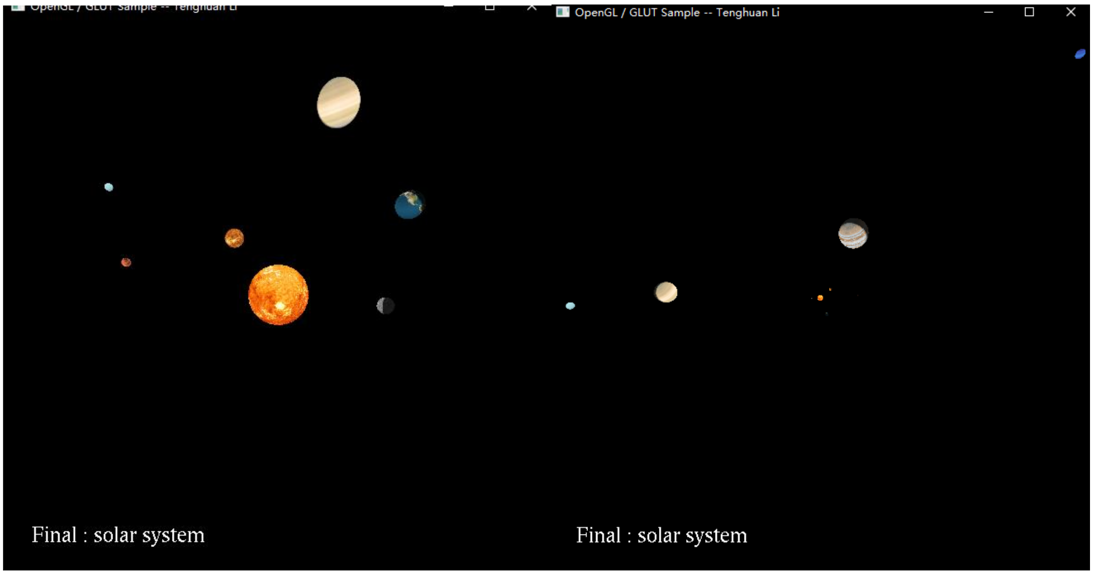
```
//draw the orbit
Orbit = glGenLists(1);
glNewList(Orbit, GL_COMPILE);

//The Mercury’s orbit
glLineWidth(1.);
glBegin(GL_LINE_STRIP);
glColor3f(1., 1., 1.);
for (float i = 0.; i <= 7.; i = i + 1. / 360.)
{
    float x = 1.5 * 0.387 * 6 * cos(i);
    float y = 0.;
    float z = 0.387 * 6 * sin(i);
    glVertex3f(x, y, z);
}
glEnd();
```

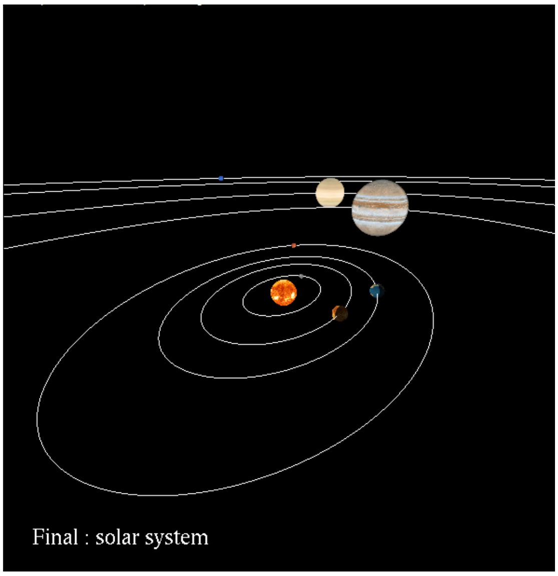
6. According to a certain proportion of the long axis and the short axis, the elliptical planet orbit is realized

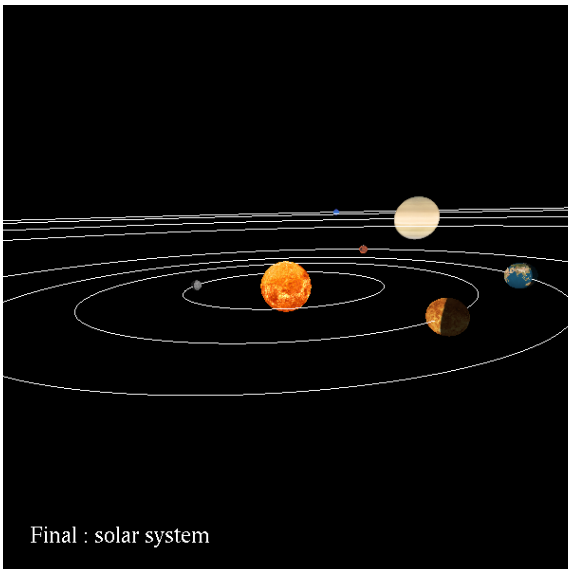
7. This project is not so different from my proposed project, and I have basically achieved all the functions that I hoped to achieve in my proposed project.

8. In this project, I learned some interesting astronomical knowledge. At the beginning, I did not know how to apply Kepler's third Law in my solar system and how to use it. After this experiment, I had a deeper understanding of Kepler's third fixed rate and planet-related knowledge

9. My final project show:


OpenGL / GLUT Sample -- Tenghuan Li
OpenGL / GLUT Sample -- Tenghuan Li
Final : solar system
Final : solar system





10. The video showing

    https://media.oregonstate.edu/media/t/1_o2uxq79p