CS 550 -- Fall Quarter 2015

Introduction to Computer Graphics Final Project Report

Solar System and Cube Earth

Instructor: Prof. Mike Bailey

Name: Kai Shi

ID: 932-504-512

December 2015

Proposal:

In final project, I'm going to simulate the orbits of the solar system planets with OpenGL. This project includes Sun and eight planets. All planets have their own orbits, and they are scaled by real size.

(1) Animate:

All planets will revolve round the sun and also have their own rotations. The rate of rotation speed between each planet will also be decided by the real rate. Use "F" key can begin and stop the animation.

(2) Texture:

All planets and the sun have their own textures. Saturn will have its ring around it. Using right button of mouse can control texture on and off.

(3) Lighting:

Also, the sun is the point light in the simulation of solar system. If texture is off, we can see five shiny, five dull object materials and five flat, five smooth glshademodels. If texture on, lighting will work on texture, all planets will look both textured and lighted.

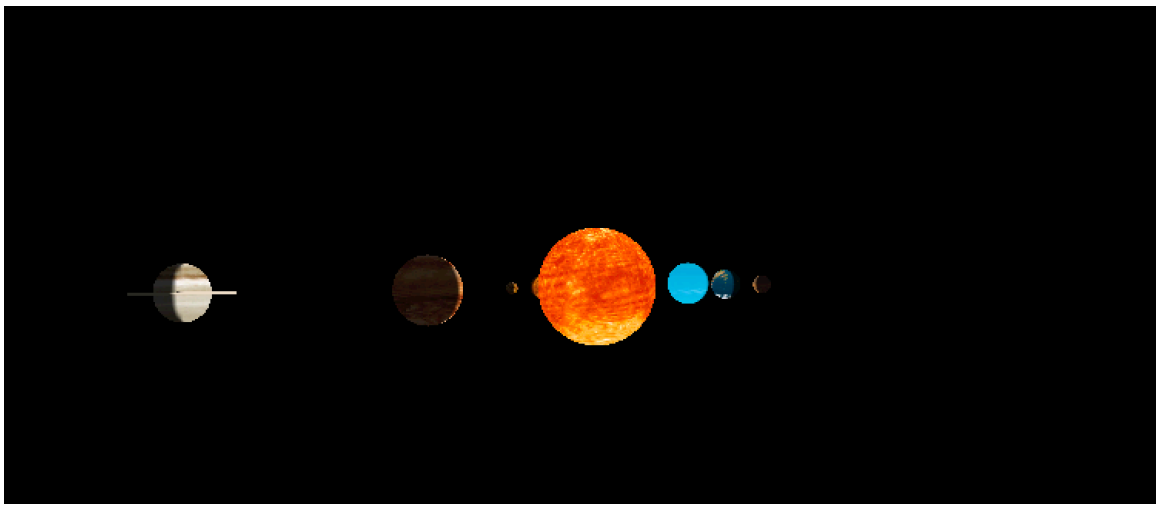
(4) Cube Earth:

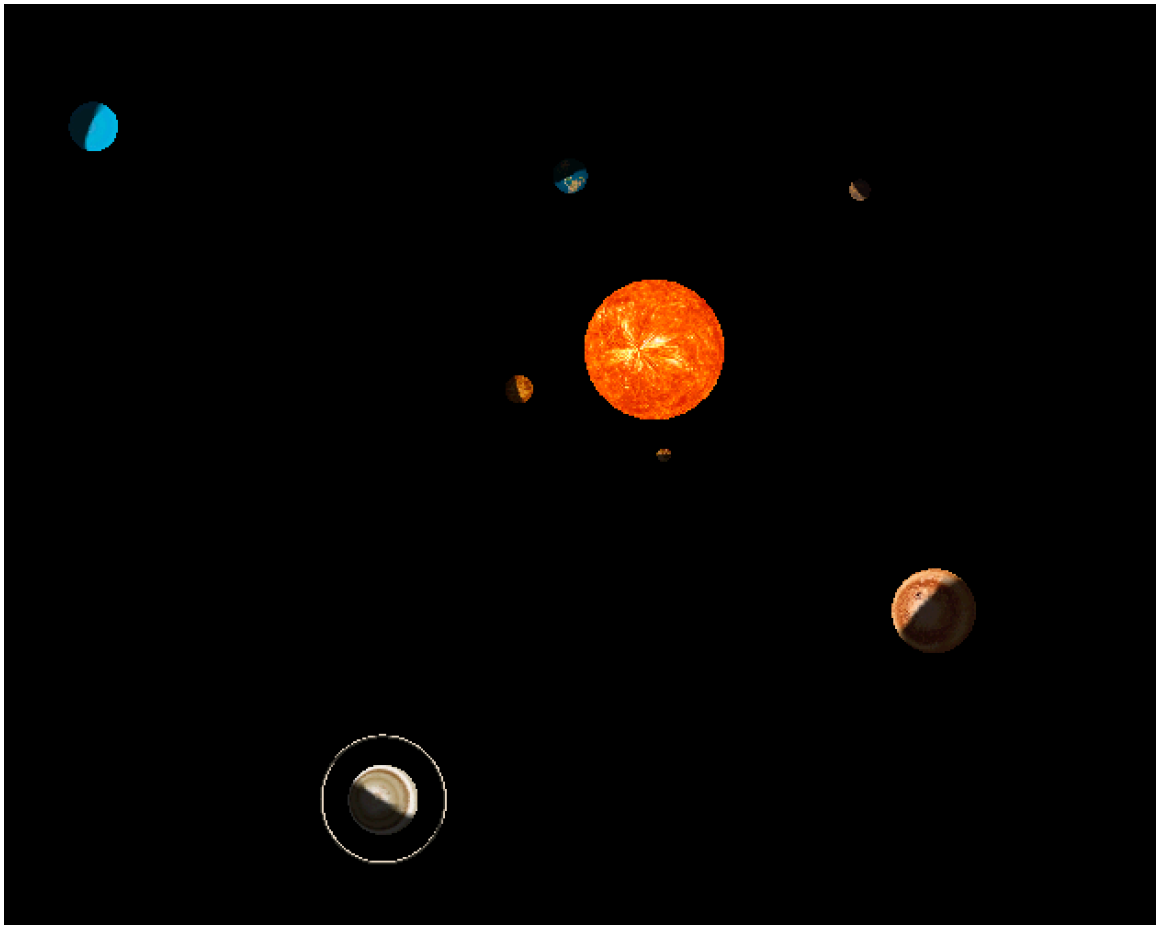
Using right button can change the earth from sphere to a cube. Texture and lighting must be still on the cube of course.

Implement:

In final project, I simulated the orbits of the solar system planets with OpenGL. This project includes Sun and eight planets. All planets have their own orbits, and we can see the day and night of every planet. My project includes four parts: Animation, texture mapping, lighting and funny part.

Here are results:





(1) Animate:

All planets revolve round the sun and also have their own rotations. The rate of rotation speed between each planet is decided by the real rate. That means in real solar system, Mercury revolves fastest among eight planets. I use MS_PER_CYCLE that specifies the number of milliseconds per animation cycle to control the rotation speeds in the Animate function.

My solar system also has the same rate with the real model. The rate of rotations also comes from real rate between these planets. Using "F" key can begin and stop the animation.

First, draw one sphere, then, glRotated it. This rotation is the own rotation of this planet. Then, translating it to its orbit, like `glTranslated(10.6, 0, 0)`. After doing this, we rotate it around (0,1,0) again. This planet revolves round the sun (0,0,0). The distance between sun and every planet is exaggerated by the real distance.

My solar system follows Kepler's law. That means the square of the period of a planet's orbit (e.g., 365 days for Earth) is proportional to the cube of its orbital radius (e.g., 93M miles for Earth).

I write a function

```
float Kepler(float earthR,float planetR,float earthT)
{
    float planetT;

    planetT = sqrtf(powf(planetR, 3.)*powf(earthT,
2.)/powf(earthR, 3.));
    return (2*M_PI/planetT);

}
```

This function uses the Kepler's law by:

earth Radium * earth T = other planet Radium * planet T

So, planet T = sqrtf(powf(planetR, 3.)*powf(earthT, 2.)/powf(earthR, 3.)). This function returns the speed of rotation around the Sun. Then, every planet's rotation around the Sun is like this : `glRotated(Kepler(8.5, 17.45, 2*M_PI/earthV),0,1,0);`

and the Earth's rotation is just earthV.

(2) Texture:

All planets and the sun have their own textures. Saturn will have its ring around it. Using right button of mouse can control texture on and off.

I use BmpToTexture function in project3, and load these nine spheres' textures. All dimensions of these texture images are powers of two. Also, glTexEnvf mode are set to GL_MODULATE for lighting.

(3) Lighting:

The sun is the point light in the simulation of solar system. If texture is off, we can see five shiny, five dull object materials and five flat, five smooth glshademodels. If texture on, lighting works on texture, all planets look both textured and lighted.

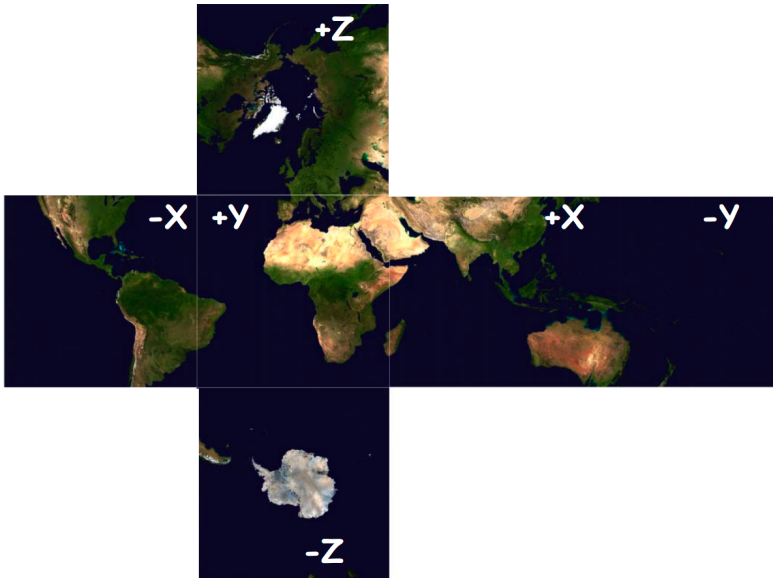
(4) Other:

Using right button can change the earth from sphere to a cube. Texture and lighting still be on the cube of course.

In this part, I use glBegin(GL_QUADS) to draw six faces of the cube. Also, in this step I set (S,T) coord2f. Here is the example to draw the top face:

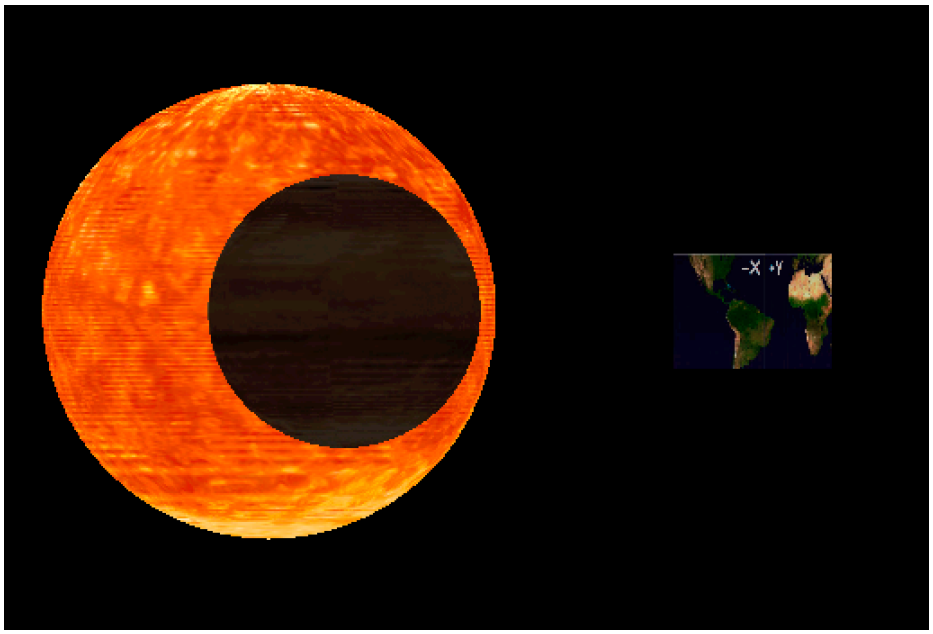
```
glBegin(GL_QUADS);
        //top
   glTexCoord2f(0.0f, 1.0f); glVertex3f(-0.5f,  0.5f, -0.5f);
   glTexCoord2f(0.0f, 0.0f); glVertex3f(-0.5f,  0.5f,  0.5f);
   glTexCoord2f(1.0f, 0.0f); glVertex3f( 0.5f,  0.5f,  0.5f);
   glTexCoord2f(1.0f, 1.0f); glVertex3f( 0.5f,  0.5f, -0.5f);
glEnd();
```

Then, I load the texture of every face. All textures are like this:



(Picture comes from Professor Mike Bailey's handouts of CS 457 / 557 -- RenderMan and OpenGL Shaders, http://web.engr.oregonstate.edu/~mjb/cs557/Handouts/cubemapping.1pp.pdf)

Here is the cube Earth:



Things different from proposal:

In proposal, I'm going to make all planets be scaled by real size, but when realize it, Mercury, Venus and Earth would be too small to be seen because they are too tiny compared with Sun or Jupiter and Saturn.

So, I have to exaggerate the planets' diameters. The diameters only show which one is the bigger than another one, and it's not the real scaled size.

In my proposal, I'm not going to apply the Kepler's Law. But in my project, I used this law because it makes my model more realistic.

What I have learned:

I have learned more about how to texture mapping. Images must be changed to bmp format and image's pixel dimensions to be a power of two. Also, cube drawn by opengl built-in function doesn't have S.T. coordinates. So, I draw every face of a cube and then use the texture mapping on it.

When using the lights, I need to change the Texture Environment mode to GL_MODULATE, which allow the light shines up through a texture.