Computer Graphics
Sphere Solar System

You are to use WebGL and JavaScript to make a program that recreates the solar system. The focus of this assignment will be geometric object creation, lighting and texture mapping.

Starting code:

Please use your Cube Solar System code as the starting code for this assignment.

Viewport control:

You should first add a button that allows you to switch between split viewport (as you had in the cube solar system), map only view, and ship view.

Geometric Object Creation:

In the cube solar system, everything was a cube. In this version you will have spheres and rings.

Instead of having a cube, the Planet class will use a UV sphere instead. I specifically want a UV sphere rather than a repeatedly sub-divided sphere. As discussed in class, a UV sphere is a series of triangle strips (like a cylinder) stack on top of each other. You should be able to easily change the number of strips used to form the sphere. It is probably easiest if you start by just changing the geometry (vertices) and assign a fixed color to your sphere's vertices.

You will also want to add a Ring class. The Ring class should take (among possible other parameters) an innerRadius and an outerRadius. You should create the vertices for the ring using these values. The best approach is to use a single triangle strip to draw the ring. It is a bit like a flattened cylinder.

Saturn needs to have a Ring around it. You should add an attachRing method to the Planet class so that Planets can have a single ring added (much like you added moons to planets).

The Ring should also be used to provide orbit paths for the Planets. Create a button that allows you to turn on and off the orbit paths. Each Planet should have a Ring created for its orbit path that you will either render or not depending on the state of the button. Orbit paths are Rings that have nearly the same inner and outer radius that is out at the orbit

distance. Note that orbit paths are not relative to the Planet's position/rotation/etc. but instead relative to the Planet it is orbiting.

Note: In all the example labs, I use a single shader pair for all rendering. That is acceptable for this assignment as well. The alternative is to have a different shader pair for each type of object you draw (Planet and Ring). If you wish to code your solar system this way, you can as well.

Lighting:

Next you will need to add lighting and shading to your solar system. The light will be fixed at the origin (sun's position). Just like in lab5, you will need to implement the lighting in the vertex shader (Gouraud shading of the Phong lighting model). This implies that you will need to separate out the projection, view, and model matrices in your JavaScript code and send them to the shaders separately. You will also want to send material and light information from your JavaScript to the shaders.

You will need to change the sending of colors to the sending of normals in order to make the lighting calculations work. Sphere normals should be easy to calculate. Ring normal should point up.

Keep in mind that in the real solar system there is basically no ambient reflected light (thus, the dark side of the planets is very dark). Feel free to add some ambient to make your spheres easier to view.

One issue that you will have to deal with is emissive objects. That is, objects that glow on their own rather than being lit by other lights. Emissive objects are typical of objects that represent light sources in our scene (i.e. the Sun). Additionally, you may make all your Rings emissive objects as well to simplify the assignment. Your Ship should also be emissive.

To make lighting emissive, you should only include (a fairly bright) ambient component when doing lighting calculations. That is no diffuse or specular components are used in the final color computation.

Texture Mapping:

Lastly, you will add texture mapping to planets. I have provided images for the planets and moon. Just like in lab6, you will have to load in the texture images and do the texture mapping in the fragment shader. You will also need to provide texture coordinates. Texture coordinates are easy to compute for UV spheres. No texture mapping is necessary for the Ring or Ship.

Submission:

You are to turn in a ZIP of your project directory to me by the due date on canvas. Included in your ZIP file should be a README that acts as a cover sheet for the project. It should include your name(s) as well as the current state of your project (i.e. what works and what doesn't).

You may work alone or with a partner.