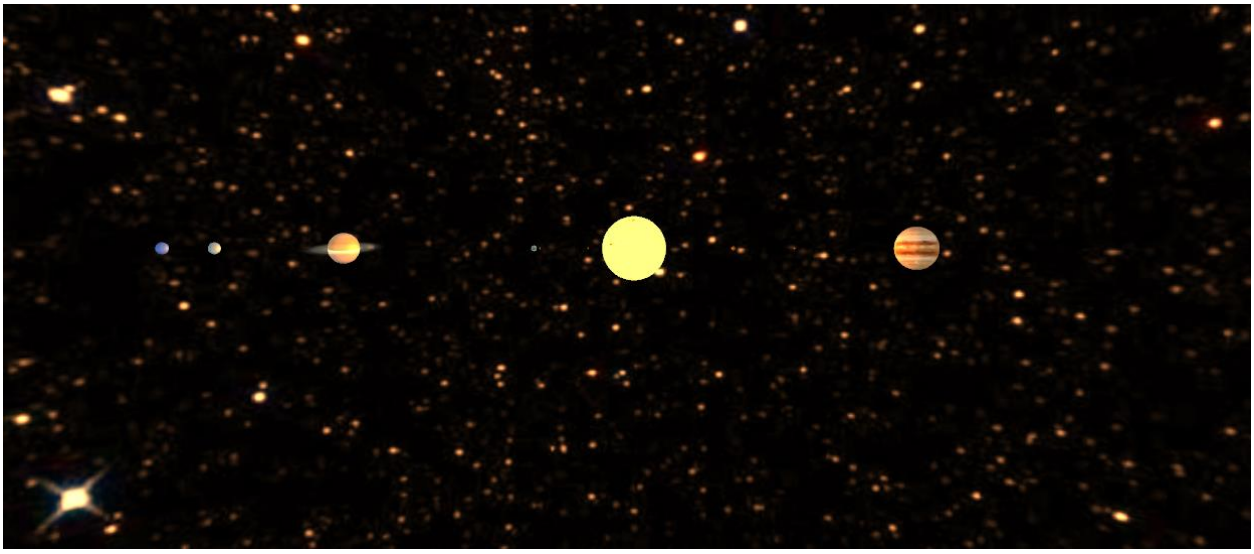
Virtual Orrery – Assignment Report

An orrery is a model of a solar system that illustrates the positions and rotations of the solar system. Our task was to create a virtual orrery using only native WebGL. I completed all the minimum requirements of the task and added a few extensions as well. This report will detail the requirements, how I fulfilled them and what extensions I added.

Description of Program

I completed all minimal requirements of the assignment with some extra functionality. Each of the planets are contained in their own .js files as I prefer the separation of the data and I found it made it easier to add the moon as a satellite for Earth and the rings to Saturn. The sun is rotating at the origin point and contains a point light. The light uses the Phong shading model and I have coloured the light slightly yellow/orange as I found the white light looked slightly unrealistic and wasn't as pleasant to look at.

I initialize the perspective to give a good view of the orrery and then initialize the buffers and shaders for each planet. While having separate buffers and shaders for each planet may have increased the amount of repetitive code in my program, it meant I could separate out the matrix operations and drawing code to each .js file. I found this made it easier to debug any problems I had when writing the program.

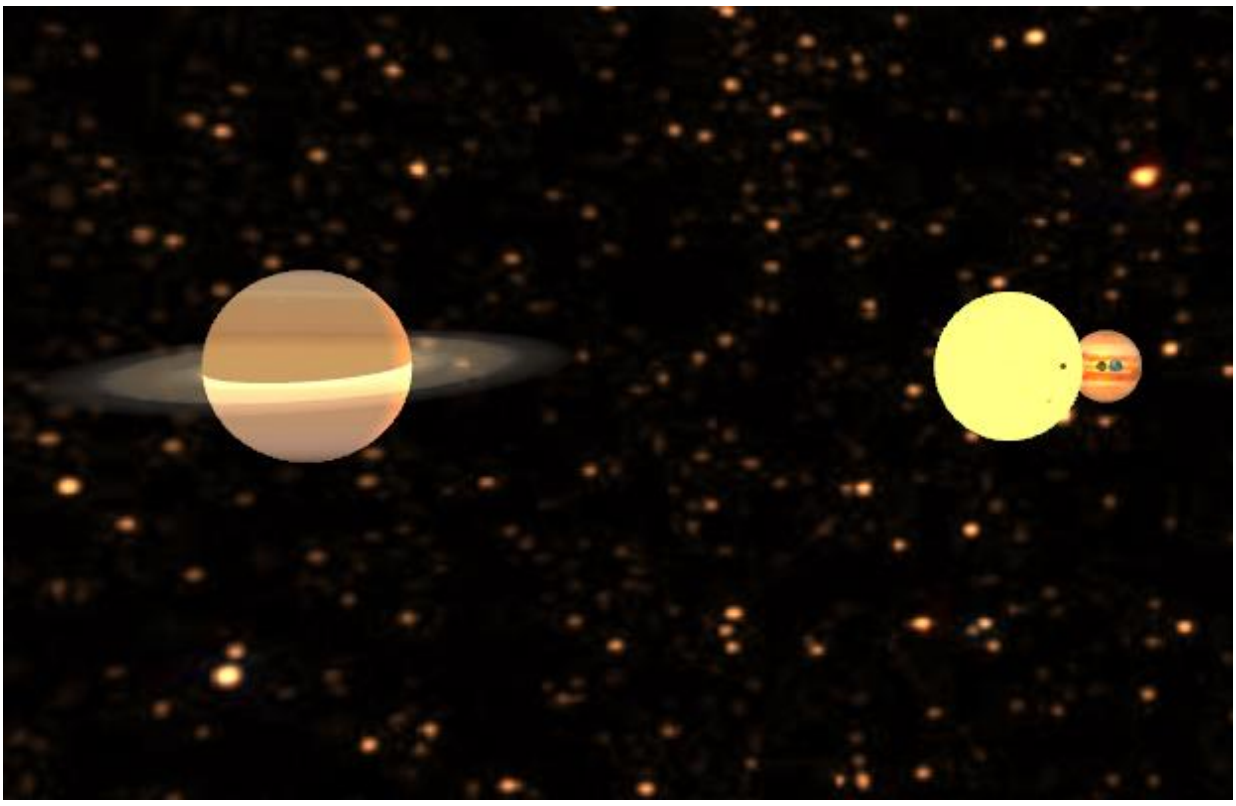


View of Solar System

EXTRA FUNCTIONALITY

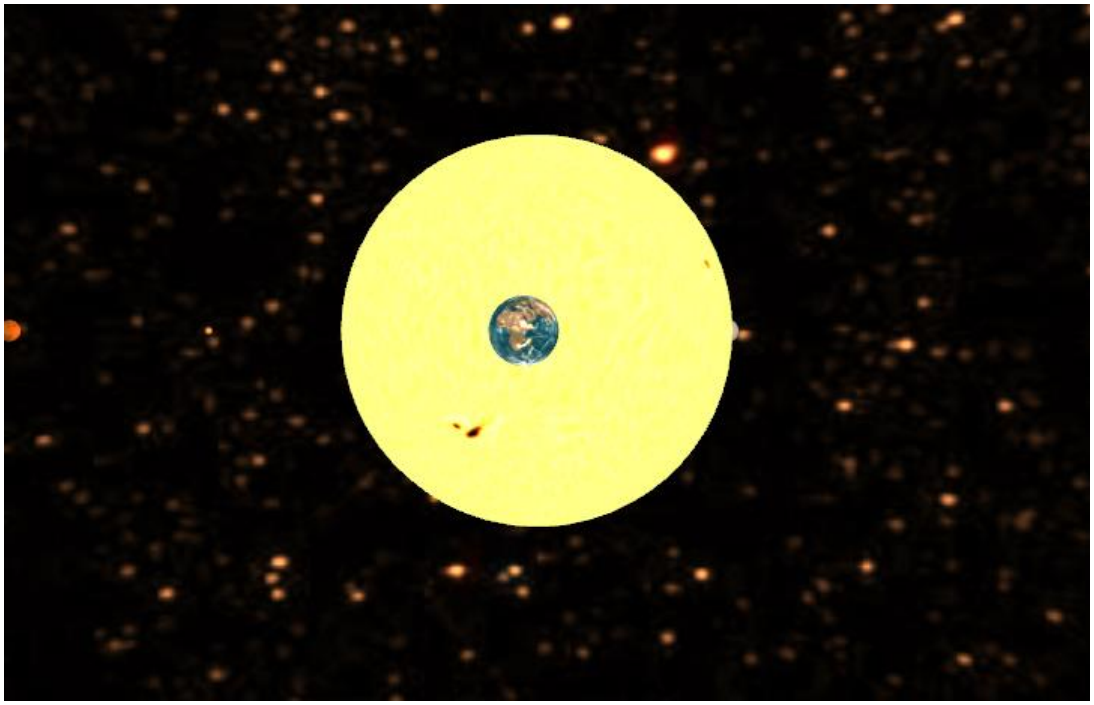
I have included the ability to zoom and rotate the orrery so that you can move to a better view of the solar system. I have included a reset button as I found that it can be easy to get to a position where you cannot see much of the solar system and hard to return to a position where you can. I have created a star sphere as a background to the solar system to make it seem more realistic.

I have drawn Saturn's rings on a plane and given it the same orbit as Saturn so that it remains in the same place. I found it difficult to attach it to the matrix that Saturn was using because it added the rotation to the plane, which was an effect that I didn't want. I used the blend function to make the rings transparent but found I had to draw the rings in a specific place in the order of the planets so that the blending didn't affect any of the other planets when Saturn passed by.



Saturn's rings

I used multi-texturing on the earth to give it a layer of clouds. I found a cloud map that has an alpha channel and then blended the two textures together on the earth. I found that a similar problem occurred to Saturn's rings when I didn't draw the planet in the correct place in the order. The earth does seem to look like a transparent sphere but the cloud map makes the earth look more realistic.



Earth with clouds, created with multi-texturing

I also included elliptical orbits and tried to make the scale of the planets as close to realistic as possible. The elliptical orbits were done using the formula provided in the assignment brief and I used the NASA fact sheets to create realistic sizings. I also changed Venus and Uranus' axis rotations to fit how they rotate within our solar system.

$$r = R(1+e)/(1+e*\cos(\theta))$$

$$\delta\theta = \delta t*R*R*a/(r*r)$$

$$\theta \mathrel{+}= \delta\theta$$

PROBLEMS ENCOUNTERED

I haven't placed boundaries on the zooming as I wasn't sure how to place boundaries using the perspective matrix. This does mean that you can go outside the bounding sphere and see the star sphere from the outside. This is something I feel I should have looked at more and fixed.

As mentioned in the previous paragraph, I had some problems with the transparency of the rings of Saturn and the Earth's textures. Most of this was solved by drawing the planets in a certain order.

I had some trouble with matrix multiplications and working out where to reset the matrix to an identity matrix. I realized after doing most of the matrices that I could have used the push and pop functions to make it easier, but I found this way helped me understand how the multiplications were happening more and made it easier to do the elliptical orbits.

CONCLUSION

Having had no experience in WebGL before this assignment, it has definitely been a significant learning experience in both WebGL and the theory behind graphics programming. I found the mathematics hard at first and decided to use the matrix multiplications in my program directly so I could further my understanding of the mathematics. By using multiple tutorials on WebGL and conferring with my fellow classmates, I found that I managed to get further with this assignment than I originally thought and I am pleased with how the virtual orrery has turned out.

REFERENCES

Learningwebgl.com, (n.d.). Learning WebGL. [online] Available at: http://learningwebgl.com/blog/

Mozilla Developer Network, (n.d.). Getting started with WebGL. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/WebGL/Getting_started_with_WebGL/