Final Project Report

Patrick Chapman and Jacob Margrave

The Initial Proposal

For our final project, we propose to make a visual representation of the Solar System using WebGL as our base language. We may also use other languages such as JavaScript or HTML if the need arises. Since the distances between the planets are vast and the sizes of each planet seemingly small in comparison, we do not expect these figures to be entirely accurate near the conclusion of this project (though it would be rather cool to be able to incorporate these into our final design). Our main aim is to represent their orbital motions around the Sun, as well as represent the planets and all their glory using the computer graphics techniques that we have been learning throughout the duration of this class. Here is how we plan to go about this (note: the order in which we do these tasks may not be in this order):

1. Create spherical objects to represent each planet: this seems like the first logical step for incorporating our design. Since hard-coding the coordinates for each of these objects would simply take too long, we will have to use other techniques (such as loops or other language functions) to generate each spherical shape, whilst ensuring that each object does not use too many coordinates, since using too many object coordinates would slow down the program.

2. Texture each planet with real-color maps and bump maps: Each planet will, of course, have textures representing their actual appearances based on imagery obtained from various spacecraft that have flown by them during the course of the space age. Solid planets, like Mercury, Mars, and Earth, will use bump maps to simulate the various mountains, valleys, and craters that dot their surfaces, whilst gas giants, like Jupiter and Saturn, will not require any bump mapping since they do not have solid surfaces. If time permits, we may decide to use specular light mapping on Earth to bring out the oceans and even use some other texture map technique to bring out the city lights on the night side of Earth.

3. Lighting: For this part, we plan on creating a single source light to represent the Sun, as well as using a dark ambience to simulate the real environment of space and both day and night on each of the planets. This may or may not require some extra ingenuity on our part, but we shall see.

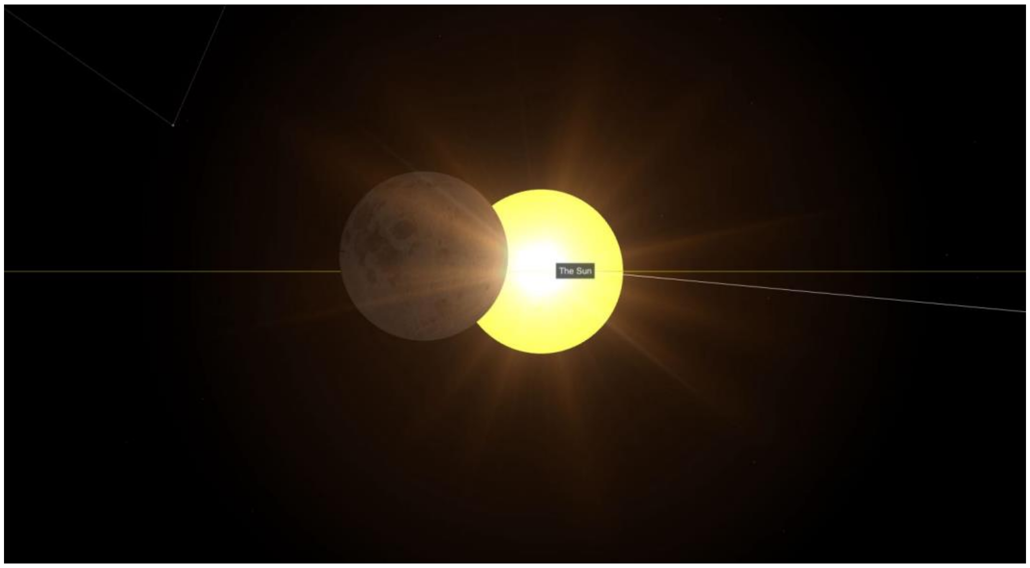
4. Create the orbit motion of the planets: This part will probably be the hardest one to accomplish and may require some additional research before we attempt something like this in our project. One idea could involve setting a central point (or "axis") for the planets to revolve around and then simply setting their speeds accordingly by the amount of time it takes for each planet to revolve in terms of one Earth year, which can be represented with a variable called earth_Year
to represent a much shorter time duration that we find reasonable for our program (after all, no one wants to watch our program for 248 years just to see Pluto go around the Sun once!). Below is a table of simple calculations that we may use for determining the revolution periods for each of the planets in terms of time within the program. The actual revolution periods are given in the table as well for reference. If we wanted to set earth_Year to, say 15 seconds, we can then use the values for the other planets in the right column to calculate their times. So, Mercury for example, would take 0.24*earth_Year seconds within our program to go around the Sun. Later, we may decide to make the earth_Year variable of varying type so as to not hard-code its value. This could be useful for giving users the option to change the speed of earth_Year, which would in turn change the speed of every moving object within the program accordingly.

Planet	**Actual Time of Revolution (in days)**	**In terms of program time (Earth = 1)**
Mercury	87.969	0.24
Venus	224.701	0.62
Earth	365.265	1
Mars	686.971	1.88

Jupiter	4,332.590	11.86
Saturn	10,759.220	29.49
Uranus	30,688.500	84.02
Neptune	60,182.000	164.77
Pluto	90,560.000	247.94

Some other extraneous features we may (or may not) decide to implement if time permits include:

1. **Planetary Rotation**
2. **Changing camera views:** for viewing each planet up close
3. **Making the Moon orbit around Earth:** this will definitely require an extra level of ingenuity on our part.
4. **Putting rings around Saturn:** This may or may not already be included with our spherical planet models.
5. **Creating a random solar system**



**A picture of the idea*

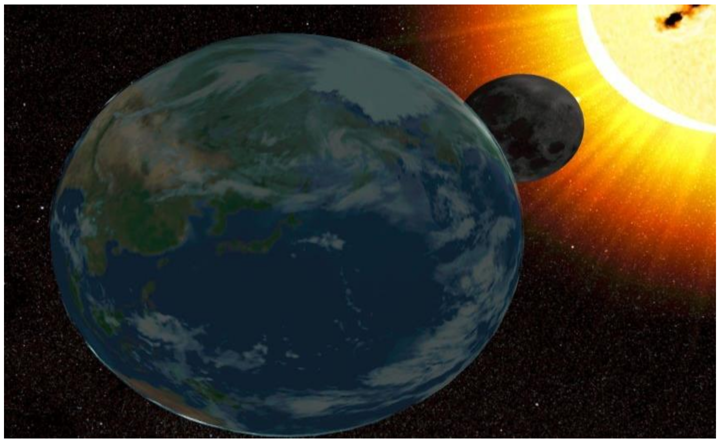
The Project Update

Currently our project is going quite well. We have a working solar system with all of the planets, moons, and even rotation. For our solar system, we are trying to model our program based off of the real world attributes (ex: distance, tilt, orbit, speed, etc.). Of course everything is not perfect, but it is looking very good as of now. We are currently using three.js to build some of our features.

One of more interesting things that we have working our program right now is getting the clouds on earth to spin independently of earth’s rotation. This creates quite a cool effect and makes the clouds seem to pop out from the earth a little bit. We also have moons that are rotating around their respective planets. This makes the solar system seem to be a little less empty.

One of the things that we are going to continue working on is making the sun look more vibrant/realistic. We currently have a decent ray effect coming out from the sun. However, we would like to be able to make this more animated instead of just a static rendering.

Finally, we would also like to be able to add some more camera features. This includes things like being able to lock onto specific planets. Currently our camera rotates and we can move freely around the solar system.



**A picture of our progress*

The Finished Project

We are quite happy with how our project has turned out. We were very fortunate that libraries such as Three.js exist. I think that we could have made a solar system without it, but it would have been a much simpler and boring solar system. We were able to get all of our main goals accomplished and all of the extras except for the random solar system generation. I believe that getting the random generation to work would require a minor haul in how the code of our program is structure, but it is entirely doable. However, we are completely satisfied with the progress of our program and we got basically everything we wanted done.

I think one of the biggest updates that we did during the time between the finished project and the update is modifying the solar system to be more “real”. The planets are spaced much further out, and the relative sizes of the planets are much more realistic. This of course makes it harder to see the planets from a zoomed-out distance focusing on the sun. To combat this issue, we had to add controls that would allow us to focus on the different planets. Now, getting the camera to focus on the planet was not necessarily hard, but there were definitely some issues that came along the way and aren’t still entirely fixed. The first issue is that the default zoom levels on the planet seem to be all over the place and seem to change based on the size of the canvas. This requires the user to have to zoom in/out until the planet is in the appropriate range. We also had an issue where if we had camera panning enabled, it would break our camera controls. To combat this, we turned off the camera panning for now. The final important issue that should be noted is that for the planet to actually get focused by the camera, the user must rotate the camera after they select a new planet to focus on.

We also decided to add 18 moons and 4 dwarf planets to our solar system. The moons were interesting because we had to set them up to orbit their respective planets. Three.js made this actually easy with some of their rotational tools that come in their library. We even got Pluto and Charon to orbit around each other like they do in real life. We were of course able to add all the appropriate texture and it turned out how we wanted it to.

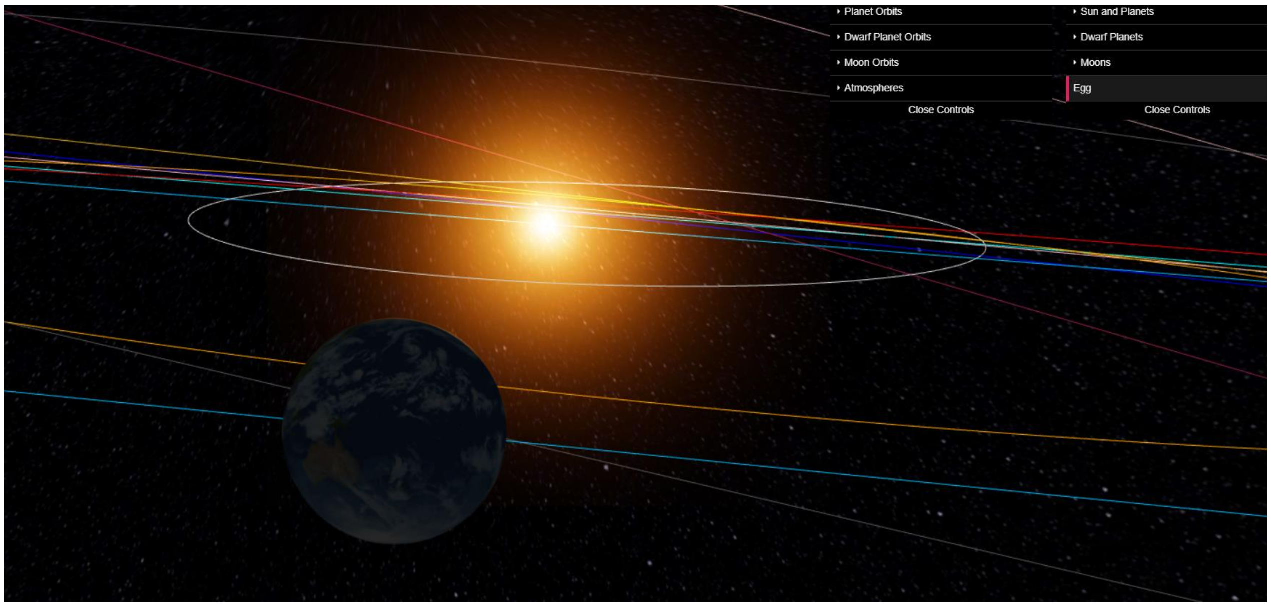
Orbits were also added to each of the planets and also their moons. These orbits were just different colored paths that showed the path that the planets took around the sun and also their respective angles. We created separate controls for showing planet orbits, moon orbits, and also dwarf planet orbits. This worked really well and gives an idea of where all the planets are located respectively to one another.

We decided to add some atmospheric effects to some of the planets and moons. These atmospheres could actually be disabled so that the user could view the surface of each planet as well. For instance, Earth had clouds that would rotate independently of Earth’s actual rotation. This gave the clouds a more authentic effect and made it seem as if the clouds were separate objects from just the Earth.

We decided to enhance the detail of the planets and make them seem more authentic by adding bump mapping and trying to add in some specular maps as well. Three.js made this easy because we could import separate bump maps and specular maps that we could then apply to our planet geometry. The bump mapping works well on the planets with a lot of craters and really makes it feel as if there is some depth to the planets. We also decided to make some specular maps on some planets and moons. We added a specular map for the moon, Titan, that mainly highlights where the lakes are supposed to be on the planet. We did this to try and make it obvious which parts of the planet are the lakes. It might not be the most accurate representation of how the lakes actually are built and highlighted, but we think that it looks pretty decent.

We also have rings around planets like Saturn, they also have an animation that clearly show that they are spinning. We have a separate texture for these rings and with Three.js we were able to attach them to the planet objects. This worked quite well, the rings being animated added a level of authenticity to the planets that was really nice.

One of the final little things that we decided to do was to make an “Easter Egg” planet. This “Easter Egg” planet was literally just an Easter egg that had both of our names written on it. We decided to put it way out of the way from the rest of our solar system. The texture of course is not very cool or anything like that, considering it was done in less than 5 minutes, but we thought it was kind of funny.



**A picture of our final project*