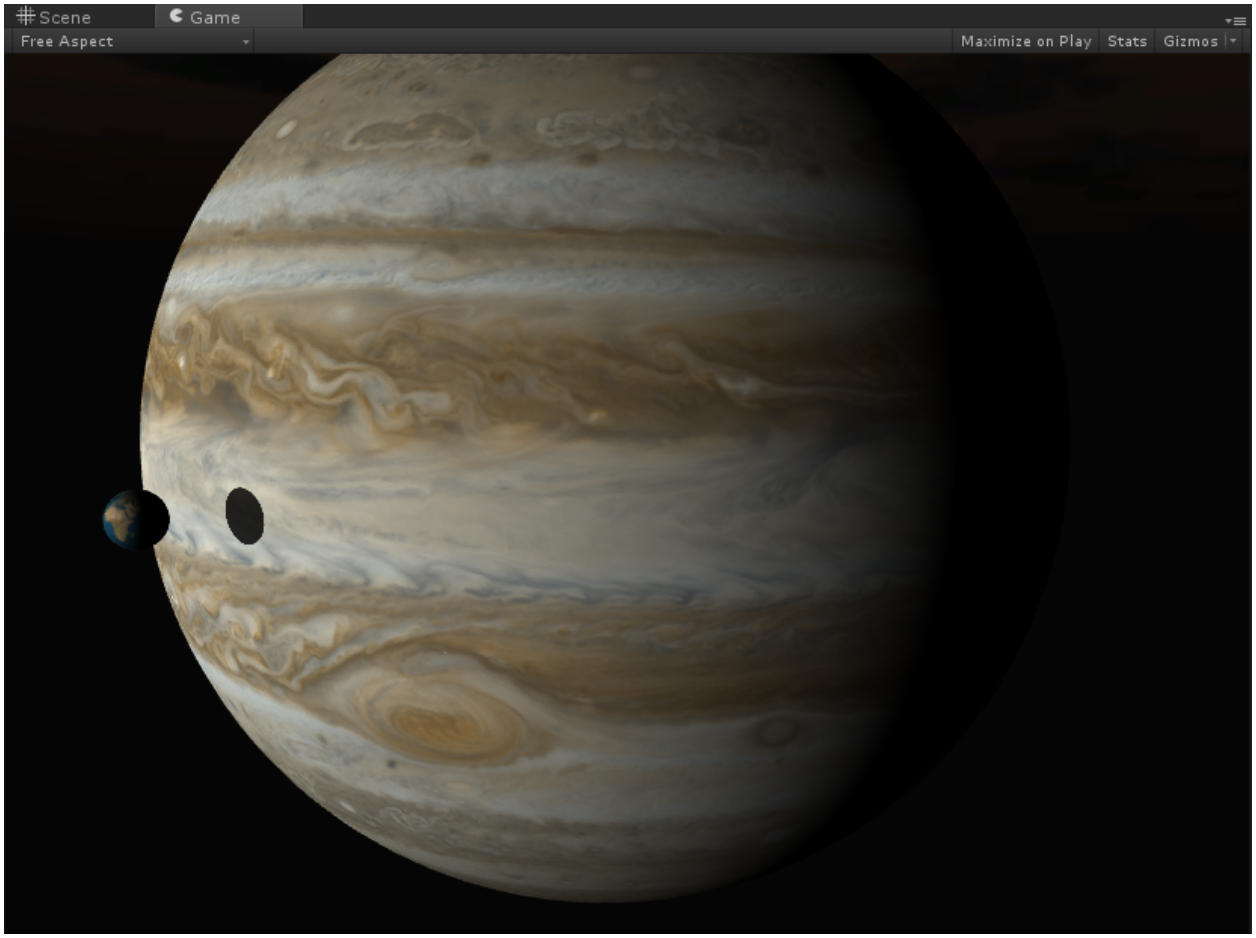
Developer Notes

9/21/2013: Better Rendering

- Better Jupiter texture
    - The Jupiter texture looked really bad in our first iteration of Jupiter. This really bugged me so I looked into increasing the quality. Immersion in our games is a key factor so it is really important to have good textures.
    - It turns out that the texture we initially imported into our project was a rather low resolution image with text on top. This was probably the biggest culprit in making the texture and our Jupiter asset looked bad.
    - Adjusting the way Unity renders textures helped a lot. I switched rendering with 1024x1024, mip mapped, trilinear filtering to 4096x4096, not mip mapped, point filtering mode. Now the texture is much crisper and our Jupiter is a lot closer to our final target.
- Enabled shadows
    - I looked at a Jupiter reference image and noticed something really interesting. One of the moons of Jupiter was casting a shadow on Jupiter. The interesting thing about this shadow is that it is a fairly standard soft shadow. There is nothing particularly difficult with getting this rendered in our game because this type of shadow rendering is fairly simple.
    - However, trying to get this to work with Unity proved more difficult than I first imagined, which is unfortunate. I could probably add my own shadow mapping technique fairly easily in a custom engine, so it's odd that Unity gave me such a hard time. This might be due to the distance values we use, though when I tried moving things together it didn't help much, so I'm still not entirely sure what is going on.
    - Unity has reserved soft shadows to Unity Pro, but I still can't get it to work even with my Pro version. Very unfortunate. I will have to look more into this later since I've spent too much time on it already.
- Tweaked the directional light
    - Not a big deal, just adjusted some brightness values and direction of the light to make the lighting on Jupiter seem more realistic. Things are looking good.
- Investigated Anti-Aliasing techniques available in Unity
    - There are two methods for AA in Unity: multisampling and post. Post is reserved to Unity Pro so I set multisampling on for now since Taylor doesn't have a Pro license. For the presentations and final turn in, we should probably turn on post instead because it allows AA on more than just edge aliasing (for example the current shadow produces aliasing that is not smoothed with multisampling).



3 hours

9/22/2013: Sound

- I read about Unity sound.
    - There are two modes: 2D and 3D.
    - We will be using 3D in order to get audio split between left and right ears (2D is used primarily for music).
- Added Voyager recording of Jupiter's electromagnetic radiation to our Jupiter asset.
    - There was a problem with Unity 3D sound and our scene: Unity calculates volume based on distance. Since we are very far away from our Jupiter asset, you can not hear anything with default sound settings. I fixed this by adjusting the volume curve on the sound component in Unity.
- We still need to write a script that will calculate volume based on head direction.
    - When the player is looking straight at an object, then play full volume. As the player turns their head away from the object, the volume decreases. We can probably do this by setting the volume inversely proportional to the angle between the view vector and head-object vector.

1 hour

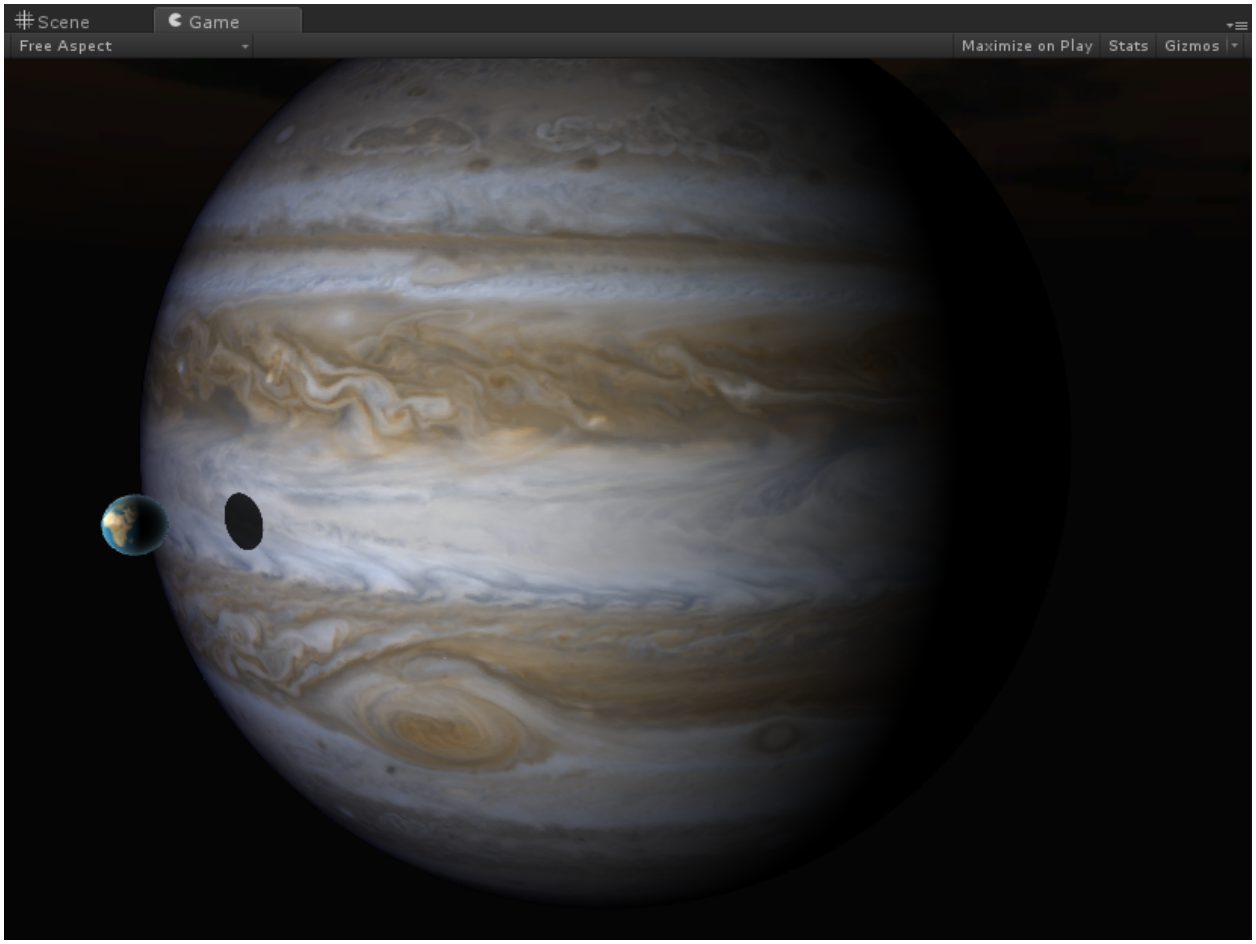
9/23/2013: Design Thoughts

- The movement of the player definitely needs to have a logic behind it. Otherwise, it is confusing and meaningless.
- While the surface level goal of the game is to give the player the opportunity to explore the solar system, it should also be about the exploration of the in-game law that governs the players movement. Through playing the game, the player will explore this law in parallel to exploring the solar system. The progression of the player will be based on these two parallel aspects of the game. This law (or mechanic) is currently the biggest challenge in making this game. The question that needs to be answered is: what is interesting in moving through look and how does that relate to moving through our solar system.
- If we are going to have a method of exploration through looking at interstellar objects, than the sky map has to be static. If there is a star on your left that transports you to Jupiter, that star should always be there and always take you to Jupiter. This is not entirely correct and there still needs to be further logic, but I think it's in the right direction.

2 hour

9/24/2013: Learning Shaders for Unity

- Applied Rim Light shader from Unity manual to approximate Earth's atmospheric glow.
    - This shader is not accurate and Earth will probably not be included in the final game, but I used this process to learn Surface Shading in Unity and to give our placeholder Earth a better look.
- I took the Rim Light shader and altered it to approximate Limb Darkening on Jupiter. I learned quite a bit about writing shaders for Unity.
    - Jupiter looks much better now. After this and getting better textures, we've got something that is close to final. It would still be nice to get a more accurate shader however.



4 hours

9/25/2013: Meeting with Richie at Awesome Inc.

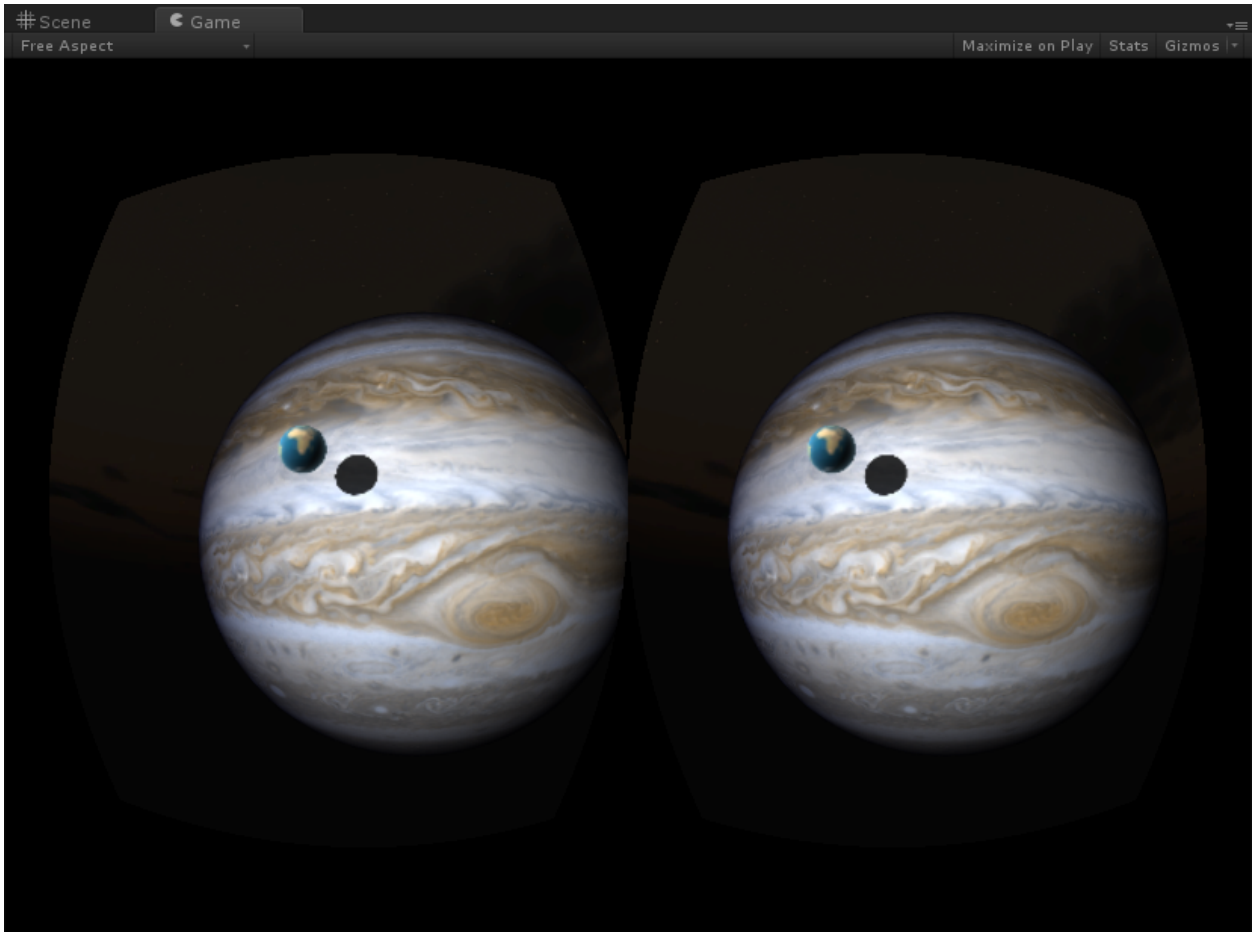
- We need to create a balance between realism, consistency, and usability.
    - For instance, the night sky is not static, but it may be better to create a non-moving sky in order to better communicate to the player their ability in the game.
    - Flashing lights and sounds can be used to draw the player to important items even though that is not realistic. However we must be careful to retain the immersion of the player and make the game feel real.
- Until now we have set up Git, Unity, and created base assets. While that is all useful, we need to put more importance on implementing the actual game mechanic.
    - In order to do this, it might be helpful to separate the project into two Unity scenes: one that is for assets and another that is used for game implementation.
    - Getting a game running will allow us to test it and make sure we have a design that is both good and finishable.

- It might be good to create a Unity scene that is just for prefab creation.
    - We can create prefabs in this scene, save them, then move them into the main game scene.
- Create parent-child hierarchy of game objects for easier management.
- Don't have separate scenes or loading of "levels" when moving the player between the different planets. Instead, have just one scene in Unity that contains all objects.
    - When moving the player, disable and enable the appropriate objects. If the user moves from Jupiter to Saturn, disable Jupiter and enable Saturn.
    - This is easier to implement.
- RunJumpDev has monthly meetings with "microtalks" at 7pm in basement of public library.
- The class presentation will be considered our first milestone
    - Jupiter and Sky Map in good shape
    - Rift support
    - Eye candy (screenshots, reference images)
- Make an inward sphere and map Nasa sky map.
    - This will serve as our sky map in the game. It is important to use a sphere and not a Unity skybox because the Rift does not play well with the skybox.

1 hour 30 minutes

9/25/2013: Rift Support

- Downloaded and integrated the Unity Rift package into our project.
    - This was a lot easier than I expected and makes me really glad that we went with Unity for this game.
    - It was as easy as importing the package and dragging and dropping a Rift camera (pre-built and included in the package) into the scene.
- There are still issues with the Rift integration.
    - The scale of the environment is not entirely correct. Part of this is due to the way the Rift performs stereoscopic rendering of the scene and the way our eyes work. Essentially, when a small sphere is 4 inches from your eyes, you will see it in 3D due to the separation between your eyes. However if a large sphere is farther away, you will not see it in 3D because the difference between the distance between your eyes and the distance between you and the sphere is much greater. We have to work to make Unity Rift camera render in this way: the distance between the you and the object is much greater than the distance between your in game eyes.



1 hour

9/27/2013 to 9/30/2013 - Presentation Preparations

- Worked with Taylor online and in person during the last several days to get ready for the presentation. We have decided to focus on the basic overview of our project and where we are in general.
    - Talk about Oculus Rift with Unity
    - Our game and why we decided to make it
    - How we are going to make the game and why
    - Testing to make sure everything is good
- I started getting sick on Sunday but luckily it's not so bad

3 hours

9/31/2013 to 10/05/2013 - Sick and Government Shutdowns

- Got sick and haven't been able to do anything. Taylor has been having issues with getting to Nasa website to get assets for the sky sphere

10/07/2013 - Presentation and Meeting with Richie and John at Awesome Inc.

- We gave our presentation today. It went better than I expected after being sick for a week.
    - I screwed up the transition between Detailed Design and Testing!
- Later in the day we met with Richie and John who were nice enough to reschedule our meeting to Monday from last Wednesday. We talked about where we were in the project and where we should go from here
    - We first discussed issues with getting assets from Nasa after the government shutdown. Taylor has been trying to get the skysphere done and committed, but has been having issues because he is unable to get the correct assets from Nasa website
    - The major discussion point after this centered around issues with scale using the Oculus Rift. Currently, using the rift and looking at our planets does not show correctly because the stereoscopic creates a sense that we are looking at two balls the size of basketballs or soccer balls. This is a major issue because it completely breaks our immersion in the game.
        - We might be able to fix this using one of two methods: rendering our planets to our sky sphere, either realtime or before, or figuring out how to

adjust camera rendering (maybe using multiple cameras for rendering) to get correct scale.
        - Putting a platform of sorts that is much smaller in front of the player might help in getting the scale to show correctly. For example you can have a podium or something the player is standing on.
    - After discussing how major immersion issues are, we have decided to focus on immersion correctness going forward rather than gameplay implementations. Because our game is almost entirely dependent on exploration and the space the player is in, having bad immersion breaks everything. Having fantastic movement gameplay is meaningless if the space you are exploring is bad.
    - We should watch the movie 'Gravity' for "research"

2 hours

10/10/2013 Writing Review Meeting Plan

- Worked with Taylor to write the meeting plan for tomorrow.

30 minutes

10/15/2013 Oculus Rift Trouble

- Something is stopping Unity from performing post processing for the Rift: there is no warping applied for lense correction.
    - I've looked everywhere in our projection and can't find out what is causing this issue. I'm going to have to either reset to factory settings or maybe try to open the sample Rift project to see if that has the same issues.

2 hours

10/16/2013 Looking for Rift Solution and Meeting with Richie

- The main discussion revolved around the Rift not working and what we can do to fix it.
- We also discussed what we can do going forward when something doesn't work, and having smaller tasks so we don't get stuck in similar way.
- The other major issue discussed was asset management. We have a lot of high resolution textures, so we have to figure out how to manage these (particularly when considering the limits GitHub introduces).

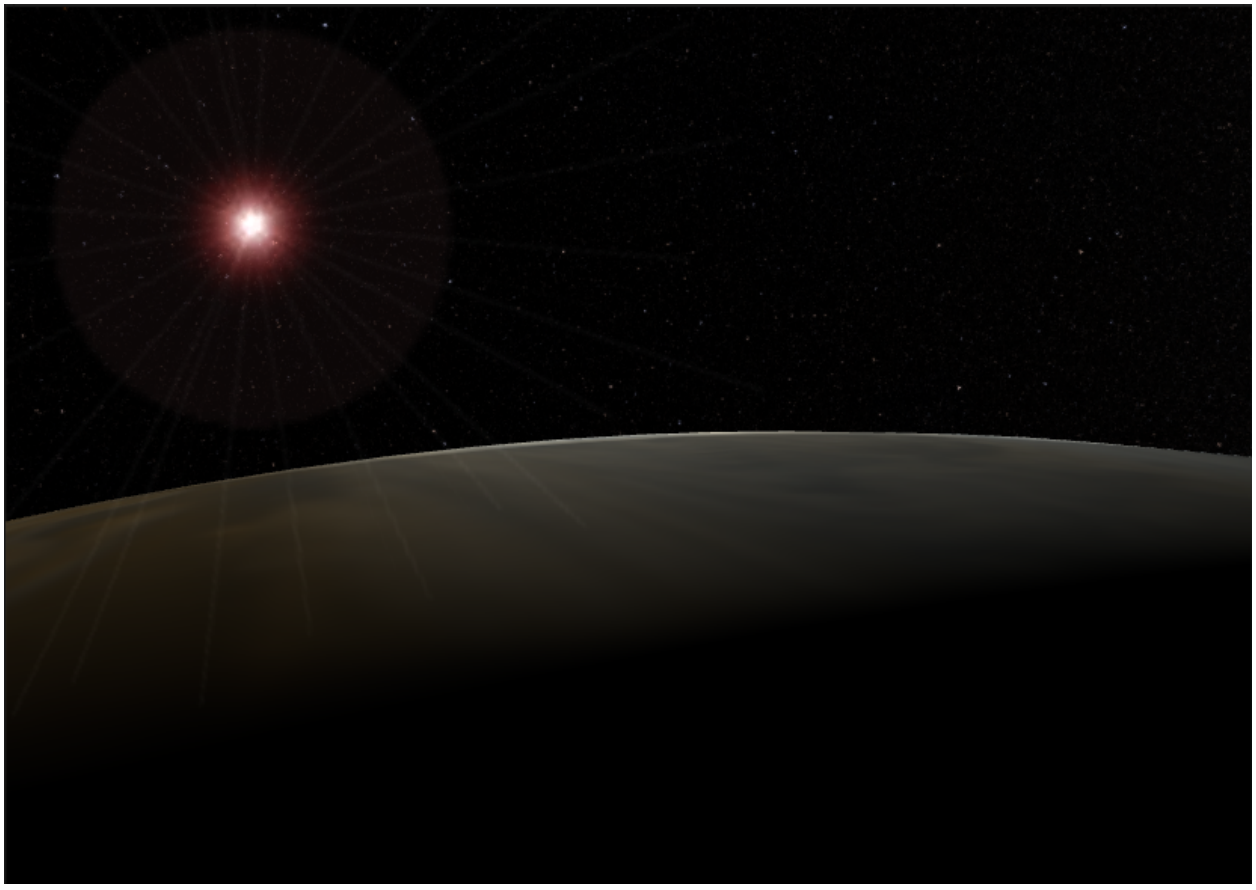
1 hour

10/20/2013 Research and Making Progress Again

- After a week of headache, Unity finally applying post processing again. I have no idea what I did or what was causing the issue. I really hope this doesn't happen again. I do know that the issue wasn't caused by the back and forth between Unity Pro and the free version, because the sample Rift Unity project that comes with the SDK was also not working. At least we know that and don't have to worry about resolving any Git issues.
- Now that the Rift is working, I tried our scene again to refresh my memory on the scale issue. It still is a problem. Something is not looking right. I decided to see the movie Gravity to see how they portray Earth correctly and give it the sense of scale even with 3D. This definitely helped and I came out of it having a better understanding for what we need to do going forward to fix our scale issue. I found the following while watching the movie:
    - The first scene opens with only the shot of the Earth. The 3D has an effect where taking note of the edges of the movie screen, you can see inward depth toward the Earth. However the Earth itself is entirely non-stereoscopic and has no depth at all.
    - The camera is much closer to Earth in a relative sense than our camera is to Jupiter. The Earth takes up almost the entirety of the screen while still maintaining to remain mostly out of the screen (so we see maybe 20% of the Earth, yet it is still taking around 80% of the screen). The massive amount of screen space the Earth takes up while still remaining mostly offscreen is a major contributor for giving the movie the correct sense of scale.
    - Following just the shot of Earth, the space shuttle starts to come into view. It is fully 3D and looks really small in comparison to Earth. It gives a lot of contrast, which helps with scale. While the Earth remains flat, the shuttle is fully stereoscopic.
    - The movie manages to create scale effectively by making it the priority within the first minutes of the movie. The opening is carefully paced to give this priority over anything else. It is important that the movie opened with just the shot of the Earth with nothing else, then panned camera to the right to view the shuttle as a speck, followed by the shuttle coming closer to the camera where the stereoscopic is then established. This series of events is crucial to creating scale.

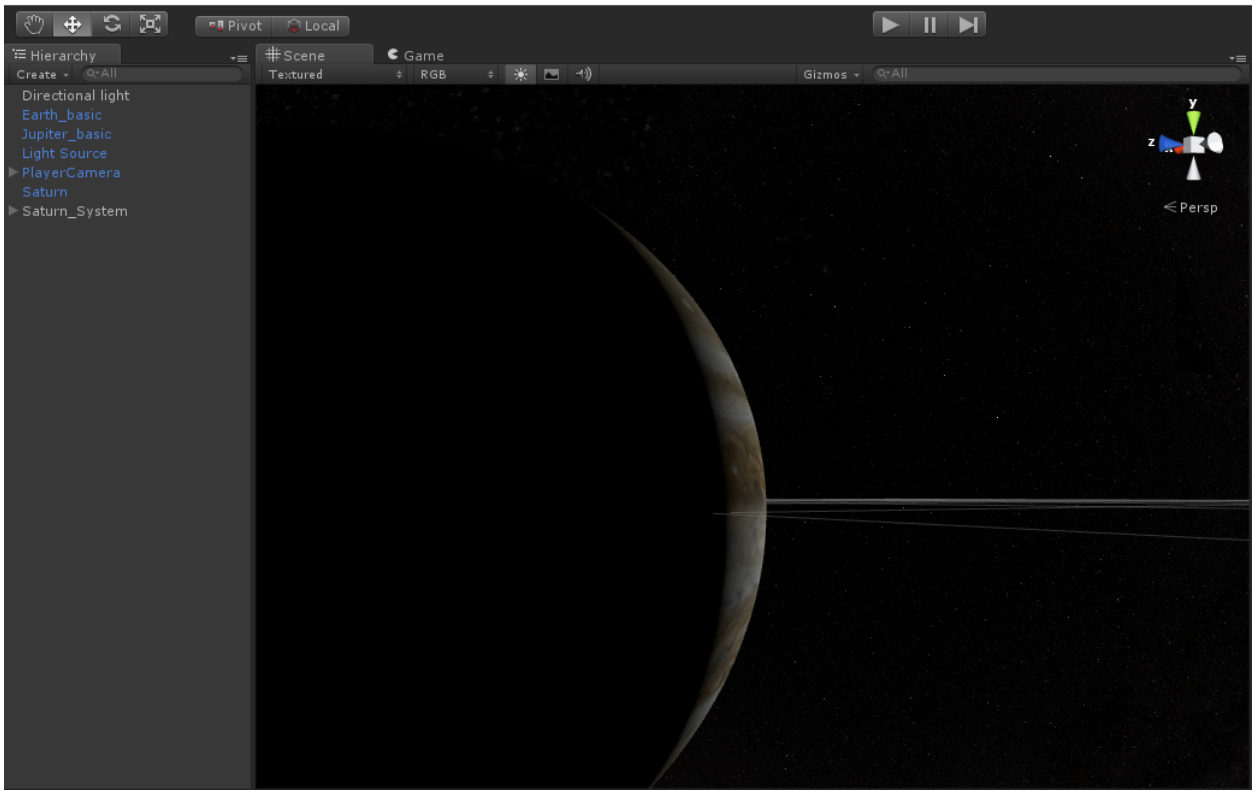
10/22/2013 Implementing Better Scale

- Moved camera much closer to Jupiter to fill the screen and create a better sense of scale.



10/28/2013 Space Toolkit and Work

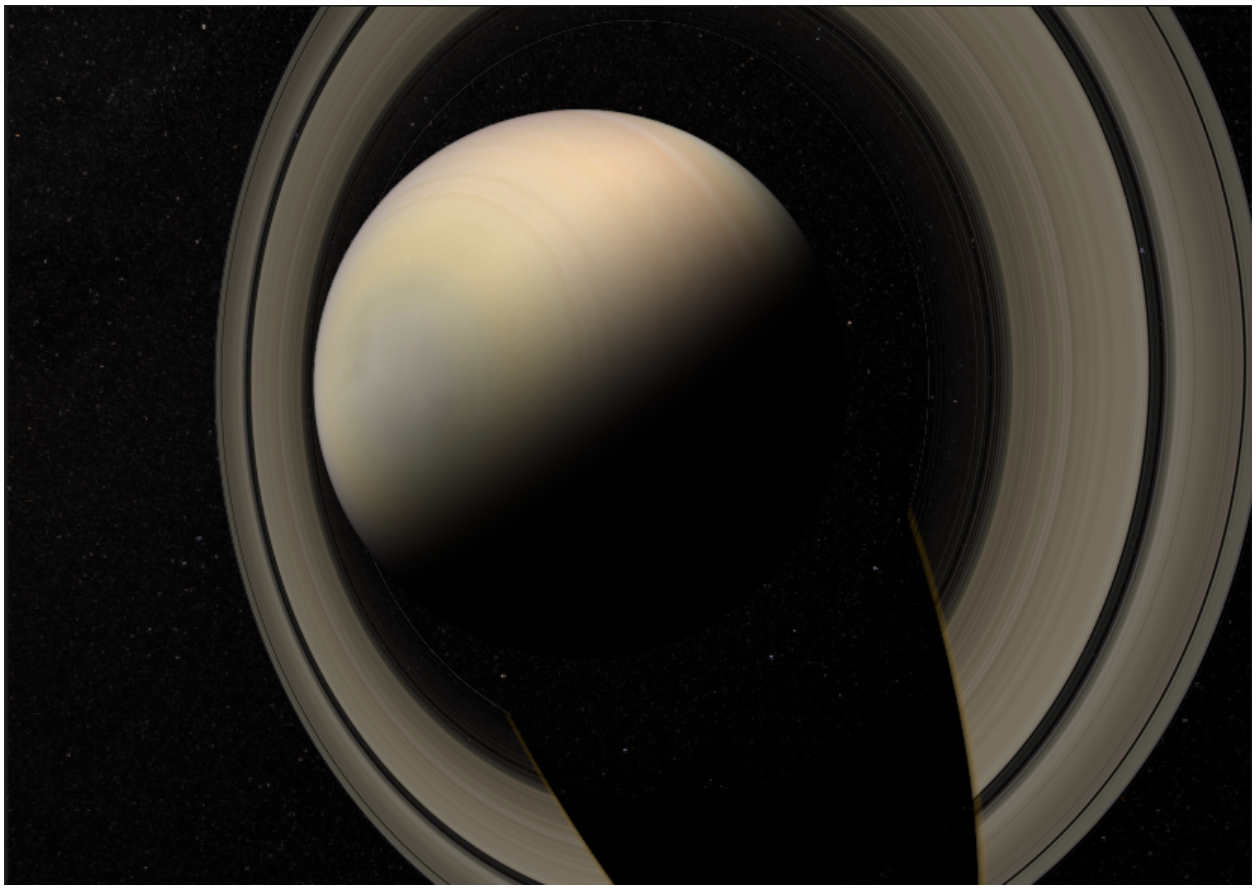
- I bought an asset from the Unity Asset Store names ‘Space Graphics Toolkit’ that will make it easier for us to quickly implement more planets. It provides shaders, scripts, and example planets. This will make it more likely that we will get a working game by the end rather than just a subset of components.
- Improved the sky sphere by including a new 8k texture with more appropriate exposure. I also disabled depth testing so the sky will fall behind every other object, and disabled lighting so the sphere is only sample for its texture and no lighting equations are applied. These are in the new shader titled ‘SkyShader.shader’
- Created a dual camera system. This is just a work in progress system that simply contains both cameras under a “Transform’ object, giving the cameras the same positioning.
- Put in test objects for testing scale as suggested by Richie and influenced by Gravity. These do a great job of creating scale. The player sees them in full stereoscopic, and in comparison to everything else, this gives great perspective of size.
- I committed the required files of ‘Space Graphics Toolkit’ and the above changes.



3 hours

10/30/2013 Work Done

- Updated the Oculus Rift Unity Integration to version 2.5
- Imported the full Space Graphics Toolkit after we were having some problems with missing files. This includes examples and documentation.
- Modified our test scene to include the example Saturn from the Space Graphics Toolkit. It looks great!
- Finally figured out how to set the near and far clip planes for the Rift camera. This will now allow us to make everything much bigger.
- Made everything in the scene bigger due to the new clip plane sizes, then committed all changes listed above.



2 hours

11/01/2013 - 11/08/2013 Work on Coding Assignment

- Worked on Coding Assignment

11/20/2013 - Meeting with Richie and John at Awesome

- In this meeting we discussed some problems we were having and Richie told us about the change we have to make to our game.
- Problems: when viewing the game through the Rift, objects would appear and disappear as the user turned their head. In particular, Saturn in our scene would pop in and out as you look at it and then away from it. Another problem we had was getting the Rift to be detected properly with our Macbooks. This problem made it impossible for us to demo our game during our presentation, which was a real let down. Fortunately we were able to get it to work for the demo before the presentation. Both Richie and John did not know how to fix either of these (since they had not run into them before), but gave us some good ideas.
- Change: we were told that our change is to have rotation in our game. The planets rotate and also having orbiting moons. During the meeting we talked about having the player rotate as well and the problems that would cause for our game. Because our game is about players looking around and activating scenes by staring at objects, a rotating player would not be a good idea. Not only will rotating the player (orbiting them around a planet or moon) make the player have difficulty in looking around with precision, but it would make them motion sick as well.

1 hour

12/01/2013 - 12/09/2013 - Change Implementation

- Taylor got the change request implemented into the game. The planets now rotate and have orbiting moons. It looks great. The game now feels alive and is no longer just static images. Even better, we get natural eclipes and sunrises that really add to the immersion of the game.
- We made the decision not to have player rotation final. Having this would go completely against the game design and our original plan to not make the player motion sick. When we met with Richie, he agreed that this was the right way to go.