Solar System in OpenGL

Project Report

Andrei Grosuliac

40133614

Concordia University, Department of Computer Science and Software Engineering

April 5, 2023

Introduction

The solar system has always been a big fascination for me. Considering that I maybe never be able to travel to space to it see it in all its magnificence, this project is the perfect occasion to create a realistic model that can try to mimic an immersive experience in space. However, a simple solar model is not something unique and has been done multiple times, so I wanted to take it a step further. One concept that has intrigued me was the common presence of binary solar systems in our universe. This is why I decided to transform our solar system into one with 2 suns instead of 1 to see how it would affect the movement of our planets and just how it would look in general. I had set 5 objectives from the beginning and this report will go over all of them explaining the implementation details as well as the inspirations from external sources. But first, here is a general overview of the libraries used:

Libraries

GLFW : Essential to create and manage windows and user inputs for *OpenGL* projects. Used in labs.

GLEW : Essential to be able to use *OpenGL* functions. Used instead of popular alternatives such as GLAD since it was shown in the labs.

GLM: Essential to be able to perform math operations with *OpenGL* (matrices and vectors). Used in labs.

ASSIMP: Popular library for *OpenGL* used for model loading. Used in lab 5.

SOIL2: Upgraded version of *STB* used for texture loading. *STB* used in lab 5.

Objectives

1. Create 3D models of the planets and stars

From the beginning, the choice to create custom models in blender versus finding them online was not decided and would come down to time constraints. Since I had no prior experience with Blender before, I judged that it would take too much time to learn how to create my own models and it would not add anything special to the application as decent models can be found online. So, every model in this project has been taken from free online resource. These include the 8 planets of the solar system and the 2 suns. Most of them are simple spherical models except Saturn, Uranus and Neptune which have rings around them.

To import these models, this is where the *ASSIMP* library came into play by using the Model class that imports the object and then separates it into Meshes with the Mesh class. This latter part is heavily inspired from capsule 5 which has already this process implemented. Then, *GLM* transformations are applied to scale, rotate and translate these models in their respective position in the screen. The vertex shader used in this part is also heavily inspired from the labs to generate a perspective projection to the screen.

2. Implement accurate movement of the planets

This objective has been to core of the application and what has pushed this solar system to be unique. For this part, a lot of research was done to see how our planets would behave we lived in a binary solar system. The conclusion that came up is that in order for our planet Earth to stay habitable, it would have to orbit the center of mass of both suns that orbit around each other. The other alternative would have been to orbit around only one of the suns while passing between both suns at some point in the orbit path which would make life on earth impossible as there would be sun rays coming from all directions. Now, the choice remained to either implement a circular or elliptical orbit since from research, there is no definitive answer to which type it would have to be. Therefore, it was opted to be an ellipse since it would be more challenging to implement and would be more unique. Finally, all of the planets including the suns are also rotating around themselves which is what happens in space.

This part has been implemented after researching the mathematical equations or orbit and ellipses and translating that into code. The position of each planet and sun over time is calculated as a shift in the angle from the center of mass over time. For the center mass, it was calculated given the ratio of the scale between both suns, one being 3x bigger than the other. The GLM library is crucial for this part as it performs the necessary transformations to show the movements of planets over time. A simple line shader is also created to generate the white orbit lines that the planets have to be able to see their orbit path through space more clearly. This part of the project has been the most challenging since all equations had to be implemented from scratch through research and dealing with ellipses is more complicated than circles.

3. Implement a realistic lighting model for a binary star system

Since the distance between planets in space is very large which means that there is no global illumination between planets, a local illumination model is implemented. It consists of 2 moving points lights being the 2 orbiting suns. It used all 3 ambient, diffuse and specular components to calculate the resulting light for each surface. One of the suns shoots red light while the other white. Although this is not realistic, it has been implemented to display a more interesting combination of resulting light through time on the planets as they orbit around 2 orbiting suns. Coming back to the scale of the distance between planets being very large, this justifies that no shadows have been implemented since planets are too far away from each other to cast shadows. The only reason the model makes that distance seem very small is to have a more enjoyable experience for the user so he can see all the planets in one glance instead of having to move the camera all over to observe every planet.

For the implementation part, since rendering illumination has been explored intensively upon doing the ray tracer assignment, it was not hard to transfer and apply the local illumination knowledge over to *OpenGL*. The fragment shader's code for lights is heavily inspired from the labs since the lighting model works in the same way in this project. However, small modifications needed to be done to account for multiple light sources in a fragment shader file. To account for the lights moving, the only modification needed is to change the light's center position to their respective sun's position in the same loop that the planets move over time.

4. Use textures and astronomical data to enhance visual fidelity

The plan for this part was to insert different kind of textures varying between 2D for smoother surface and 3D for rockier planets. Considering the same reasoning for the models, the decision fell to finding textures online instead of finding a way to create them myself. However, this time, finding free 3D textures was not feasible. Unfortunately, this resulted in only 2D texture being used in this project.

To import these textures, the *SOIL2* library is used. It is essentially the *STB* library used in the labs with some extra features which were not really needed for this project. The reason for using is a personal issue where I had some compilation problems with the *STB* library and the *SOIL2* worked right away so I just stayed with it. Since the *SOIL2* library has similar functions to *STB*, the code to import and apply textures is very similar to the one in the labs.

5. Create an interactive, orbital camera system

A camera system that allows the user to move through space to observe the planets is essential for the immersiveness of the application. A lot of different actions have been added in surplus to basic camera movement options to give the user a lot of options to explore the space and control. Different variables can be modified too by the user such as the speed of rotation of planets and camera movement, the presence of orbital lines and the possibility to view the solar system from different perspectives such as from the earth or from above.

The basic camera system is heavily inspired from the labs. Using the scroll wheel to zoom has been removed since it is not fluid and has been replaced with the “w” and “s” keys. One feature that I have not seen in the labs is to move the camera directly up or down the Y axis so this as been implemented as well with the arrow keys.

Difficulties and Improvements

Unfortunately, the reality of this project is that almost 50% of the time spent was to technicalities. Setting up the project and linking the external libraries was a process new to me and it ate up a lot of time and generated many bugs. Due to personal restriction, the project needed to be setup and worked upon in Windows and Linux so I had to learn how to setup it up in 2 different ways on two different systems which both introduced their set of bugs. Furthermore, I have different graphic cards on both systems so some *OpenGL* functions did not work properly when transferring from one OS to another. Furthermore, whenever I had bugs in the shader files, it was very difficult to debug since you cannot output out anything from the *GLSL* files except a color onto a screen pixel.

On a final note, there are many things that I can think of implementing that could upgrade this project if I could redo it and not waste as much time on technicalities. First, adding moons to some planets introduces the possibility of adding shadows and global illumination to the lighting aspect. Having 3D texture would make the experience even more realistic since not many planets have a smooth surface, but instead a more of a rocky one.