Dylan DE JESUS
Kevin QUACH

Université Gustave Eiffel 2023-2024
Master 2 Science de l'Image

Report VisuSysSol

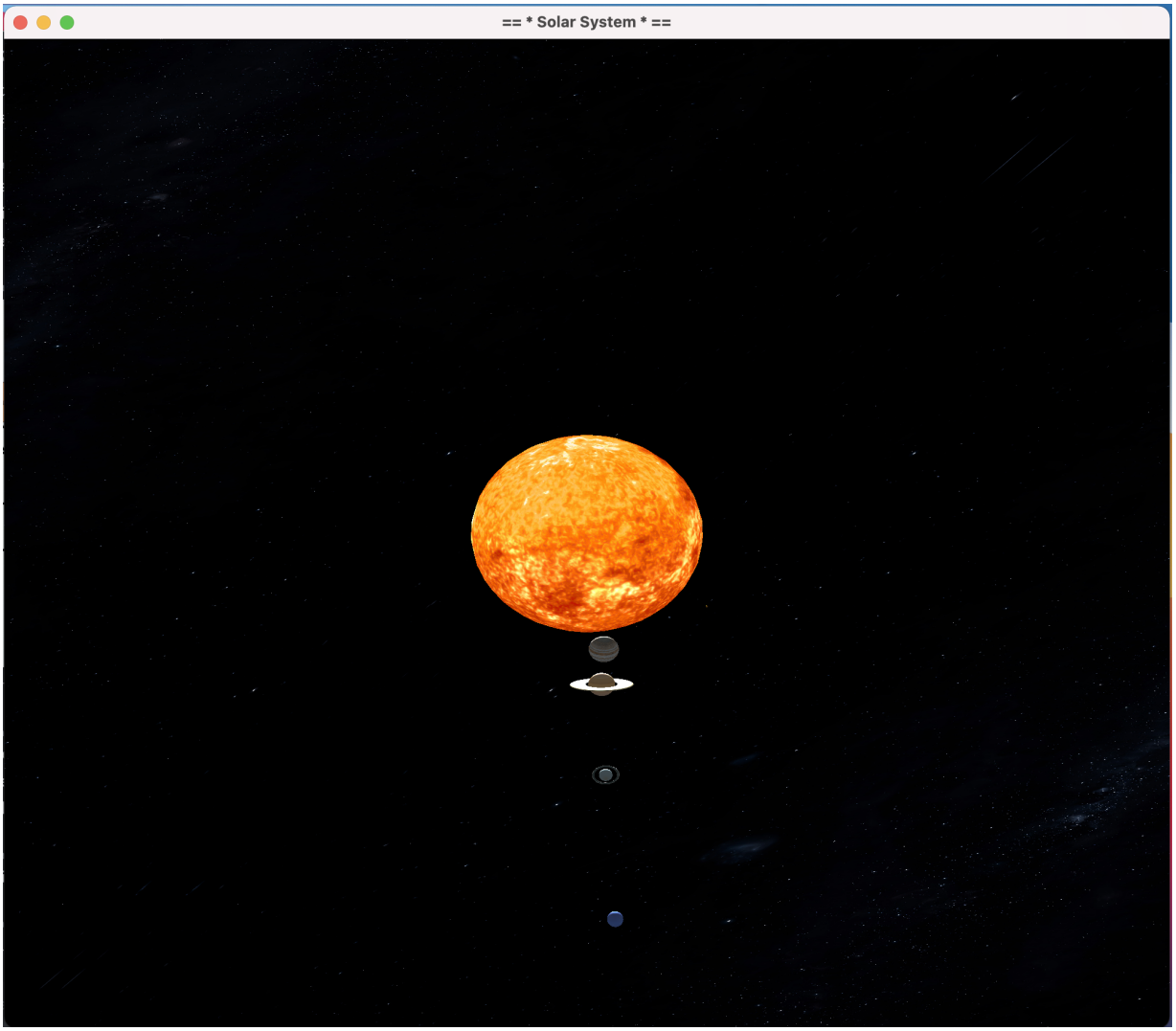


Table of content

Introduction

VisuSysSol is a simulation of the solar system made in C++ using OpenGL. Allowing the user to observe the planets of the solar system and their satellites, it offers three POV modes: a bird-eye view from above the sun, a profile view centered around the sun and a planetary view for each planet. You can rotate, zoom and unzoom the camera around the subject in each POV.

User Manual

Building the project:

Create a build folder in the project's main folder:

- **mkdir build**

Move inside the created folder and use the following commands:

- **cd build**
- **cmake ../**
- **make**

The executable has now been created in the bin folder, to run it, use:

- **./../bin/SolarSys_**

Interacting with the app:

Once the app has successfully started and loaded all the models, you can start interacting with it.

The camera is set on the bird's-eye view of the sun by default. Here are the different commands you can do to interact with the app:

- **UP ARROW : Switch to planetary POV if you're not already in it and change the focused subject to the next planet.**
- **DOWN ARROW : Same as UP ARROW but switch to the previous planet instead.**
- **RIGHT ARROW : Speed up time by a fixed amount, cannot go over a set cap**
- **LEFT ARROW: Slow down time by a fixed amount, cannot go under a set cap.**
- **T : Makes a time leap**
- **SPACE : Switch to the bird's-eye view POV of the sun**

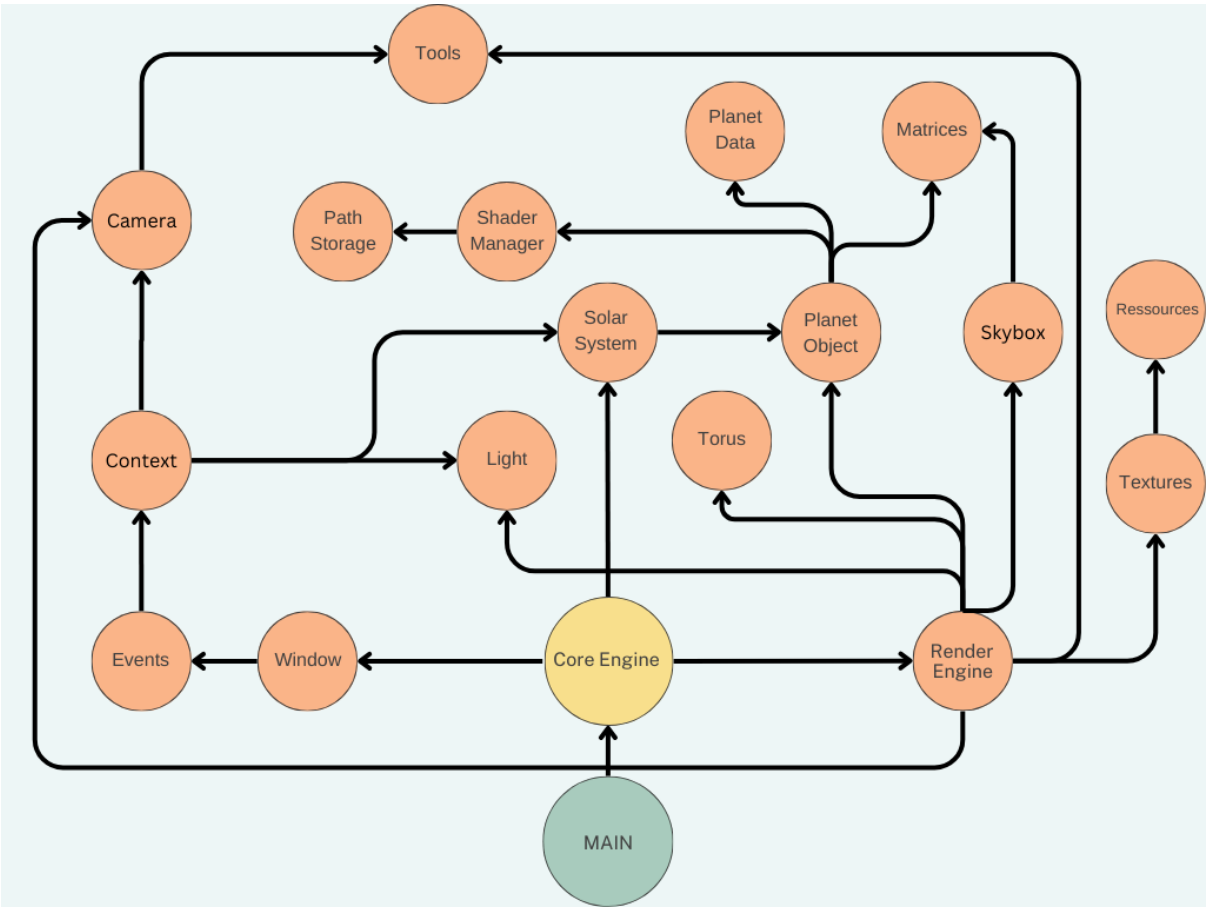
- **ESCAPE : Quit the app properly**

While in the bird's-eye view of the sun, you can press:

- **D : Reduces the distance between planets to better compare their orbital period**
- **P : Switch to a profile view of the sun, you should use it with D to better view the planets**
- **MOUSE MIDDLE BUTTON : Zoom and unzoom the scene**
- **MOUSE RIGHT BUTTON : Makes the scene rotate around the center element of the view**

Modularization

We wanted to implement a well structured app and that's why we took a lot of time at the beginning of this project to think about a clear and complete modularization of the project's elements we had in mind.



The root of our visualization app is the input module, it calls the main loop that manages the program in the core engine part. The core engine module manages stuff like light sources, camera objects, the time management, and the objects to render. It iterates on all the elements of our “solar system object” that stores the planets (with its information like the textures, transformation matrices…) and update the matrices through the time spent to male it rotate/be animated, then it sends it to the render engine part to be drawn in the scene.

The render Engine part takes some primitives like spheres (planets), torus(rings), cube(skybox) and does all the GL stuff to make sure that the element will be rendered properly and send information (such as matrices) to the corresponding shader.

Choices

We’ve decided on letting the user be able to rotate the camera around the subject of each POV instead of restricting movements as we think it’s always better to allow for more freedom of movement especially for a visualization app like this.

If we kept the exact proportion of distances between planets from real life, we wouldn’t be able to see anything as planets are ridiculously far apart from each other. In order to see the planets relative to the neighboring planets (and the sun if it’s not too far), we’ve divided the real distance by a distance unit to reduce it to an acceptable value for visualization. This means that the distance between planets isn’t accurate to real life but the proportion is.
The same has been done with the planet's size to reduce it to an acceptable value for the application as having a sphere of size 12742 and more would be ridiculous. In spite of this solution we found to display the real proportions we couldn't see the whole solar system inside the window in the main view, so we decided to implement an extra visualization concept (in fact it isn’t a new point of view) that keep the good proportions of distance within a far smaller range (that could entirely be seen by the user).

During the development of the app, we ended up significantly increasing the value of the distance unit. This was done because some of the planets were way too far to be displayed in the window (i.e Jupiter from Mars POV). By increasing the distance unit, we reduced the distance between planets and some far away planets managed to be displayed though not all of them. However this had another side effect, as the distances reduced, some satellites ended up inside of their planets so we ended up having to find a good proportion between distance unit and planet size unit to display as much as we could. Some of the satellites are, unfortunately, still inside their respective planet though. Moreover because of the decreased distance, some satellites with little to no orbital inclination ended up on the ring of their planets.

We've added a "pseudo ambient light" by setting a minimum light factor to the app. In other words, when part of an object wasn't illuminated by a light source (i.e the side of the planets opposite to the sun which is the light source) , we would set a minimum lighting factor so that we can slightly see the backside of the planets. Without it, the backside of the planets would appear completely black which isn't great for visualization, the primary goal of this app.

Concerning the textures, it seemed that we took costful ones, so we avoided the satellite's display in the bird's-eye view of the sun. We also made some other compromises to make sure the app could run properly. If any problem/slow-downs still occurs we advise you to replace it by others less detailed.

Finally, for the skybox, we first wanted to build a bounding box around the whole solar system but it seemed to be too expensive in resource cost so we decided to display it as a 1 by 1 cube. We first put it around the cam and disabled the Z-buffer to make the texture of the space fit the background but we found that even when the cam was moving and not the skybox there wasn't any bug in the display, so we decided to keep it as it was (less matrices update is also good for the performance).

Project Management

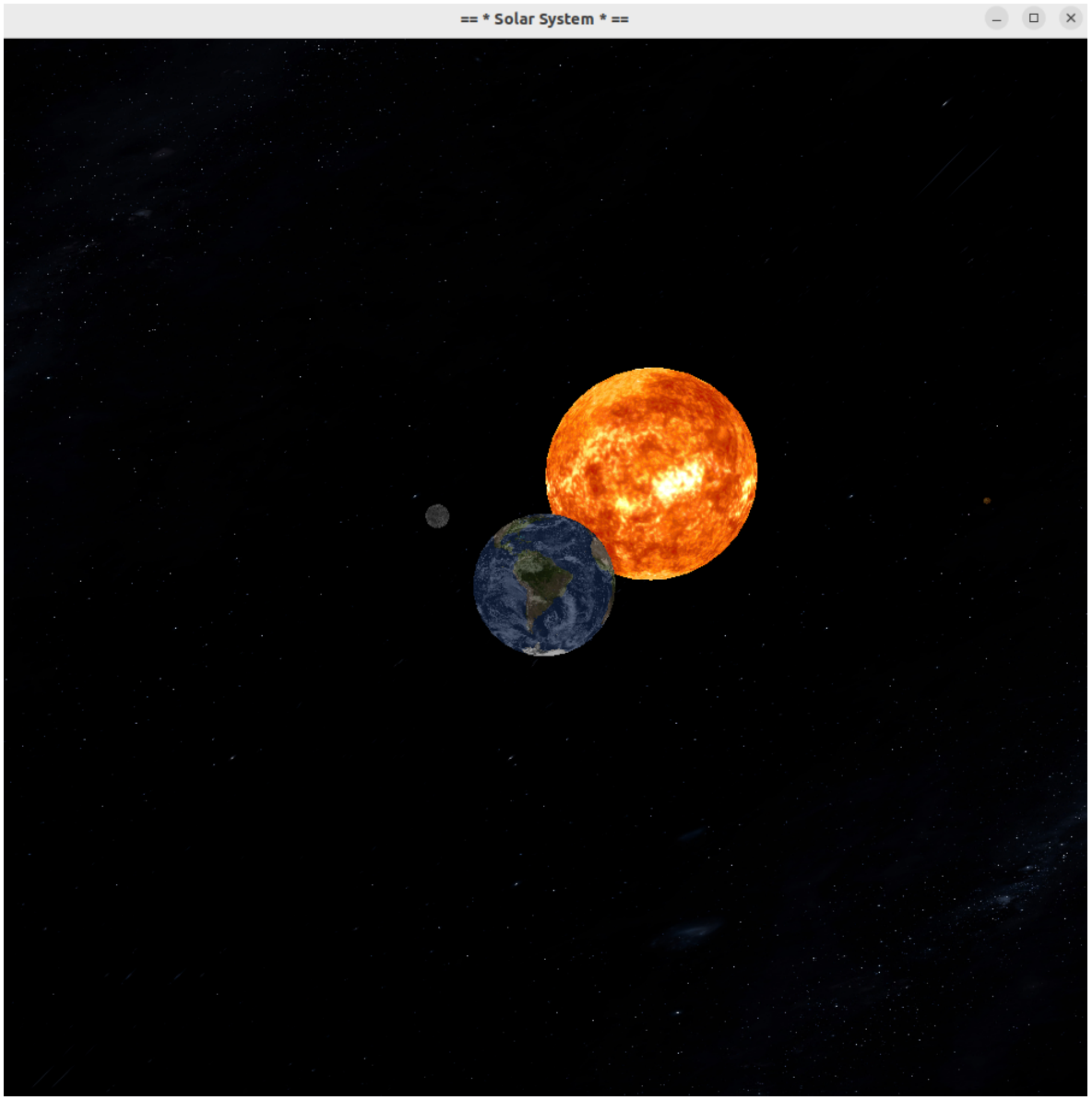
Dylan	Kevin
● Main structure of the code ● Window management ● Some of the event handling ● Profile POV ● Default POV ● Light management ● Rendering system ● Skybox ● Multi texturing ● Shader Management ● Structure of the satellites system ● System of the Planet creation	● Camera ● Adding planets ● Some of the event handling ● Planetary POV ● Planet size and distance conversion ● Planet rotation time ● Planet revolution time ● Time handling ● Torus and torus texturing ● Pseudo ambient light ● Axial tilt ● Orbit inclination

Results

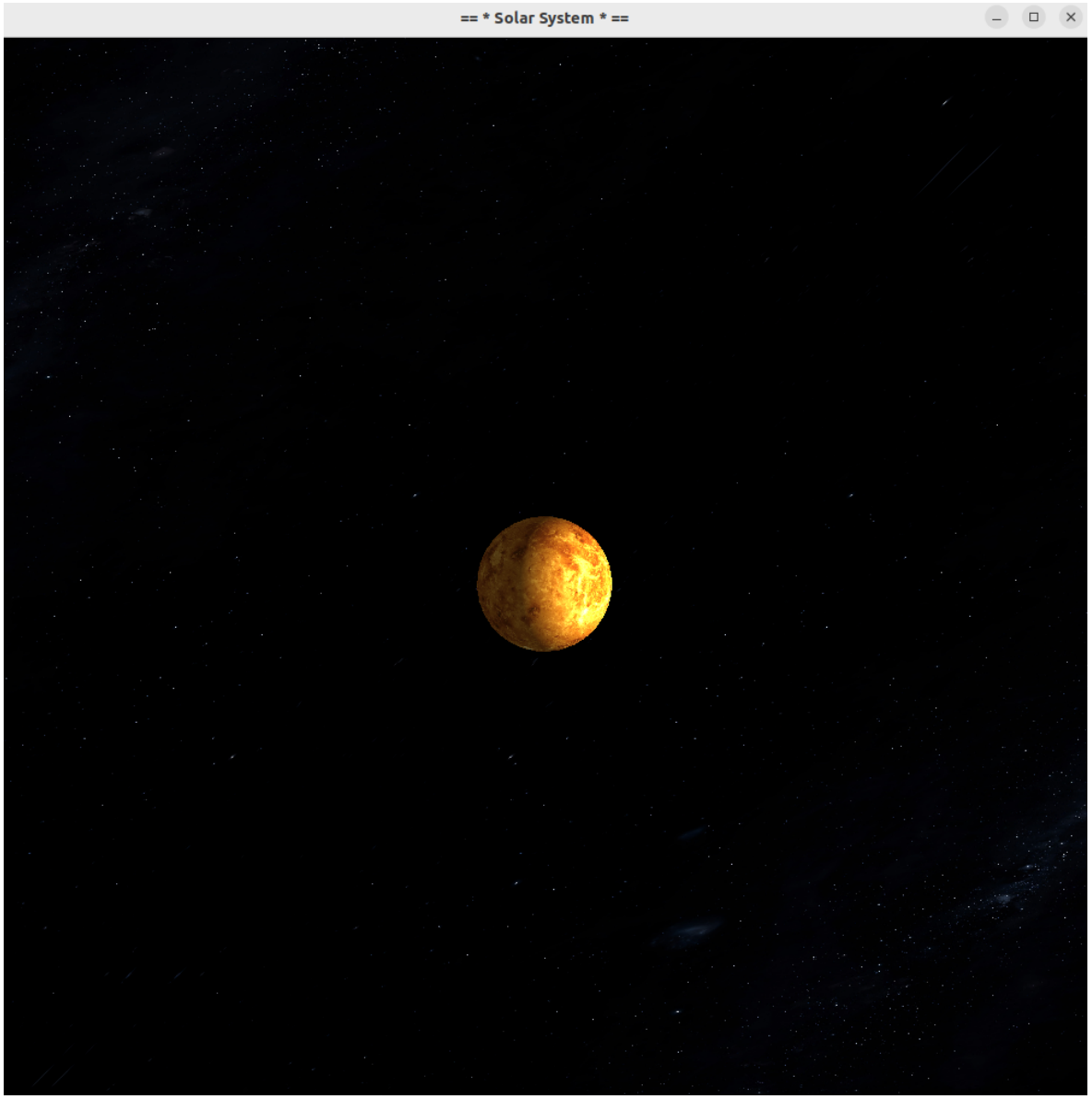
The end result of our project was quite satisfactory in our opinion. The camera controls are smooth enough, the POVs were implemented well and there weren't any

major issues with most of our other implementations. We would have liked to be able to do more, such as adding a thin ring on the planet's trajectory to display its orbit but we couldn't add it due to lack of time. A problem we didn't solve was the placement of some satellites that ended up inside their planets, such as some of Saturn's satellites as explained in the "Choices" part. We had plans to add comets but again, we couldn't because of a lack of time. All in all, it was a pretty fun project to work on and helped us apply what we've learned as well as learn a few new things such as managing the skybox or modelizing toruses. Although sometimes a bit frustrating as we ended up stuck on a few things such as managing the satellite or the modelization and texturing of a torus.

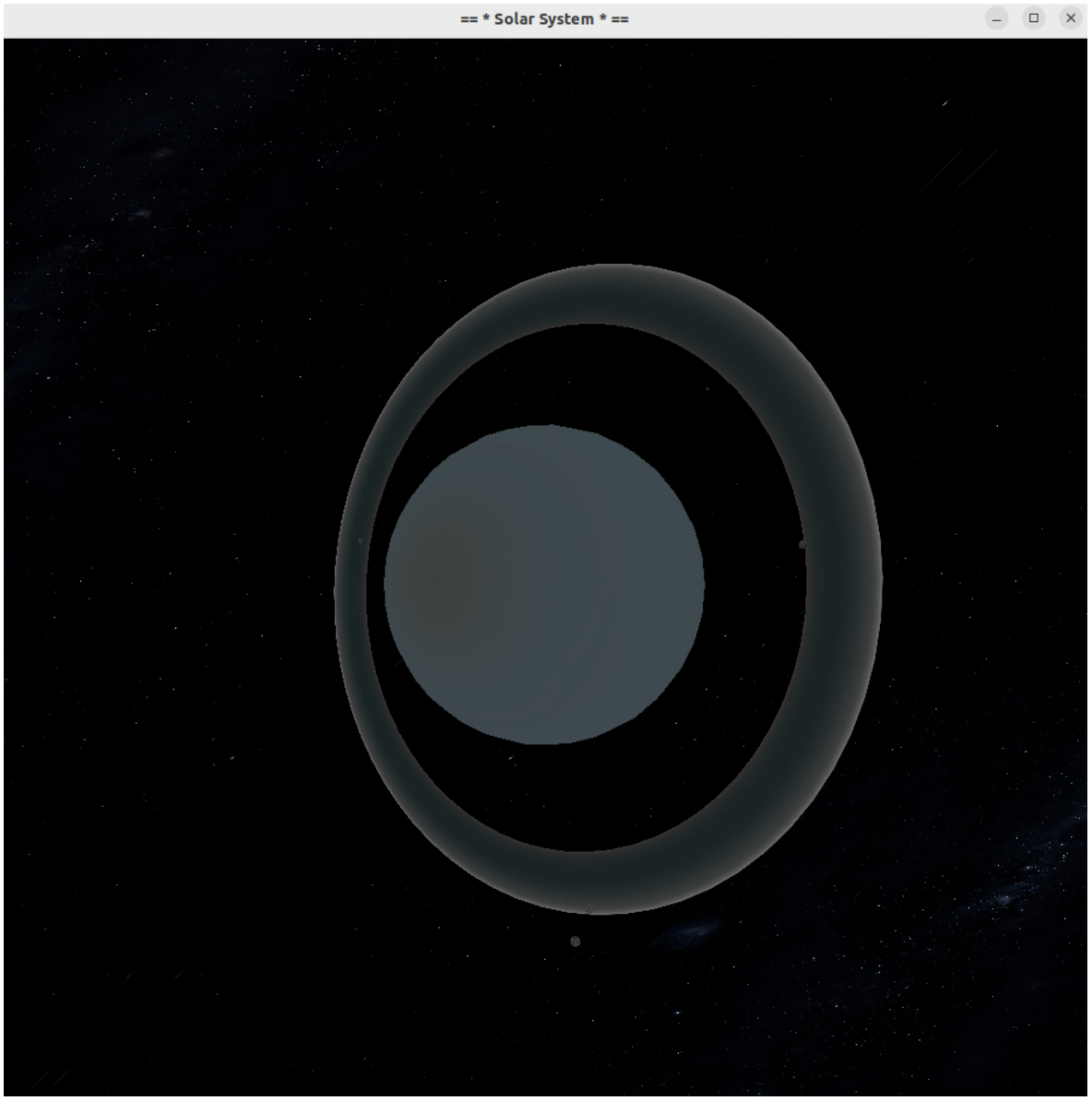
You can find the project's source code at the following link.



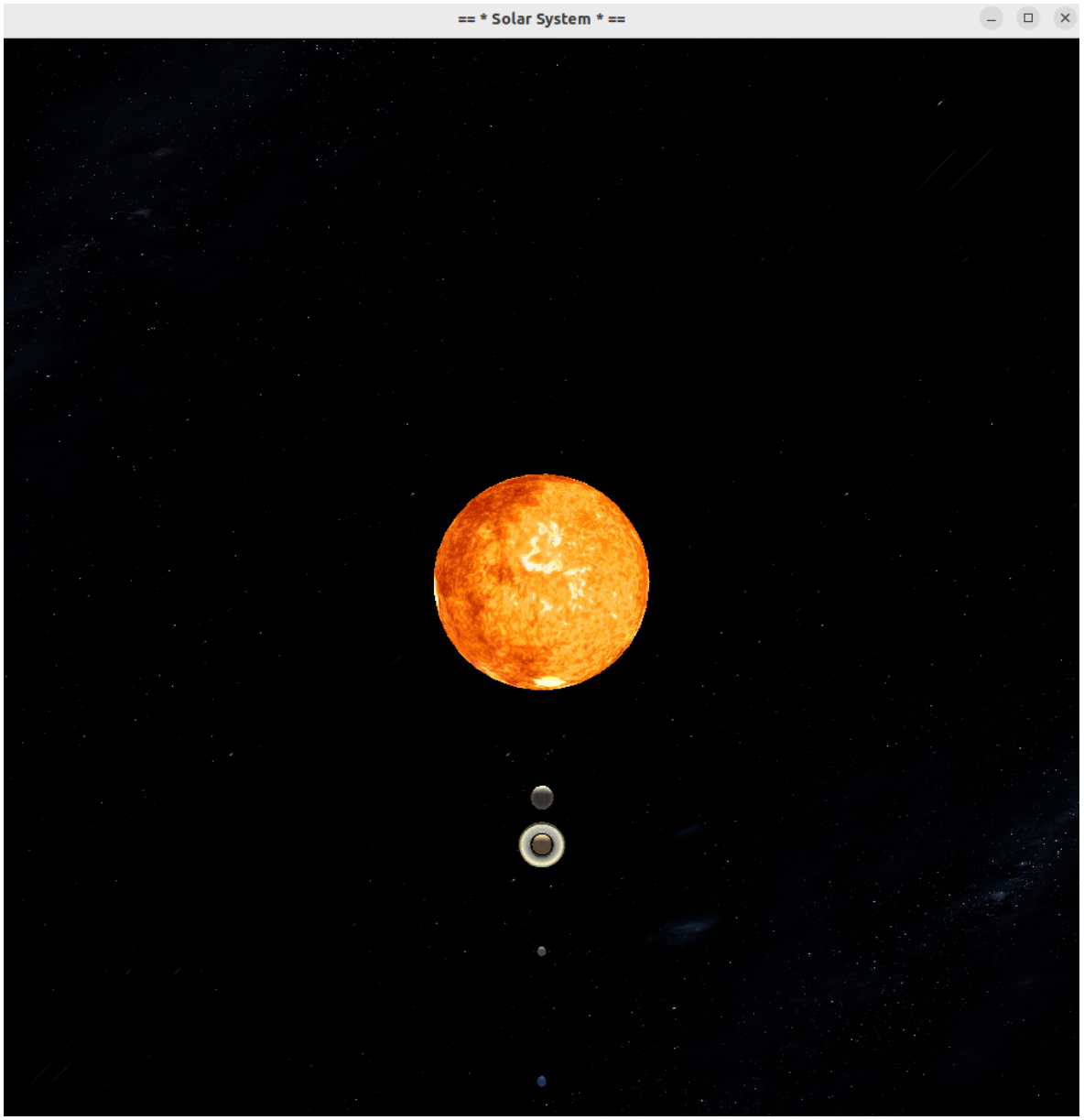
Planetary POV of the Earth with the Moon orbiting it



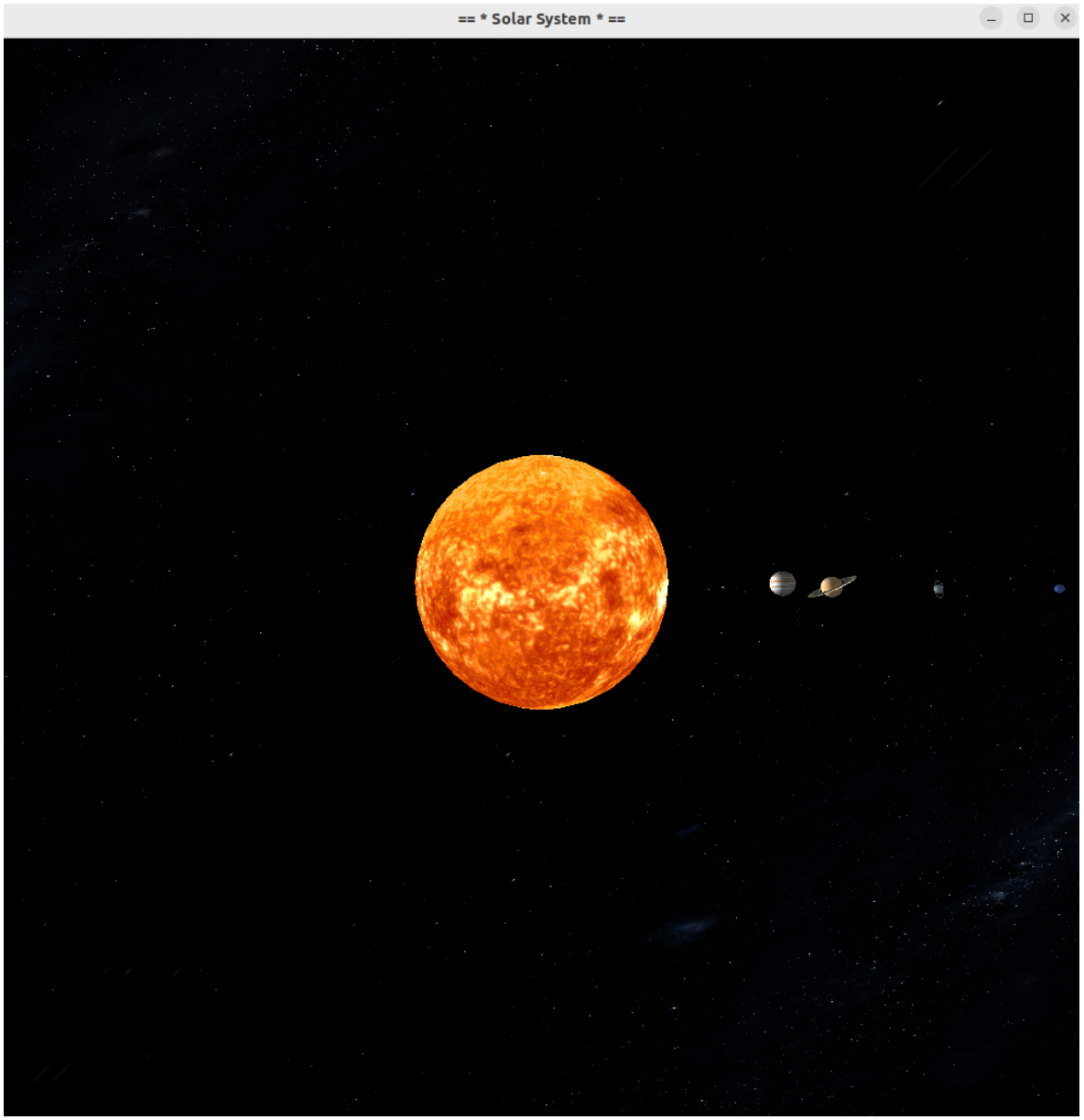
Planetary POV of Venus, whose side facing the Sun is lighted up by the later



Planetary POV of Uranus with its tilted axis and its moons orbiting it.



POV of the solar system’s ecliptic with reduced distance mode



Profile POV of the ecliptic with reduced distance mode