# Computer Graphics Course: Solar System Project Report

Federico Favia, *EIT Digital Master School, VCC, University of Trento*, 2019

Lecturers: Ha Nguyen Hoang, Nicola Conci

## I. INTRODUCTION

The purpose of the project was to build a realistic model of the Solar System by making use of the computer graphics library called OpenGL[6]. In order to achieve this goal, an object-oriented structure in C++ language has been built.

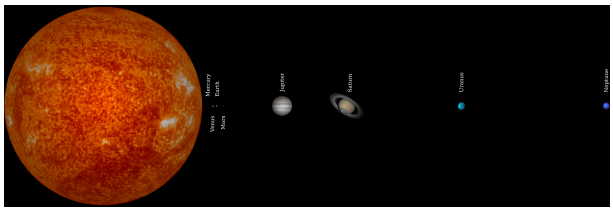

Fig. 1: Scaled-size representation of the Solar System planets.

VisualStudio is the source-code editor used for the project and the most important functions for lighting and rendering scene are part of GLEW and GLUT libraries. The public domain library stb_image.h - v2.22 is used to load textures.
The Solar System[3] is composed by the central star, the Sun (placed in position (0, 0)) and eight planets (Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus and Neptune) which circulate around it through an elliptical path. For practical reasons, in this project the orbits are simplified to circles.
Textures and lightining are used to render the scene in a more realistic way. All the calculations to obtain a proportional and realistic view are done based on the Planetary Fact Sheet by NASA[2]. The sizes of the planets are proportional compared to the Sun and the orbital distances are too. This turns out in two effects:

- The nearest planets are really small compared to the Sun, therefore the user needs to zoom in very much to see them.
- The model is really huge. To see the farest planet, Neptune, the user has to move very far with the special keyboard feature.

However, this is approach has been used to be more realistic as possible, altough the distances are not proportional compared to the size of the planets. Otherwise the model will end up in a complex and bad user-experience.

## II. IMPLEMENTATION

The solution is placed in the folder OpenGLRoot\solar_system_favia. About the implementation, besides the OpenGL initialization, display mode and lightening properties, the main feature of the program is the initialization of the planets through the class function:

*planet(const std::string& _texture_path, float _radius = 1.f, float _speed = 0.f, float _orbit_distance = 0.f, float _revolution_angle = 0.f, float _axial_tilt = 0.f, float _rotation_motion_angle = 0.f)*

This way all the variables of each planet are set, such as the path of the texture, the radius, orbit distance from the Sun, speed, axial tilt and rotation motion speed. Then the planets are drawn, as well as their orbit lines and their rings if present, and the scene is rendered. The other important function is the one for checking the user input on the keyboard:

*void pressKey(unsigned char key, int x, int y)*

The user can press + or - to zoom in or out, WSAD to move the view, IKJL to rotate it and R to reset it.

## III. ISSUES AND TEXTURES HANDLING

The issues arising from the project were first of all of geometric nature. As explained before, the model result is really huge. Then a trial and error approach and some approximations on the distances have been taken into account. Approximations on the rings of Saturn and Uranus have been made too. The field of view has been calibrated

from 5 to 750 in order to have the possibility to see the furthest planets. The other problem was the handling of textures, a topic not in-depth analyzed during the course. To solve it literature and methods on the web[5] have been used with the help of a public domain library (stb_image.h). This library can load the texture image path in the CPU and will give a pointer to a buffer of RGBA pixel. Then this pixel array will be uploaded to the GPU through OpenGL. The textures of the planets are all in ".jpg"format[4].

## IV. RESULTS

Some screenshots of the obtained results are shown below.

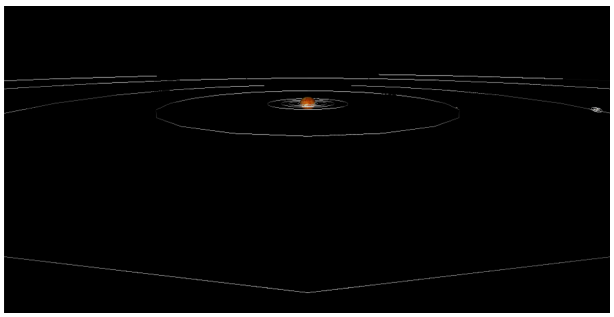

Fig. 2: Default view, centered on the Sun.

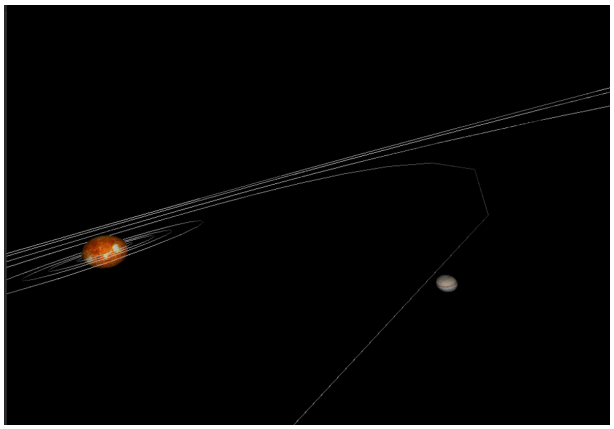

Fig. 3: View moved up to Jupiter, rotated.

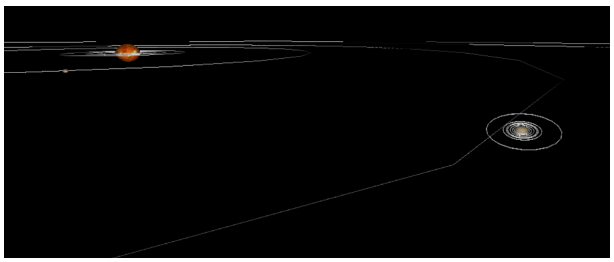

Fig. 4: View moved up to Saturn, slightly rotated.

Regarding the results, the most evident problem is the huge space occupied by the solar system, even though it is not perfectly accurate to reality. Therefore, to show the farest planets it is needed to move the view very far from the sun. Another problem occured is that even if the translation of the planet and the radius of the orbit are based on the same parameter (*orbit_distance*), sometimes the planets' movements seem to deviate a little from the respective orbits.

## V. CONCLUSIONS

In conclusion, the aim of the project is achieved. It is a pratical application in which computer graphics is used to model a realistic phenomenon such as the Solar System. It fullfils in particular the requirements of reality and approximated proportions. On top of that, the user can navigate in the environment through the use of the keyboard. One of the challenges in this project was setting up properly the programming environment with compatible libraries since the time during the course was too few in my humble opinion. To overcome this problem, useful tutorials on the web have helped[1]. The other big issue was the handling of textures, solved as explained earlier. Possible future improvements are the creation of satellites for each planet, or the movement of the view centering on each planet when pressing a key. Finally, a starred sky texture or some entertainment effects like sounds could be added too.

## References


[1] Opengl setup with visual studio. https://www.youtube.com/watch?v=gCkcP0GcCe0.

[2] Planetary fact sheet - metric. https://nssdc.gsfc.nasa.gov/planetary/factsheet/.

[3] Solar system wikipedia page. https://en.wikipedia.org/wiki/Solar_System.

[4] Solar textures. https://www.solarsystemscope.com/textures/.

[5] Textures in opengl. https://www.youtube.com/watch?v=n4k7ANAFsIQ.

[6] Prof. Nguyen Hoang Ha. *Project course on computer graphics*. Slides, a.y. 2018-2019.