[CG] Project Report (Phase 2)

Guilherme Sampaio A96766 | Luís Pereira A96681 | Rui Oliveira A95254 | Tiago Pereira A95104

April 2023

1 Introduction

The following report refers to the group project of the Computer Graphics class of 2023, authored by Guilherme Sampaio, Luís Pereira, Rui Oliveira and Tiago Pereira. These elements make up group 60.

The goal of the project is to create a simple graphics engine using OpenGL and GLUT in C++. It is divided in four checkpoints: this report refers to the second one only. In this checkpoint, an hierarchical transformation system must be implemented. To demonstrate this new functionality, a model of the solar system was required.

As a result, this report will be divided into two main sections: one about the implementation of the transformations, and the other about the making of the engine itself. There will also be a section prior to those explaining how to run the project. Some corrections have been made in order to make the code compile for Windows. As everyone in the group uses Linux systems, this problem went undetected in the previous phase.

2 Running the project

2.1 UNIX

Nothing has changed from the previous phase. Compile the project by running make and run it as the statement proposes.

2.2 Windows

To run the project in Windows, a CMakeLists.txt file was developed. All toolkits needed are included under toolkits/. The configuration is shown in Figure 1. In essence, the "*Where is source code*" should be the root of the project, *Where to build the binaries* the bin/ folder, and TOOLKITS_FOLDER the toolkits/ directory.

After configuring and generating, the Visual Studio project should be good to go. After building the solution, the project can be run in similar fashion to the UNIX version. The executables are placed under bin/Debug. The engine is called cg.exe and the generator generator.exe. All arguments are the same as their UNIX counterpart.

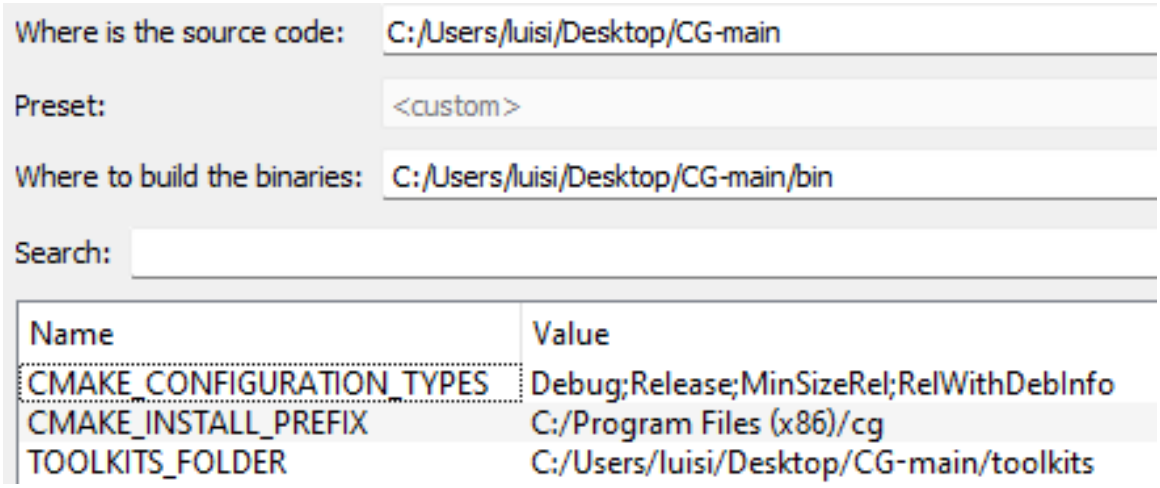

Figure 1: CMake Configuration

3 Transformations

In the second part of the project, it was required to implement three different types of transformations: translations, rotations and scales. To do this, some changes had to be made to both the XML format and to the way objects are rendered by the engine.

In the XML configuration file, now there can be multiple <group> tags, as well as a <transform> tag inside each <group> tag. In addition, each group can have nested subgroups inside it, making it a tree-like structure.

To address these changes, the class Group was created, as to store the information of a <group> tag in the XML file. This class contains, first of all, the list of its inner subgroups, as well as all the models and transformations to apply to it. Each transformation has its own class, which is a subclass of Transformation. To render a Group, one starts by calling glutPushMatrix. This will put a new matrix on the stack, meaning the transformations can be applied without interfering with the other previously computed ones. Then, the transformations are applied in the order they appear in the XML file, followed by the rendering of the models, as in the previous phase. Each subgroup is then recursively rendered and, finally, glutPopMatrix is called, removing the matrix from the stack completely. This makes the rendering of a group have no side effects for other groups.

4 Model of the Solar System

The required demo scene for this phase was a static model of our solar system. The XML file can be found in the xml/ folder of the project. All models were created with the generator program created in the previous phase. More precisely:

1. The model for the planets / moons / sun is a sphere with radius 1, 128 stacks and 128 slices;
2. The model for the ring of Saturn is a donut (torus) with radius 2, length 0.5, 100 stacks and 100 slices.

One priority for the model generation was to use a high number of slices and stacks, as to make them as close to a real sphere as possible.

The group tried making the model to scale. However, the very different sizes of planets, combined with the massive distance between them made it so that most planets would not be visible, or they would be very small. As a result, some scaling factors took place, as explained in the XML file via comments:

1. Distances between bodies were shortened by a factor of 50, except for the distance between the Earth and the Moon, which was shrunk by a factor of 25 instead;
2. Terrestrial planets are 10x bigger;
3. Gas planets are 5x bigger.

The final result of this scene is shown in Figure 2.

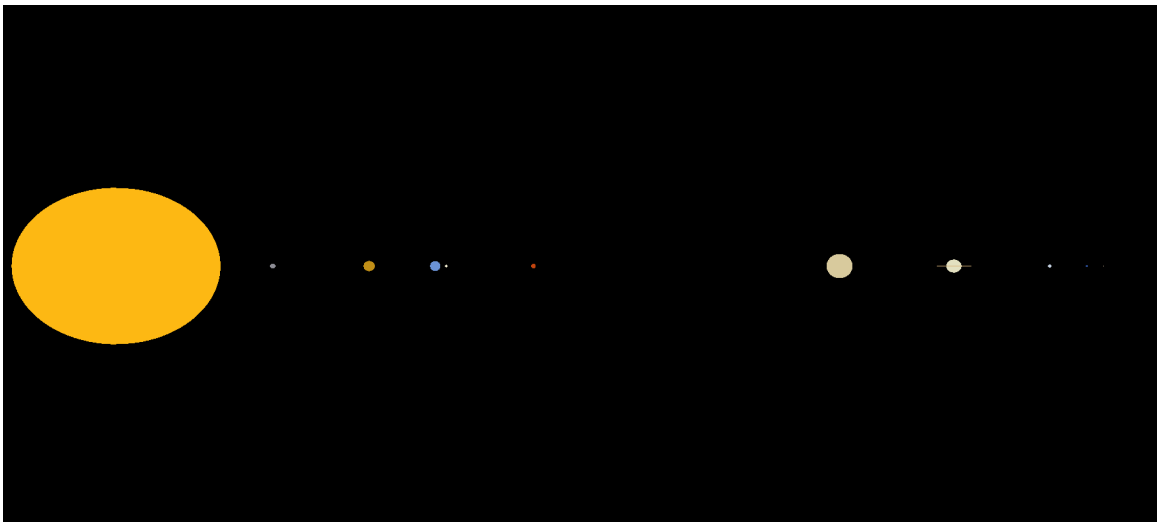

Figure 2: A simple model of our solar system (without the axis).

4.1 Additional Features

As can be seen in Figure 2, the bodies are colored. This was implemented by changing the call to glPolygonMode to use GLU_FILL instead of GLU_LINE.

The value of the color is being stored in the Model class, alongside its corresponding Shape. In the XML file, it is an optional extra attribute - color - of the <model> tag. That attribute corresponds to the hexadecimal color code of the string. The parsing and casting to the correct colors to pass to glColor3f function is done by the Model::parseHexColor static method. It is trivial: extract every value for each color in hexadecimal (ignore the leading '#' character, split the string in 3 groups of 2 characters), cast said value to an unsigned integer and divide it by 255. If no color is specified, the object is painted white.

In addition, the possibility of omitting the rendering of the axis was added. This was simply adding an optional attribute axis to the window tag. For that value to be set to false, the string

false must be provided, in any case. The value defaults to true. The group provides a model of the solar system with and without the axis.

5 Conclusion

As a result of this second phase, the group has delivered a system to represent and render transformations hierarchically as requested, as well as some additional features, like the coloring of models. This adds more realism to the requested model of the Solar System.

Overall, and in conclusion, the group believes this phase went well and is looking forward to the next iterations of the project.