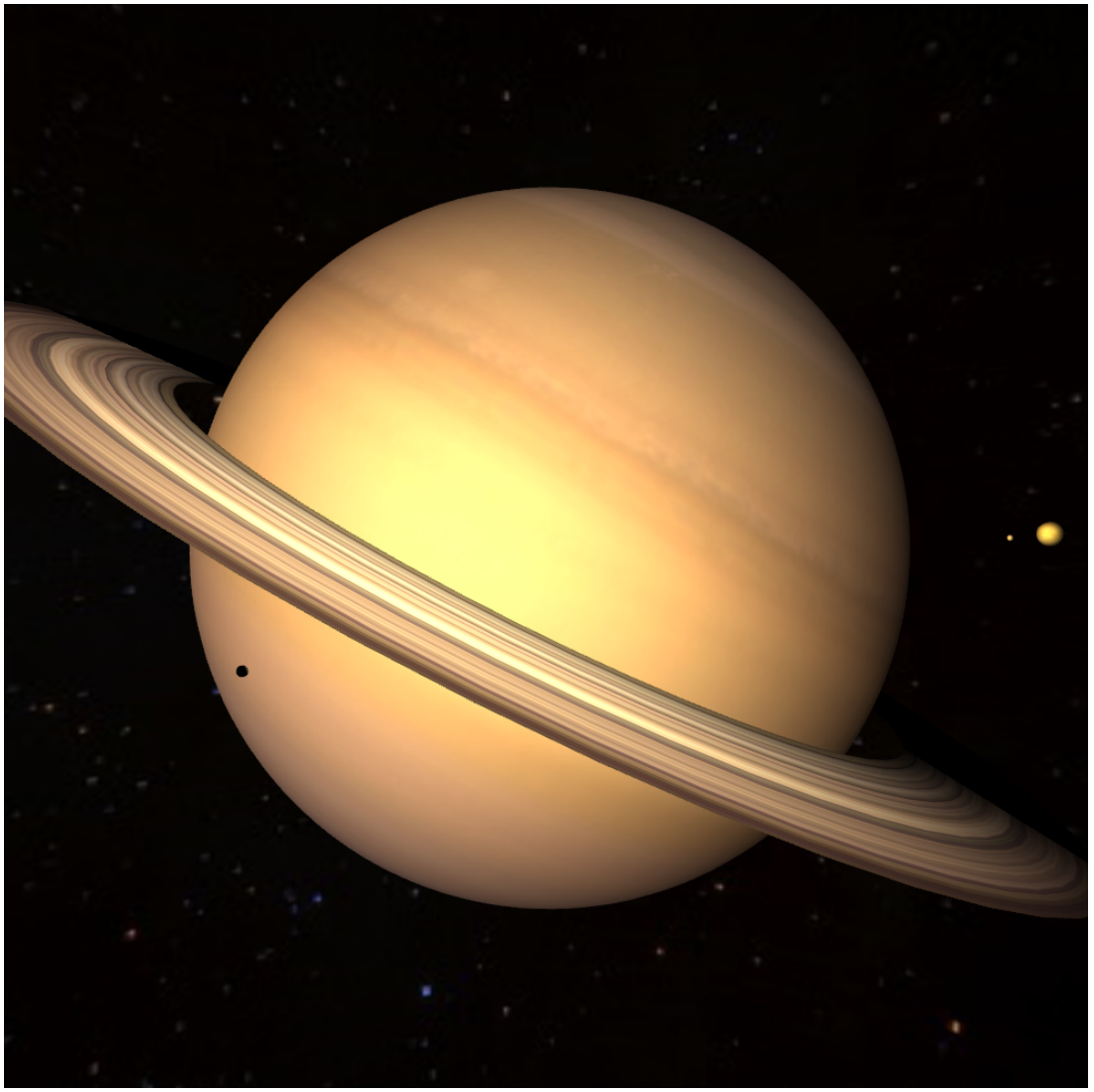
Project VisuSysSol

by Johan Ledoux

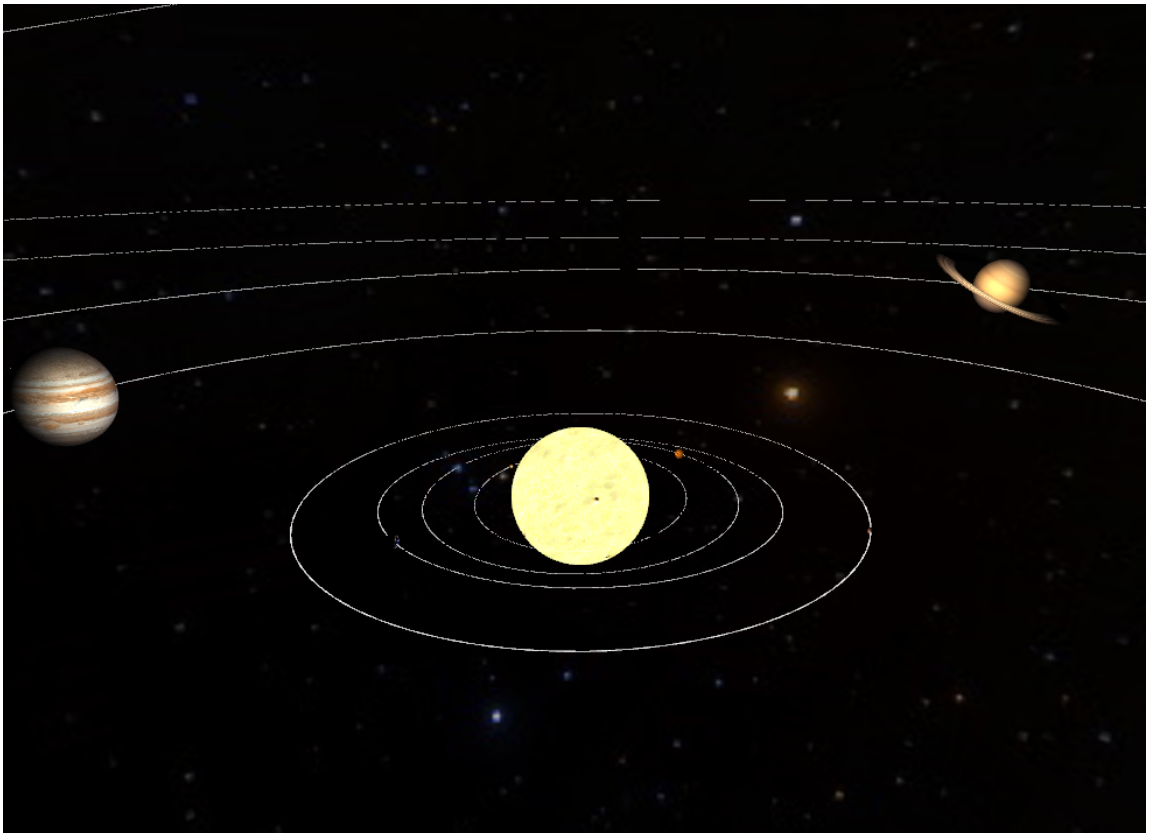
Scientific visualization of the solar system in C++ with OpenGL

Code repository : https://github.com/ZigzagAwaka/VisuSysSol



UGE Master 2 SIS - 2023/2024

Contents



How to use

Installation

After downloading the code from https://github.com/ZigzagAwaka/VisuSysSol, you can follow the instructions in README.md to install the project (but please, make sure to have g++ and cmake already installed on your computer).

Usage

The following section is there to present every possible interaction within the project window. Push the corresponding keys to do specific actions.

Left / Right click	By pressing a button of your mouse and then moving your cursor on the screen, you can rotate the camera in any direction.
Scroll mouse	You can zoom (or move forward) when using the scroll button of your mouse.
Z,Q,S,D	If using the freefly camera, allow you to move in any direction (keys are W,A,S,D on an english keyboard).
L	Line-render mode, every model will be drawn by polygon lines.
P	Point-render mode, every model will be drawn by polygon points.
F	Fill-render mode, every model will be drawn normally.
O	Hide orbits of planets, or unhide them if already hidden.
V	Modify the actual view of the scene, it will allow you to observe specific planets from a closer look. You can hit this key multiple times to loop through all planets and eventually go back to the global view.
T	Switch camera type between trackball and freefly.
-	Decelerate speed of the simulation (- key of the keypad).
+	Accelerate speed of the simulation (+ key of the keypad).
Spacebar	Pause or resume time of the simulation.
Escape	Close the project window.

Advices

When starting the program, the simulation speed is very high by default and it is recommended to slow it using the subtract(-) key when switching to a specific planet view, otherwise the moons will move so fast that it's going to be really hard to see.

When looping through the planet views, if you take a close look at Earth or Mars and then switch to Jupiter it's possible that the camera is going to be stuck inside of it (you will probably see nothing). It is recommended to zoom out of Jupiter to fix this.

When rotating the camera, it's recommended to only move the mouse very slowly or by clicking and releasing little by little, especially if using the freefly camera.

If you have a bad PC and the simulation is laggy, you can set the lowConfig flag (main.cpp line 10) to true before compiling the project in order to get less detailed models.

Code description

Textures are stored in assets/textures/ and they are copied automatically to bin/assets/textures/ during compilation.

Source files are located in the src/ folder. I also added personal source files in the glimac/src/ folder since they all use the template of the glimac library (Circle, FreeflyCamera and TrackballCamera).

Concerning the primary source files, here is a little description of them :

- **main** - It's where the program begins with the window creation and everything related to user inputs.
- **engine** - OpenGL related stuff such as textures and models loading as well as draw functions for every objects.
- **planets** - The "*global library*" of planets parameters, functions and shader programs. Used to automate every draw function in **engine**.
- **camera** - Manage camera movement.

Choices

Global explanations

Because the goal of this project was to make a scientific representation of the solar system, distance, diameter and axis or rotation needs to be chosen carefully. However, the size of planets does not need to be exact even if their ratio needs to be real. And by having a "fake" size, distance between objects is also going to be wrong.

This is why I decided to draw every planet with ratio parameters between them. It's also a way to make something great to look at.
With this, I decided to add a lot of offset and factor values to almost every planet parameter (this can be seen in planets.hpp lines 58-67). I adjusted every value by hand to have the best looking result while still having a scientific-valid ratio between each parameter of planets.

As for the rotation of planets, I didn't manage to make them orbit on an ellipse, so I chose to make them orbit on a simple circle represented by the average distance to the Sun.

When observing a planet view, I chose to only draw the asked planet (with moons) at the center of the scene because the contrary was not mentioned in the subject and this was easier to do. However, even if the Sun is not drawn in that case, the light received on the planet is adjusted just as if it was present.

Order of representation

By looking in the comments of the code, it's possible to read something like "global order". This is a way to know at which index will come which planet, and this order is displayed at the beginning of planets.hpp (line 17).
Functions that loop through every object are using the global order (load textures, display a specific planet and moons, …) and it is synchronized to PlanetParams and OrbitIndexs structures in order to have everything drawn automatically.

Project management

I worked on this project alone (I failed to recruit a friend, that's why).

I started the project by copying my files from the practical sessions and I modified everything after. This was an easy way to start with a solid base because a lot of major stuff was already made by my past-self (rotation, light and camera, …).

As for things that didn't work. As I said in the previous section, I didn't implement the elliptic rotation of planets simply by lack of understanding.
I also wanted to make a more realistic Sun with a fragment shader but this was more complicated than I thought, so I simply added a little "moving" effect on the surface of the Sun and gave up the original idea.

And finally, I didn't adapt the light on moons and they might be dark even if the Sun should be lighting them (I kept the same code as planets, so they "think" the Sun is the planet to which they are rotating to and I was too lazy to fix this).

Results

Some pictures of the project in action: