Interactive Graphics - Final Project

Solar System

A.Y. 2018/2019

Gianmarco Cariggi 1698481
Marco Costa 1691388

10th July 2019

1 Introduction

For the final project of the course of Interactive Graphics we decided to make a simulator of our solar system, trying to reproduce it in the most faithful and realistic way possible, using of course assumptions and approximations to simplify its implementation.

1.1 Requirements

- Hierarchical models
- Lights
- Textures
- User interaction
- Animation

1.2 Environment

The project is developed in WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the use of plug-ins. It is fully integrated with other web standards, allowing GPU-accelerated usage of physics and image processing and effects as part of the web page canvas.
Click here to check if your browser supports WebGL.

1.3 Libraries not developed by the team

- Three Js (rev. 107dev): a cross-browser JavaScript library and Application Programming Interface (API) used to create and display animated 3D computer graphics in a web browser using WebGL.
- Materialize (v. 1.0.0): a design language that combines the classic principles of successful design along with innovation and technology.
- jQuery (v. 3.4.1): it takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code. Italso simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

1.4 Assumptions and approximations

As is well known, the planets follow an elliptical orbit (a "flattened" circumference), and the sun occupies one of the two focuses. Rotating along this orbit, we have the 4 seasons (also due to the inclination of the axis). For simplicity of implementation and positioning of the various planets, we have approximated the orbit with a circumference, and the sun occupies the center.
Another important fact is that the orbit of all the planets varies continuously (for example, the orbit of the earth is not the same from one year to the next), this is because there is the gravitational influence of all the celestial bodies present in the universe (and in particular in the solar system, being closer). In the solar system, besides the sun, Jupiter has an important influence, being the biggest planet. In our simulator we do not take into account all this, the planets revolve around the sun with the same orbit (we can say that the sun is the only body that influences the gravitation of the planets).
The last detail is the actual position of the planets. Probably they will not be located in the real position, because in our simulator this position depends on the date, and in javascript the function Date.getTime() returns the milliseconds passed from January 1, 1970 to the moment in which the function is called. In theory we should have the milliseconds passed from the birth of the solar system to today, or set a certain initial position so as to match the real position of each planet with that in the simulator. But this is very complex, so we preferred to concentrate on other things. It must be noted however that every planet makes a turn around the sun in the real time of revolution and makes a turn around its axis (inclined with respect to the equatorial plane) in the real time of rotation (for example the earth takes exactly one year to make the revolution around the sun and 24 hours to make a turn on itself).
Here it's possible to see the real position of the planets.

2 Usage

3 Implementation

Appendices

A Sources

- Wikipedia - for data and descriptions
- Threex.planets.js - for loading earth clouds texture and have a ring geometry with a radius texture
- Planet pixel emporium - for textures