Solar System 3D Simulation

Project Writeup

Javier Ramirez Moyano

For my Perception and Multimedia Computing term 1 project, I coded a 3D drawing example that includes both camera positioning and lighting, as well as animation. This drawing is an interactive three-dimensional simulation of the solar system, which shows how the eight planets in the system orbit around the sun and how these planets reflect the light that the sun projects. The sketch was coded in p5.js, using the WEBGL render mode, which allows for 3D sketching. I used the following tutorial as a reference to navigate WEBGL and everything that it offers: *github.com/processing/p5.js/wiki/Getting-started-with-WebGL-in-p5*.

This simulation has been designed to be informative and educational, therefore it has been a priority to make it as realistic as possible, within reason. Interactivity is also important, so a few interactive elements have been implemented. Using sliders, users can change different elements of the simulation, such as the simulation speed, the orbit distance, and the type of lighting. The simulation speed determines how fast the planets circle around the sun. Even in the slowest setting, the simulation speed doesn't match the real speeds of the planets by far, as the fastest of the eight planets (Mercury) takes 88 Earth days to complete one full orbit around the sun. Users are also able to control the distance between the planets and the sun. This distance ranges from 1/1111 to 1/48 of the real distance in the closest and furthest settings, respectively. This is also highly unrealistic but it had to be represented this way because if the real distance of the planets was represented in the simulation, it would be impossible to see any of them on the screen. On the other hand, the distance between the planets themselves is always proportional to the real distances, regardless of the orbit distance. The size of the planets and the sun is also made to scale so that users get a good idea of the relative proportions of each planet. All the information regarding the solar system and the planets in it, i.e. orbit distance, orbit period, planet diameter, etc., was taken from the website *nineplanets.org*. Lastly, users can choose the type of lighting in the sketch. The two options are "Ambient Light" and "Realistic Light". The ambient light provides an omnidirectional light to the sketch, which lights up all the planets and the sun equally, while the realistic light only lights up the planets on the side that faces the sun. Apart from these three aspects of the simulation, users are also able to control the position of the camera using a group of three sliders. These sliders control the X, Y, and Z-axis of the position of the camera, while the center of the sketch, i.e. where the camera points, remains on the sun. This gives users the freedom to "look around" while the planets orbit around the sun.

I believe this project is a good illustration of three-dimensional sketching and animation, as it shows both of these working together. Combining these two elements was a challenge at times, and it was necessary for me to apply the concepts learned in the labs (especially labs 5, 6, and 9) and also look for outside sources. Looking for outside sources proved to be really helpful, as there is a lot of p5.js documentation on the web.

After I finished coding the first demo, I asked my peers for feedback, and there were a few points that came up repeatedly. I focused on fixing those issues in order to improve my program. The first issue was that the "Realistic Light" made it hard to see the planets, especially the small ones. I thought that giving users the option of changing the type of light would be a good solution for this, without having to take an important aspect of the simulation away. Another piece of feedback was to put labels on the sliders, which at first I thought I couldn't do because when text is placed on the screen using WEBGL mode, it moves as the camera moves. The solution to this problem came at the same time as the solution to the first one. When I placed the buttons used to change the type of light, I realized that buttons do not move with the camera, so they can be used as labels. Lastly, a couple of people suggested implementing a button that resets the X, Y, and Z-axis slider values to the initial values, so that the camera's point of view would go back to its starting point. In the end, I was not able to implement this functionality.

Other sources I used in this project are *editor.p5js.org/kjhollen/sketches/ryZBahkKx*, in which I based my code to calculate the planets' locations, *1001fonts.com*, where I downloaded the font used in the sketch, and lastly, the p5.js.org references and examples.