Melissa Banoen-Garde
Creative Computation I
CART 253 – B
Pippin Barr

Proposal

Artistic Vision

Historically, our relationship with space has evolved yet its proximity and relevance to us remains constant. Throughout the course of civilization, in all eras and regions of the earth, we've looked to the heavens either for meaning or to decipher its universe. What nudged my curiosity are the various ways we contextualize our relationship to its nature – physically and metaphysically. What stood the test of time will (and always be) the polarity between ancient astrology and modern astronomy. These coexisting elements made 'magic' feel almost tangible throughout my upbringing. This wonderment from a distant memory will be the starting point in my attempt to program my final project; an interactive simulation of the solar system with information and lore.

Plan and technical challenges

1. 3D: `createCanvas(windowWidth, windowHeight, WEBGL);`
    - For this project, I decided to explore and implement WEBGL's render mode, inspired by Sylvain Tran's and Amanda Clément's final project from 2019. The purpose of this decision is purely based on its 'realistic' demonstration of the solar system. The introduction of a third dimension, the z-axis, will be the core challenge. Most specifically, executing motion in a 3D cartesian plane.

2. The nine planets: `sphere();`

- I've experimented with p5's 3D primitives as a means to understand its syntax and relationship with the 3D map. It was fun! I will define a superclass of planets with an empty display and motion method. The planets' superclass will extend to eight different subclasses, defining each with the purpose of emulating the eight planets.
- The challenge, although doable, will be to achieve a relatively 'accurate' orbiting motion around the sun. In numerous attempts, I've discovered that there's a relationship between `translate(x, y, z)` and `rotateY(#)`.
- Another more general challenge ("general" because I still confuse myself and take a while to program when doing so) will be to figure out a way to program the planets in an effective, organized, and neat manner using object oriented programming and arrays. So far, I've programmed the planets using only one class therefore my script is quite long but—for the prototype—I will use it as base to reorganize later. This challenge is doable as I can go back and forth with our class' notes, videos, exercises, etc.

3. `class Star.js`

- The challenge here will be to introduce the third dimension, z-axis, so the stars distribute in the depths of the 3D map. And, once again, *arrays*.

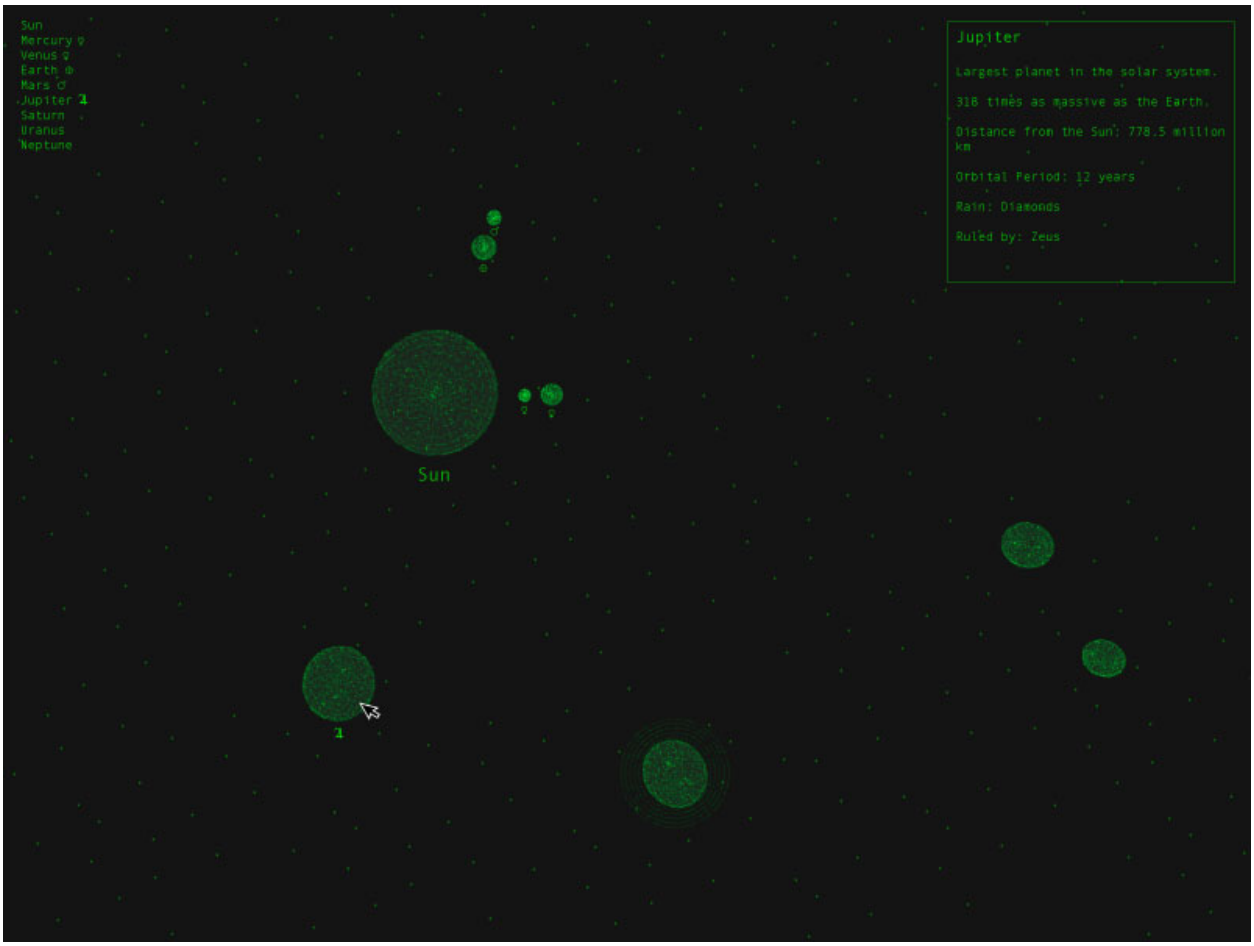
4. `orbitControl();`

- For the purpose of interactive experience, I will add this feature that allows the user to click and drag/move around the simulation.

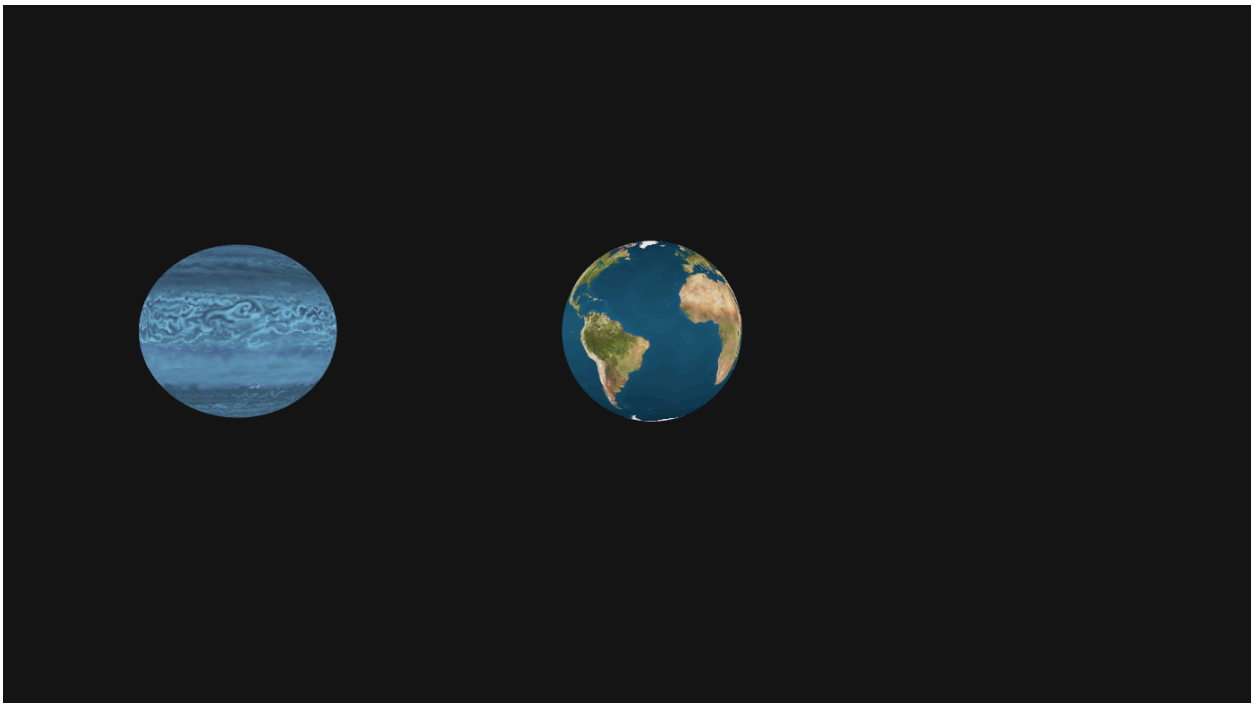
5. `mouseHover(); / mousePressed ();`

- A method in which I will add in the planetary classes which will allow the user to read each planet's respective specs, infos, and lore.
- A prospecting idea: If I have time and if I can figure out a way for the simulation to change visual settings between a blueprint-type of simulation and a more *realistic* 3D appearance involving `textures()` by pressing a key or a button, I will incorporate this feature for aesthetic purposes.

Visual sketches



This will be the overall mood / theme I aim to achieve; a monochrome monitor visual. The screen's contents will consist of the planets orbiting around the sun, identified by their own symbol which is set beside their name (top left corner) for reference. A `mousePressed()`function will trigger a box of information (top right corner).



This is a screenshot from experimenting with WEBGL rendering and `texture()`. Depending on whether I have the time or not and if I achieve its code, I would like for the aesthetic to change when a key is pressed or `mouseClicked()`. Maybe this could be an event class or a method in each classes.