COS426
14 May 2019

Shamailah Azam (sazam)
Simisola Olofinboba (simisola)
Niranjan Shankar (nshankar)

NBODY 2.0: 3D Solar System Model

Introduction

Nbody 2.0 is a three dimensional model of planets orbiting the solar system. The name was derived from the Nbody assignment from COS 126, in which students had to develop code to model a 2-dimensional solar system.

Our baseline goal for the project was to have a model in which all eight planets in the solar system orbit the sun based on the calculated force of gravity between all the objects in the solar system. However, in order to add some additional complexity to the project, we established a few “stretch goals,” such as features to navigate the scene, options on the UI to add and delete planets, setting non-uniform planes of rotation for the planets, and adding additional objects (asteroids, comets) and moons to the planets. NBody 2.0 was able to successfully meet all these baseline requirements while also implementing many of these “stretch” features.

Our project uses three.js to represent our planets as sphere objects, combined with textures to wrap around each planet. Three.js libraries also enabled us to use cameras to traverse the scene.

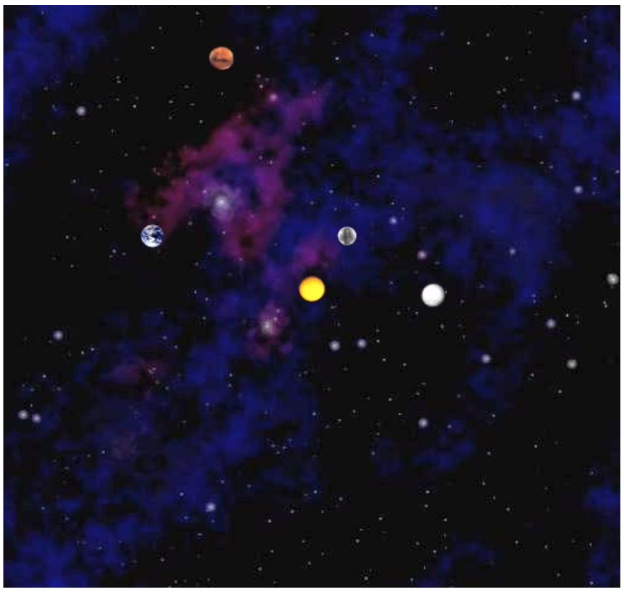
Previous Work

Nbody

Nbody is one of the assignments for the *Computer Science: An Interdisciplinary Approach* course at Princeton. The purpose of the assignment is to code a two dimensional model of the solar system with “N” bodies. The student’s role is to code the “physics” aspect of the program - that is calculating the force, acceleration, velocity, and position of each of the bodies in the solar system in order to model the system. The approach of the program is as follows:

1) Read in information about the universe (dimension, number of planets), data about each body in the system (mass, starting position, starting velocity), and the length of the simulation.
2) For each planet, calculate the net force acting on the planet in the X and Y directions.
3) From the total X and Y forces, update the position of each of the planets based on the acceleration and velocity in the X and Y direction, as well as the timestep.
4) From the new X and Y coordinates of the planet, redraw the planet using Princeton’s StdDraw.

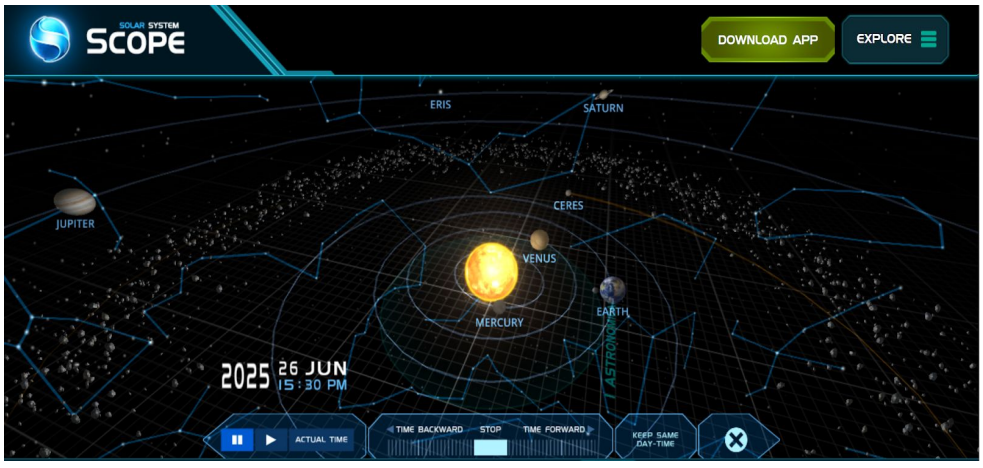
The result is a 2-dimensional simulation of the planets orbiting the solar system, as depicted in the images below. The planets that appear in the simulation, as well as the simulation speed and duration, depending on the specific file that user inputs upon executing the program.



3D Online Models:

There are several online platforms for representing 3D models of the solar system. Among these are Solar System Scope, SkyLive, and Project-Metis. Most of these models are interactive and are mainly for educational purposes: for instance, clicking on “Jupiter” would zoom in on the planet and open a panel containing information about it. The planets are all on fixed orbits, though these orbits are set up to actually represent the actual shape and trajectories planet orbits in the solar system. Many of these also contain options on the User Interface to alter the simulation speed and rotational speeds of the planets.

The following is a screenshot from one of the programs (Solar System Scope).



As we see, the Solar System Scope has the planets on fixed orbits, with options to speed up the simulation, pause, resume, and even go back in time. The orbits also seem to resemble the rough shapes of the planets' actual orbits. The User Interface is relatively intuitive and navigable. There are also other noteworthy objects in the solar system, such as asteroid belts, constellations, and dwarf planets. The model is hosted at: https://www.solarsystemscope.com/.

Approach

Our project incorporates both the 3D online models and the Nbody assignment - we provide an interactive 3D model in which the orbits are determined by the net force of gravity acting on that body. By using the properties of each of the bodies, we could calculate their planetary movements based on the forces acting on them. As with NBody, we also need to determine the appropriate starting positions and velocities for each of the planets. However, one noteworthy way in which our Nbody 2.0 differs from the original NBody in terms of physics calculations is that we have the planets revolving around the sun along all three axes - this involves calculating the net force along all three of these axes. For some the planets, we offset the starting position in the y-direction, giving it a noticeably different plane for revolving around the sun. Though the starting velocities and distances are used differently in our model compared to Nbody, these values were still helpful in determining the data for each of the planets.

Using the three.js library, we could visualize the bodies in a 3D system and adjust the orbits to make use of different planes of revolution. We added a control system that allows users to navigate through the solar system through zooming and panning with the scroll and arrow keys as being the physical controls. They can also control which of the bodies appear in the system. Bodies can be removed dynamically. There is also a pause/play functionality so that the user can stop the orbital movement to get a closer look.

Methodology

In order to simulate the solar system, the essential pieces were the sun and the seven planets, Mercury, Venus, Earth, Mars, Neptune, Saturn, and Uranus. For each planet, a sphere was created with the appropriate texture to make it appear as similar as possible to the planet's actual surface. Although the technical shape of the planets and the sun is an oblate spheroid, we made a tradeoff to use spheres as our model shape for simplicity. For Earth, in particular, we also added specular and normal map textures in order to make the Earth's surface look as textured as possible. We even layered the Earth's surface with clouds to render the atmosphere as realistically as possible. In addition, we added the moon and set it to orbit around the Earth. For the sun, we placed a point light at the center of the sun to make it appear as if the sun was emanating light. The next step was to add the physics to get the planets orbiting around the sun. We also added interesting objects into the scene such as an astronaut, an asteroid, and a rocket. We added controls so that the user can add and remove planets and objects and easily

navigate the scene. Users can use their mouse or trackpad to zoom into the solar system and on certain planets, as well as view the scene from different angles. Users can also use the arrow keys to move through the universe. We also added a starry background which is a static image that stays in place even as the viewer moves through the system. This is done by creating a separate scene and camera for the background that is rendered before the other objects in the system. The GUI that is used for interactions with the scene is built using the dat.gui library that was used in previous assignments (documentation: http://workshop.chromeexperiments.com/examples/gui/#1--Basic-Usage).

The motion of each planet is set in place based on the net forces in the X, Y, and Z direction acting on it. In order to arrive at this stage, we took the following steps: first, we set one sphere (planet) on a fixed path around another sphere (sun). Next, we designated a specific mass to the planet and the sun, and calculated the force of gravity between them. For the sun, we used the actual mass of the sun in kilograms, as was done in Nbody (1.989 * 10^30), and for the planet, Earth's mass was arbitrarily chosen for the time being (5.972 * 10^24). Once the masses were determined, we calculated and tinkered with the starting velocity, starting position, and distance from the sun needed to keep the planet in a circular motion around the sun. Next, all the other seven planets were added to the system, with the force calculation being solely between that particular planet and the sun. When these planets were set in circular motion, we altered the force calculation to also consider the force of gravity that the planets exert on each other. Our final step in terms of the planetary motions was offsetting the y position of some of the planets, so that we could have some planets moving along all three axes as opposed to just along the x-z plane. It is important to note that in real life.

Results

Prior to embarking on our project, we developed a list of features that we wanted our minimum viable product to have. We measured our project's success in part by if we were able to implement the minimum required features as well as how realistic, aesthetically pleasing, and engaging the solar system scene was. Anything that will be added to the system will be improving how realistic it is. We were able to successfully meet the minimum requirements and establish a working model of the solar system with the seven planets orbiting around the sun. We then moved onto implementing our stretch goals such as space-themed objects like an astronaut, rocketship, and asteroid. We added additional features like being able to pause the simulation so that users could get a better view of the scene.

Discussion

This approach was rather promising because it allowed us to build a pretty aesthetically pleasing and physically accurate with the tools provided by the three.js and dat.gui libraries, as

well as downloaded meshes and textures. This approach also works well as it can be easily expanded upon to add more details and reflect more of the complexities of the solar system.

If we have more time, we would have like to add more elements of the solar system and universe into the scene, such as other stars, asteroids, and other space-themed objects like satellites and spaceships; different ways of navigating the system such as following a rocket ship through the system or clicking on bodies to zoom into their features; adding a trail in the orbit of each planet to get a zoomed-in view of its features; allowing users to drag objects and place them in their location of choice in the scene; allowing users to resize objects. In order to add an “educational” element to the project to make it similar to online 3D models, we would also like to make the planet sizes scaled relative to each other (with exception to the sun), and also have the orbital periods and speeds scaled relative to one another too. For instance, since Mercury’s orbital period is 88 days, we would have it orbit 365/88 = 4.14 times as fast as the earth. Uranus is also known for its unique rotation, since it is the only planet in the solar system to rotate along the z or x-axis (depending on the position in its orbit) as opposed to the y-axis. Additionally, having an option to click on a planet and open up an informative panel about that particular planet would be an exceptional feature to have on our platform.

Through this project, we learned how to build and manipulate a 3D scene. This includes dynamically controlling and calculating the positions of objects within a scene, controlling navigation through the scene, and making a scene partially user controlled. These skills could easily be applied to future projects such as making games, virtual reality scenes, and other 3D simulations.