PS2 N-Body Simulation

The purpose of this project was to mimic how pairwise forces affect particles in a universe. PS2a and PS2b were the two parts of this task. The first section of the task required you to create a representation of our galaxy using two classes: one to represent each planet and another to govern each planet. The initial part's sole purpose was to ensure that planets could be read from a text file and presented appropriately in the SFML window. The second section of the project involves simulating the effects of paired force using accurate physics calculations. The main function was required to accept data in the form of overall simulation time, time spent in one simulation step, and the input file containing information about each planet.

Key Concepts

The Universe class may be overloaded to allow for input redirection from a file to set up the planets for simulation. CelestialBody and Universe are both sf::Drawable, allowing them to be drawn to the window. Within the Universe class, each CelestialBody is created with a std::unique ptr and stored in a std::vector. The planets are then rendered using the draw function on each CelestialBody in the vector in a loop in main. The majority of the calculations for this program are done in Universe's step() function.

What I accomplished

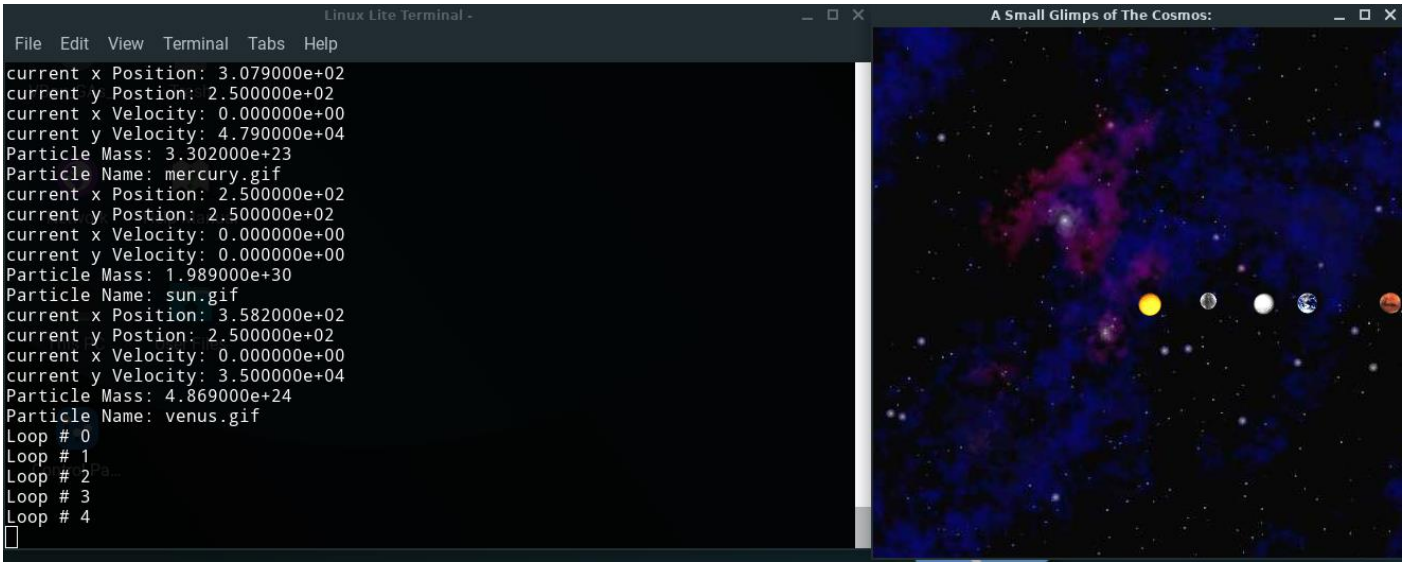
PS2a: Universe is a class which contains a vector of CelestialBody(s) and has a function to draw the planets by accessing a vector. For extra credit, I added a background image called "A Small Glimpse of The Cosmos:" and used smart pointers to avoid data leaks. The class Universe was a vector of CelestialBody which was used to hold all the planets that were fed into the program from planets.txt. I used a smart pointer to represent the universe, closing off the possibilities of memory issues(unique_ptr <universe> u(new universe());). The universe was allocated through a for loop. I had to make a pushback function in order to assign CelestialBody(s) to the vector.

PS2b: The step function was implemented in reference to the homework pdf. The step function is fed all the necessary forces via the file. Through each loop of said function, it sorts through and initializes each planet with the necessary x and y positions and velocities, acceleration, net force, force, gravitational force, and mass. Smart pointers were used to create new objects and pushback. Everything else was pretty much piggy backed off part a. I did fix my Universe class and implemented it properly. Though, looking back at it, I probably should still have kept it as its own header.

What I Learned

This task taught me a great deal. For starters, this was the first time I utilized smart pointers to initialize objects in a C++ application. I gained a lot of knowledge about the physics involved in the N-Body simulation utilizing the leapfrog finite difference approximation approach in addition to C++.

Output


Linux Lite Terminal -
File Edit View Terminal Tabs Help
current x Position: 3.079000e+02
current y Postion: 2.500000e+02
current x Velocity: 0.000000e+00
current y Velocity: 4.790000e+04
Particle Mass: 3.302000e+23
Particle Name: mercury.gif
current x Position: 2.500000e+02
current y Postion: 2.500000e+02
current x Velocity: 0.000000e+00
current y Velocity: 0.000000e+00
Particle Mass: 1.989000e+30
Particle Name: sun.gif
current x Position: 3.582000e+02
current y Postion: 2.500000e+02
current x Velocity: 0.000000e+00
current y Velocity: 3.500000e+04
Particle Mass: 4.869000e+24
Particle Name: venus.gif
Loop # 0
Loop # 1
Loop # 2
Loop # 3
Loop # 4
A Small Glimps of The Cosmos: