AP CSP Summer Explore: N-Body Simulation

Cade Brown

October 10, 2016

Abstract

(8a) We study the use of computational tools to aid in providing an approximation visual simulation to multiple bodies each with a mass and size in order to better understand and study how the stars, planets, asteroid, and other cosmic bodies interact with each other through laws of gravity. The goal of this is to create a program in which you may study how different starting parameters, such as position, velocity, number, modified gravity constant, and mass of bodies affects the motions, as well as simplified simulations which illustrate the chaotic motion of the N-body problem, and its applications toward modeling our universe.

1 Introduction

In this section, I will define certain functions, relationships, and assumptions which should be held throughout the paper.

1.1 Relationships and Equations

We will be using the Newtonian model of gravitation [1], i.e.

$$F = G\frac{m_1 m_2}{r^2}$$

Where F is the force between two bodies in newtons, $G \approx 6.674 \times 10^{-11} \frac{Nm^2}{(kg)^2}$ is a constant m_1 and m_2 are masses of two bodies in kilograms, and r is the distance between the center of masses of the two bodies in meters.

1.2 Implementation

(8b) Due the speed at which bodies in real space move is so slow compared to their scale that any simulation would take months of constant video to see a practical difference in the space. In order to develop a faster, and still approximately accurate simulation, we increase the G constant so that movement and interaction is increased. We can change this at any time, though for us, it is 10^4 times the original value.

In order to stay within 32bit floating point and 64bit double floating point values, we must create a scale for in game units (abbreviated in this paper as

$unit_{(IGU)}$, like $km_{(IGU)}$ for the conversion rate between KM and IGU units). These can all be found in the GitHub repository [2], but I can list them off here:

$$km_{(IGU)} = 10^8 km$$
$$kg_{(IGU)} = 10^{12} kg$$
$$G_{(IGU)} = 10^5 G$$

So, the gravity coefficient is 10^5 times as large in our simulation, and we did this simply to speed up the movement, like changing the time scale of the whole simulation.

(8c) There are Bodies, and Body Spawners. Bodies each have a mass, in $km_{(IGU)}$, and some sort of polygonal mesh (which is normally approximating a sphere). When the simulation starts, the spawners all create bodies with random positions, and random initial velocities, and then physics calculations are done which detail forces and movement between bodies, which are then rendered to the screen. The initial data is simply 3d coordinates which detail the list of bodies, and spawners, and these positions are updated to relay movement to the screen.

1.3 Potential Problems

(8d) One significant problem that some time optimized implementations might run into is memory storage. For a system with N bodies, my program runs in $O(N^2)$ time complexity, and $O(N)$ space complexity. There exists a trade off of storing all computed forces and computing them twice that essentially halves the time spent computing, but increases space complexity to $O(N^2)$, while keeping time complexity at $O(N^2)$, although it is twice as fast. This can get very expensive in RAM, so I have chosen to opt for the cheaper memory requirements in order to keep space complexity down.

2 Program

To demonstrate this simulation, I have written a program to model it, with some controls for the users to view the bodies in motion. You must have a new version of Chrome or Firefox for this program to function properly, see [3] for a webpage demo of the program

2.1 Controls

Use WASD keys for orbit control, use R to zoom in, and F to zoom out. Press ESC to bring up the menu. Pressing escape again toggles the menu
While in the pause menu, you can activate other functionality by clicking the buttons on the menu. You can press the buttons there to stop the simulation, reset the simulation, or change the gravity scale (default is 10000 times real world).

3 Effects

This section will explain possible effects of this computing innovation.

(8e) Utilizing this technology, international space agencies as well as domestic agencies can simulate motion of the planets, to plan for things like: cargo transfer, spacecraft landing, and other cosmological events, such as collisions between bodies or orbital patterns [4]. Obviously, to correctly model the universe as it is, we must have a very large amount of bodies being computed against. This takes very much computational power, as well as time to collect data about the bodies. However, with this simplified program I provide, we learn more about how gravity works.

(8f) The main caveat of the system would be the fact that you must have accurate data to provide an accurate simulation. So, if we had incorrect or incomplete data, our simulation might incorrectly or incompletely model the real path of our bodies. For example, if we wanted to model Earth's orbit around the sun, create celestial bodies for each planet and the sun, then run our simulation, we might get slightly off results because we do not account for things like: asteroids, moons, dwarf planets, or comets that may alter the paths [5]. Another simplification that must be made are the size of the particles. If we were to track every single atom, the storage space alone would be astronomical, and the computational time much more, so I have opted to have large particles that clump together to approximate bodies in motion.

4 References

(8g)
[1] Isaac Newton's Principia Mathematica: PDF
[2] The project on GitHub: Source Code
[3] Online version of gravity sim: Webpage
[4] ISM Properties in Hydro Dynamic Gravity Simulations: arXiv
[5] KBOS Comets, Colorado U: Article