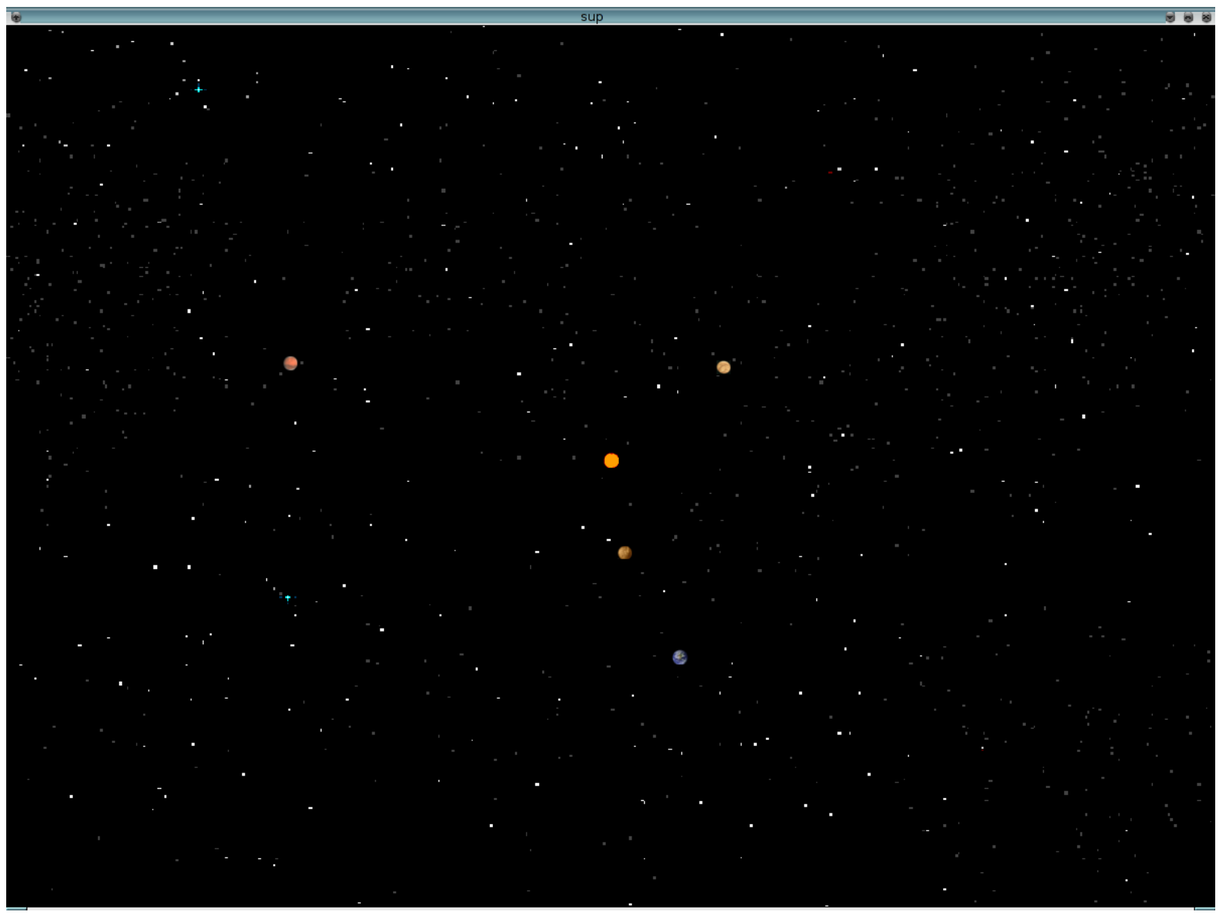
Creating 2-D and 3-D models of the Solar System using physics-based geometries in Java

Brian Tubergen

ABSTRACT

Basic models of the Solar System that involve predetermined paths for planets according to circular or even elliptical orbits can be effective for simply estimating the basic motion of the planets, but a more advanced and accurate model requires iterative physics calculations for an N-body problem. Even these real time physics calculations in and of themselves aren't particularly useful other than for visualization (although visualization has merit; indeed, 3-D graphics are worth implementing), but the ability to add additional solar bodies to the system and view the Solar System's reaction to their presence is valuable for experimentation.



1. INTRODUCTION

Keplerian models of the Solar System in which planets follow an "on wire" path of motion are very common, and indeed, even Solar System simulations that involve actual physics calculations are available. NASA's *JPL Solar System Simulator* (http://space.jpl.nasa.gov/) is one of these simulators that makes use of advanced physics equations and relevant corrections to the physical models, and one goal of this project is to recreate a Solar System Simulator and display animations of the planets' motions in real time.

Although simple Solar System simulations exist, very few of them allow users to interact with the simulation. The hope of this project is that the simulation will allow users to place a solar body at a location, assign that solar body a mass, velocity, and direction, and see what happens to the Solar System.

2. BACKGROUND

The aforementioned user interaction with the simulation would have a distinct purpose in that it would allow users to draw conclusions about what happens to the Solar System upon the entrance of a solar body. According to Daniel Perley, the passage of a body like a star into the Solar System is "an occurence which is actually not impossible in the Sun's lifetime." (http://astro.berkeley.edu/~dperley/programs/ssms.html)

Perley's Solar System simulation didn't originally implement real physics, which is an improvement I'd like to make over his model. The graphical elements of his simulation were also rather limited, whereas my project would eventually strive to implement 3-D graphics.

It's been decided that for now the project won't attempt to model advanced relativity corrections to older models of planetary motion and for the moment will simply focus on implement a Keplerian model of the solar system using iterative force calculations. The most relevant equation that follows from this model is F = ma = G*m1*m2/r^2. One can then solve for the acceleration of a planet due to gravity, and thus the motion can be simulated.

3. SOLAR SYSTEM SIMULATION

The previous section discussed the limitations of previous projects and how my program plans to expand on them; this section discusses the specifics of how my program works. As previously mentioned, algorithms follow a Keplerian model using iteratively calculated accelerations. Time steps can be changed to alter simulation speed and precision. Data is gathered from the real world both to set initial positions and velocities and to compare my simulation's output to actual results.

3.1 Display Class

The display class, dubbed *Animate01_modified.class* for the moment, creates the planetary objects and handles the program's graphical output. It then loops over some number of time steps. At each time step it renders planetary sprites onto a predetermined background, telling the planet objects to update their positions, and updating the graphics on-screen.

3.2 Computing Positions of Planets

At each time step, the display class passes each planet object an array of the current positions of all the planets. The net acceleration of each planet is calculated relative to all of the other planets; Each individual planet loops over the array of all the other planets and calculates it's net acceleration based on the equation $a = G*M/r^2$. It then updates it's velocity and position accordingly, and the display class updates the graphical display accordingly after all planets have finished these calculations.

3.3 Comparing to Real World Data

Data for real world planetary systems has been gathered from NASA's *Horizons* system from March 13, 2006 to March 13, 20067. Initial values for my program were adjusted to match initial values on March 13, 2006. Position data output from my simulation is uploaded to a text file while my simulation runs. Following completion of one year, that data is then compared to NASA's data at certain instances, and statistical analysis is done to verify the accuracy of my simulation.

4. RESULTS

Code to compare to real world data not yet written.