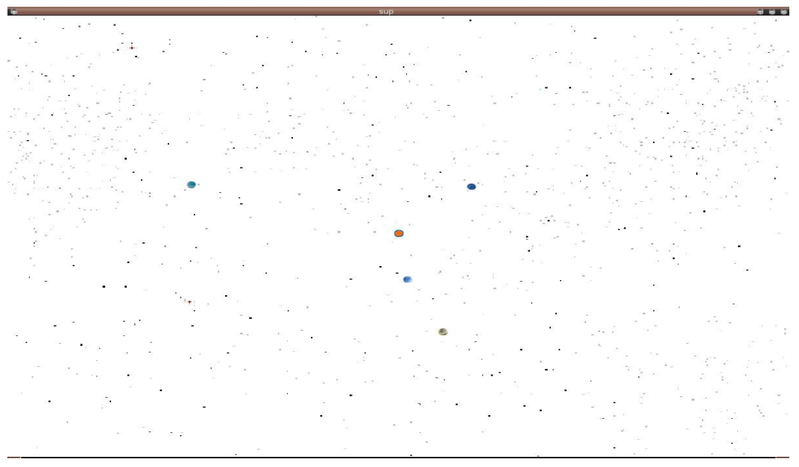
2-D model of the Solar System using physics-based geometries in Java

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 these models are limited in that they aren't physically accurate and fail to account for possible unexpected changes in the solar system. A more advanced model that would solve these issues, however, 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 in its own right), but the ability to add additional solar bodies to the system and view the Solar System's reaction to those bodies' presence is valuable for experimentation. This project, therefore, seeks to create a solar system simulation that graphically illustrates a basic Keplerian model of the inner solar system using Newtonian gravitation calculations.

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 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.

On the other hand, 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 be made customizable - future users of the simulation could, with relatively ease, place a solar body at a location, assign that solar body a mass, velocity, and direction, and view what happens to the Solar System both qualitatively on screen and quantitatively in the program's output position data. The structure and frequency of this output data could be further customized by the user.

The most important goal of the project, however, is to simply create a simulation that simulates the current, real world solar system with relative accuracy while making actual gravitational interaction calculations and not relying on an "on wire" model to move the planets. Essentially, if the simulatation is unable to simulate the real world solar system with any accuracy, then it is worthless as a model. Customization of a worthless model is probably equally worthless. In short, an accurate simulation is important.

The many times afforementioned gravitational interactions are simulated while the program runs by iteratively calculating each planet's changing acceleration according to Newton's law of gravitation. The simulation ceases to run after approximately one year and planetary positions are compared with real world data from NASA's JPL HORIZONS online solar system data and ephemeris computation service. As of now, the simulation serves as a basic visualization tool for motion due to gravitational interaction, but certainly could be customized in the future to view the Solar System's reaction to the presence of varying intrusive bodies

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 Real World Data Comparisons

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

As it turned out, the solar system simulation did a fair job of predicting and displaying the positions of the planets. When compared to real world data gathered from NASA after one year, only Mercury and Venus have non-negligible percent error; the other planets' predicted positions are very close to their actual positions. This is reasonable considering that Mercury and Venus are moving at a faster velocity than the planets' farther away from the sun. This ultimately entails more estimation in the planet's predicted path; while Neptune may move only slightly between acceleration recalculations based on the other planets' positions from Newton's law of gravitation, Mercury may move a much larger distance between those calculations. Additionally, a certain amount of rounding error is to be expected in all calculations due to constant conversions between on-screen "pixel coordinates" and real world "solar system" coordinates (that is, the actual positions of the planets relative to the sun, measured in km).

Ultimately both sources of error compounds each other and there is little doubt that the simulation becomes less accurate with time. Nonetheless, the solar system simulation may be a valuable tool for the visualization of planetary motion in real time and a fair predictor of planetary positions for small increments of time. Additionally, the fact that the simulation makes use of actual gravitation calculations lends the program flexibility in that it could be modified to visualize the reaction of the solar system to intrusive bodies. If it were possible to eliminate error in the algorithm used for estimating planetary accelerations by the current version of the simulator, then the latter possible would indeed be very va

Planet	xPos (km)	yPos (km)	nasaX(km)	nasaY (km)
Mercury	-4.02E+007	-5.54E+007	-4.88E+007	-4.56E+007
Venus	2.29E+007	1.05E+008	3.32E+007	1.03E+008
Earth	-1.48E+008	2.12E+007	-1.47E+008	1.90E+007
Mars	5.35E+007	-2.09E+008	5.42E+007	-2.06E+008
Jupiter	-2.94E+008	-7.43E+008	-2.98E+008	-7.42E+008
Saturn	-1.09E+009	8.32E+008	-1.09E+009	8.34E+008
Uranus	2.90E+009	-7.93E+008	2.90E+009	-7.94E+008
Neptune	3.42E+009	-2.91E+009	3.42E+009	-2.91E+009

Planet	X Percent Error	Y Percent Error
Mercury	17.62	21.41
Venus	30.87	2.71
Earth	0.17	11.66
Mars	1.18	1.07
Jupiter	1.09	0.13
Saturn	0.08	0.23
Uranus	0.01	0.22
Neptune	0.02	0.07