TJHSST Senior Research Project
Creating 2-D and 3-D models of the Solar System using physics-based geometries in Java
2008-2009

Brian Tubergen

March 31, 2009

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 arent 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 Systems 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.

Keywords: N-body Problem, Kepler's 3rd Law

1 Introduction - Problem Statement and Purpose

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. NASAs 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 relatively 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 planets changing acceleration according to Newtons law of gravitation. The simulation ceases to run after approximately one year and planetary positions are compared with real world data from NASAs 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 previously mentioned 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.

Perleys *Solar System Motion Simulator* didnt originally implement real physics, which is an improvement Id 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.

The *Orrery Solar System Simulator* by K. McClary is a more detailed (from a physical perspective) model. However, its been decided that for now my project wont 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

$$F = \frac{G * m1 * m2}{r^2}$$

One can then solve for the acceleration of a planet due to gravity, and thus the motion can be simulated. My project will likely include more advanced graphics than McClary's.

A Scale Model of the Solar System makes use of comparisons to planetary data provided by the U.S. Navy in order to verify the accuracy of the model. Planets also trace out their motion; the path that they have followed is marked, leaving a trail behind the planet as it moves. The latter is something that I hope to implement, but isnt necessary; the former, the comparisons to real world data, would be important to implement in order to verify my simulations accuracy. I plan on doing similar comparisons with data from NASAs Horizons system.

Symmetric Multistep Methods for the N-Body Problem describes the multitude of ways to approach the aforementioned N-Body problem, many of which are very advanced and probably not necessary for my simple Keple-

rian model. Several of these methods are interesting, however, because they allow for time reversibility; the simulation would essentially be able to step back in time. Although I dont plan on implementing this for lack of time, it could be used in future versions of my simulation.

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 simulations output to actual results.

3.1 Display Class

The display class creates the planetary objects and handles the programs 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 Position Computations

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 its net acceleration based on the equation

$$a = \frac{G * m}{r^2}$$

. It then updates its velocity and position accordingly, and the display class updates the graphical display accordingly after all planets have finished these calculations.

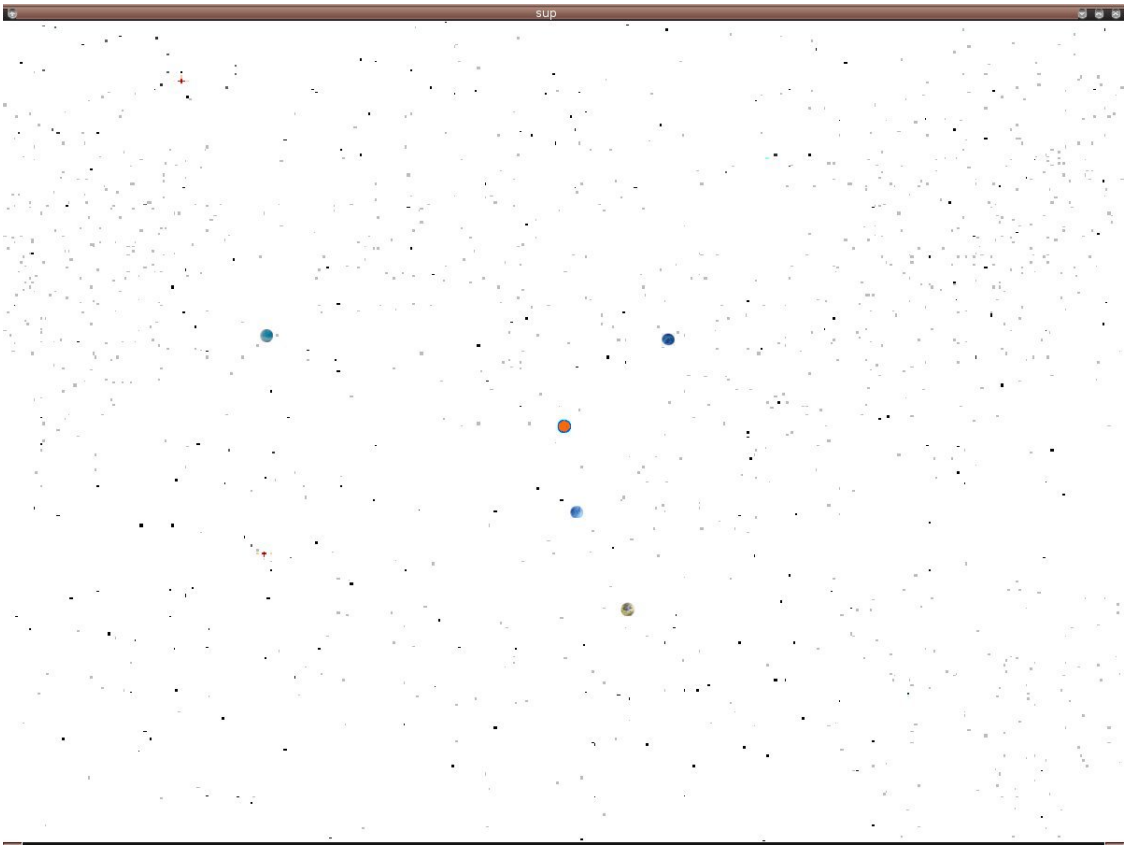

Figure 1: Screenshot of sample run with simulation "zoomed" in on inner solar system (first four planets).

3.3 Real World Data Comparisons

Data for real world planetary systems has been gathered from NASAs 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 NASAs 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 valuable.

```
*******************************************************************************
Ephemeris / MAIL_REQUEST Tue Nov 18 07:22:34 2008  Pasadena, USA / Horizons
*******************************************************************************
Target body name: Mercury (199)                   {source: DE405}
Center body name: Sun (10)                        {source: DE405}
Center-site name: BODY CENTER
*******************************************************************************
Start time      : A.D. 2006-Mar-13 17:30:58.0000 CT
Stop  time      : A.D. 2006-Mar-13 17:30:59.0000 CT
Step-size       : 60 minutes
*******************************************************************************
Center geodetic : .000000000,.000000000,.00000000 {E-lon(deg),Lat(deg),Alt(km)}
Center cylindric: .000000000,.000000000,.00000000 {E-lon(deg),Dxy(km),Dz(km)}
Center radii    : 696000.0 x 696000.0 x 696000.0 k{Equator, meridian, pole}
Output units    : KM-S
Output format   : 03
Reference frame : ICRF/J2000.0
Output type     : GEOMETRIC cartesian states
Coordinate systm: Ecliptic and Mean Equinox of Reference Epoch
*******************************************************************************
JDCT
   X     Y     Z
   VX    VY    VZ
   LT    RG    RR
*******************************************************************************
$$SOE
2453808.229837963 = A.D. 2006-Mar-13 17:30:58.0000 (CT)
  -5.730179611929864E+07  1.963895677831354E+06  5.419068833578369E+06
  -1.179887453107446E+01 -4.658790336366766E+01 -2.723423174395 28E+00
   1.921027750084662E+02  5.759096310840907E+07  9.894693725010404E+00
$$EOE
*******************************************************************************
```

Figure 2: Sample data acquired from NASA. The x, y, z, vx, vy, and vz data at the bottom between SOE and EOE are the most important for this project.

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

Figure 3: Output position data from my Solar System simulation compared to actual planetary position data gathered by NASA.

References

[1] NASA, "JPL Solar System Simulator", http://space.jpl.nasa.gov/ blmt/, 2009.

[2] D. Perley, "Solar System Motion Simulator", http://astro.berkeley.edu/ dperley/programs/ssms.htmlb̃lmt/, 2005.

[3] B. Jenkins, "Multistep Methods for the N-Body Problem", http://burtleburtle.net/bob/math/multistep.htmlb̃lmt/, 2009.

[4] B. Jenkins, "A Scale Model of the Solar System", http://burtleburtle.net/bob/physics/solar.htmlb̃lmt/, 2009.

[5] K. McClary, "Orrery : Solar System Simulator", http://www.cuug.ab.ca/kmcclary/ORRERY/index.htmlb̃lmt/, 2009.

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

Figure 4: Percent error calculations for X and Y planetary coordinates. Percent error was calculated using the formula: — (Actual-experimental) / Actual — *100, where the experimental data comes from my Solar System Simulation and the actual data comes from NASA. One can see that there is more error involved with the position calculations for Mercury and Venus than the planets farther from the sun.