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

February 26, 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 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 arent 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 Systems reaction to their presence is valuable for experimentation.

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.

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.

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 Keplerian 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 onscreen.

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.

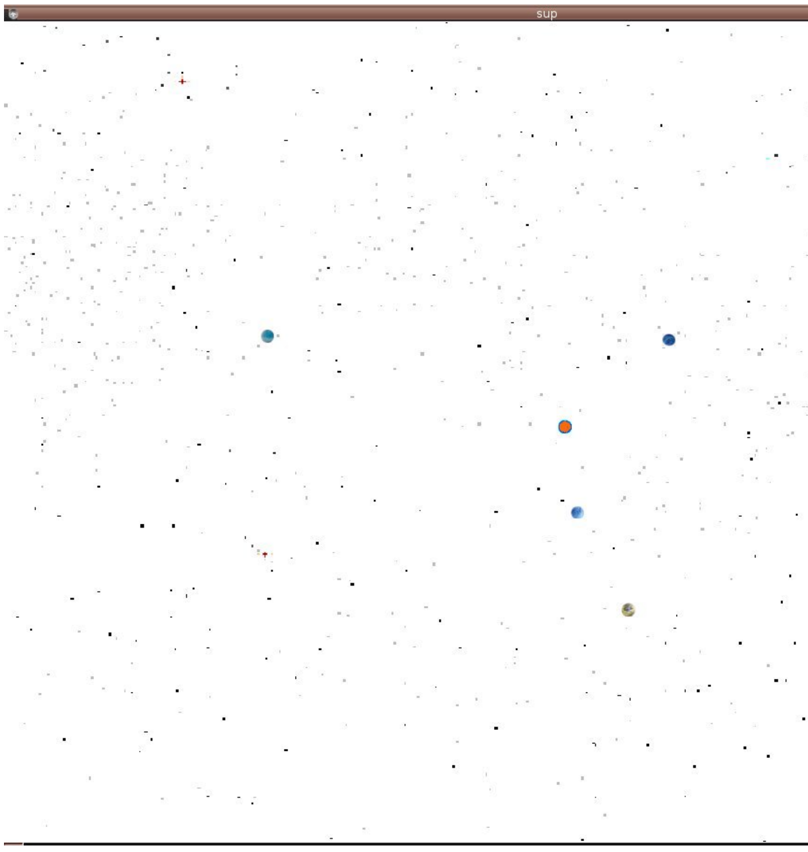
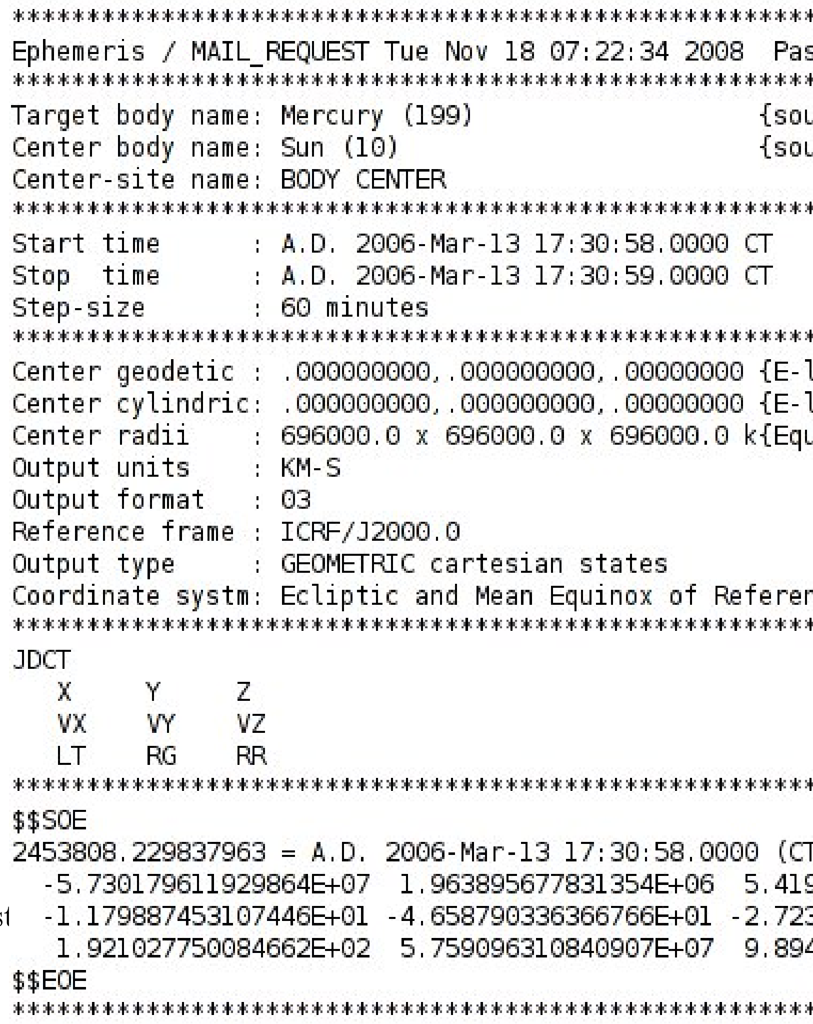


Figure 1: Screenshot of sample run with simulation "zoomed" in on inner solar system (first four planets).

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.



```
*************************************************
Ephemeris / MAIL_REQUEST Tue Nov 18 07:22:34 2008  Pas
*************************************************
Target body name: Mercury (199)                  {sou
Center body name: Sun (10)                       {sou
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-l
Center cylindric: .000000000,.000000000,.00000000 {E-l
Center radii    : 696000.0 x 696000.0 x 696000.0 k{Equ
Output units    : KM-S
Output format   : 03
Reference frame : ICRF/J2000.0
Output type     : GEOMETRIC cartesian states
Coordinate systm: Ecliptic and Mean Equinox of Referen
*************************************************
JDCT
   X     Y     Z
   VX    VY    VZ
   LT    RG    RR
*************************************************
$$SOE
2453808.229837963 = A.D. 2006-Mar-13 17:30:58.0000 (CT
  -5.730179611929864E+07  1.963895677831354E+06  5.419
  -1.179887453107446E+01 -4.658790336366766E+01 -2.723
   1.921027750084662E+02  5.759096310840907E+07  9.894
$$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.

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.html̃blmt/, 2005.

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

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

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