Sun-Earth-Moon System

I. Introduction

This project shows the simplified version of a Sun-Earth-Moon model using the Leapfrog Integrator. I chose this project as it intrigued me how Earth orbits around the sun. I was able to make the program run dynamically, so the user only has to send a file which has the planet mass, x-value, y-value, vx value and vy value initialized. Also, I made it easier for the user to choose the default files which already have the values defined. One thing I would like to add for next time is more planets that the user can define to expand the Sun-Earth-Moon system.

II. Method

Leapfrog Integrator

I used the Leapfrog Integrator to implement the project. The equations I used to update the coordinates are:

$$x_i = x_{i-1} + h * vx_i$$

This equation changed the x-coordinates depending on the planets (h: time step, i: number of time step)

$$y_i = y_{i-1} + h * vy_i$$

This equation changed the x-coordinates depending on the planets (h: time step, i: number of time step)

For updating the velocity of each planet I used the equations:

$$vx = vx + \frac{h}{2} * dvxdh(x, y)$$

This equation changed the velocity of x depending on the distance between planets (h: time step)

$$vy = vy + \frac{h}{2} * dvydh(x, y)$$

This equation changed the velocity of y depending on the distance between planets (h: time step)

Verification: Energy

To verify this model is an accurate simulation of the Sun-Earth-Moon system, the program must conserve the total energy of the system. By evaluating the kinetic and potential energies of all planets I was able to get the total energy of the system.

$$KE_i = 0.5 * m_i * v_i^2$$

$$PE_{ij} = -G\frac{m_i m_j}{r_{ij}},\ i \neq j$$

The kinetic energy is calculated by multiplying the mass by velocity squared and add it together. For the potential energy, it depends on the distance between the different planets ($r_{ij},\ i \neq j$) and the gravity constant.

Verification: Angular Momentum

To verify this model is an accurate simulation of the Sun-Earth-Moon system, the program must also conserve the total angular momentum of the system. Since we are assuming the model is in the 2D space, I used the origin (0,0) to calculate the radius.

$$L = L + m * ((x - origin) * vy - (y - origin) * vx)$$

The angular momentum is calculated for each planet in the model and add it together to give the total angular momentum of the system.

III. Test

To test my program I made a simple demo to see if Earth orbits around the Sun. I had made a class for Planets which initialized the values needed. Then, I made a class for Solar Systems which was making the Planets with the values given in the input file. By making planets dynamically through the use of object orientation it was quite easy to implement the functions. First I checked the energy by running it in a for loop and seeing if the energy was conserved with the Leapfrog integrator applied.

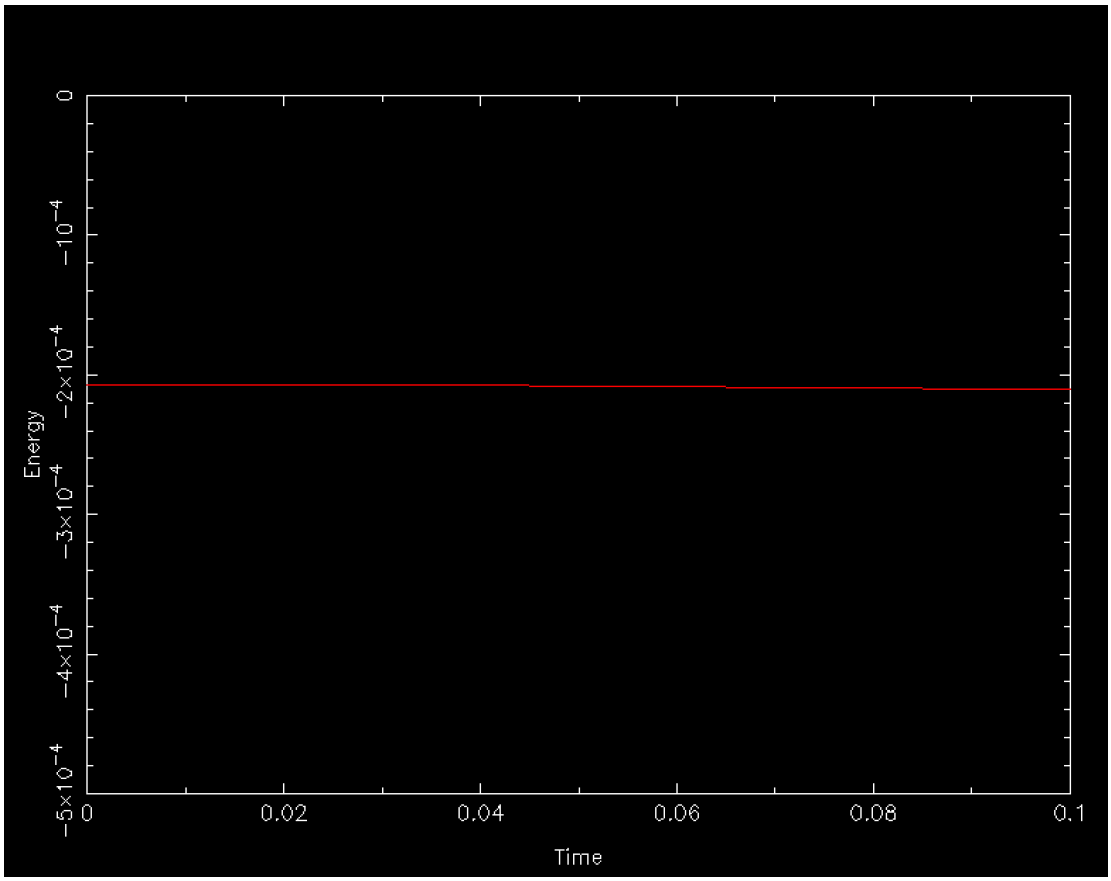

Figure 1: Total Energy vs Time for h = 0.01

As the updates to the coordinates and velocity happened the energy was conserved within the round-off error as it has to calculate the distance between the planets which usually makes it lose little precision.

Next, I checked if the angular momentum is also conserved when the Leapfrog Integrator is applied.

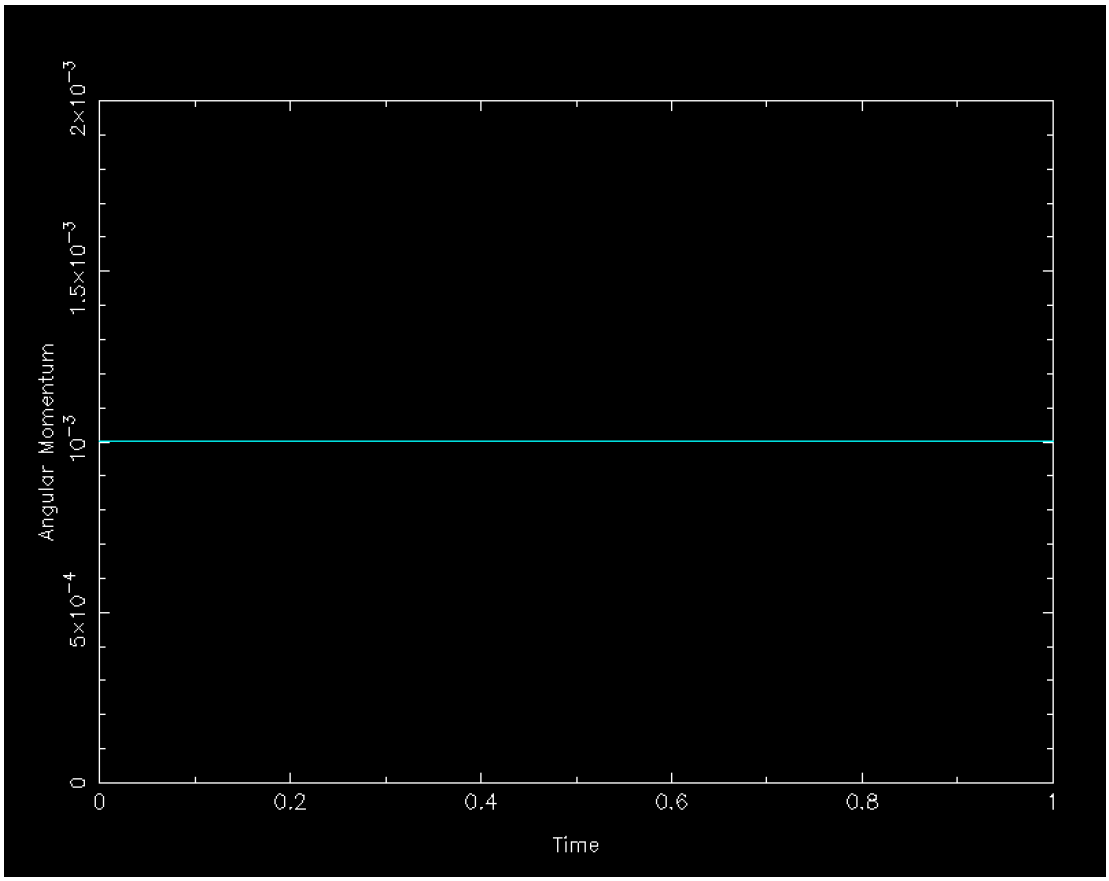

Figure 2: Total Angular Momentum vs Time for h = 0.01

As the updates to the coordinates and velocity happened the angular momentum was also conserved but without any round-off error.

Lastly, I made float arrays that stored the updated coordinates of the Earth and I used the float arrays to plot the orbits. For the Sun since it stayed in one position I just made float coordinates and plotted them as a point.

The inputs for Sun were:

Mass: 1.00

x value: 0.00 y value: 0.00

vx value: 0.00 vy value: 0.00

The inputs for Earth were:

Mass: 0.001

x value: 1.00 y value: 0.00

vx value: 0.00 vy value: 1.00

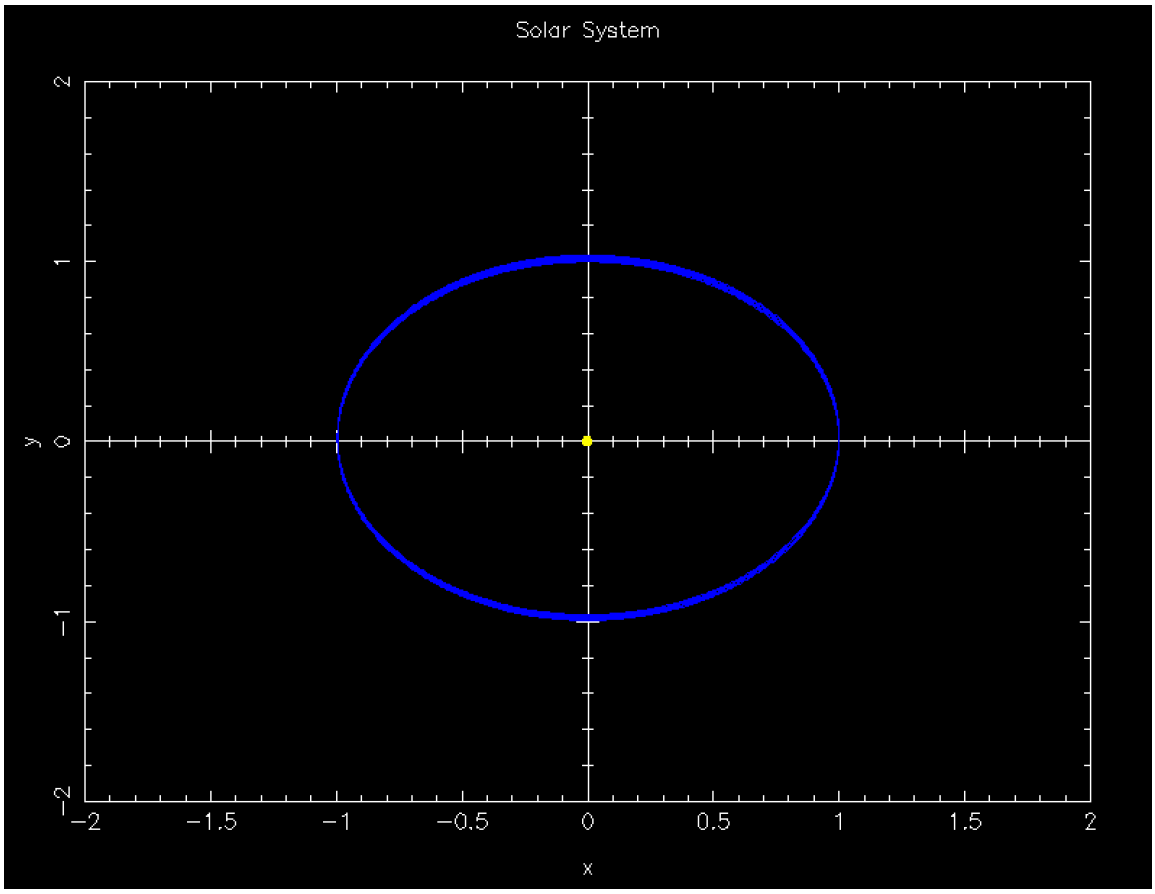

Figure 3: Sun-Earth Model (Sun: Yellow, Earth: Blue)

As the coordinates and velocity are updating the Earth is going around the Sun in an orbit. Therefore proving the demo model works.

IV. Results

For the extension of the project, I added the Moon to the model. Since I defined the planets dynamically, it was easy to add another planet as all I had to do was initialize the mass, x, y, vx and vy values.

The inputs for Moon were:

Mass: 0.000

x value: 1.01 y value: 0.00

vx value: 0.00 vy value: 1.31622776601

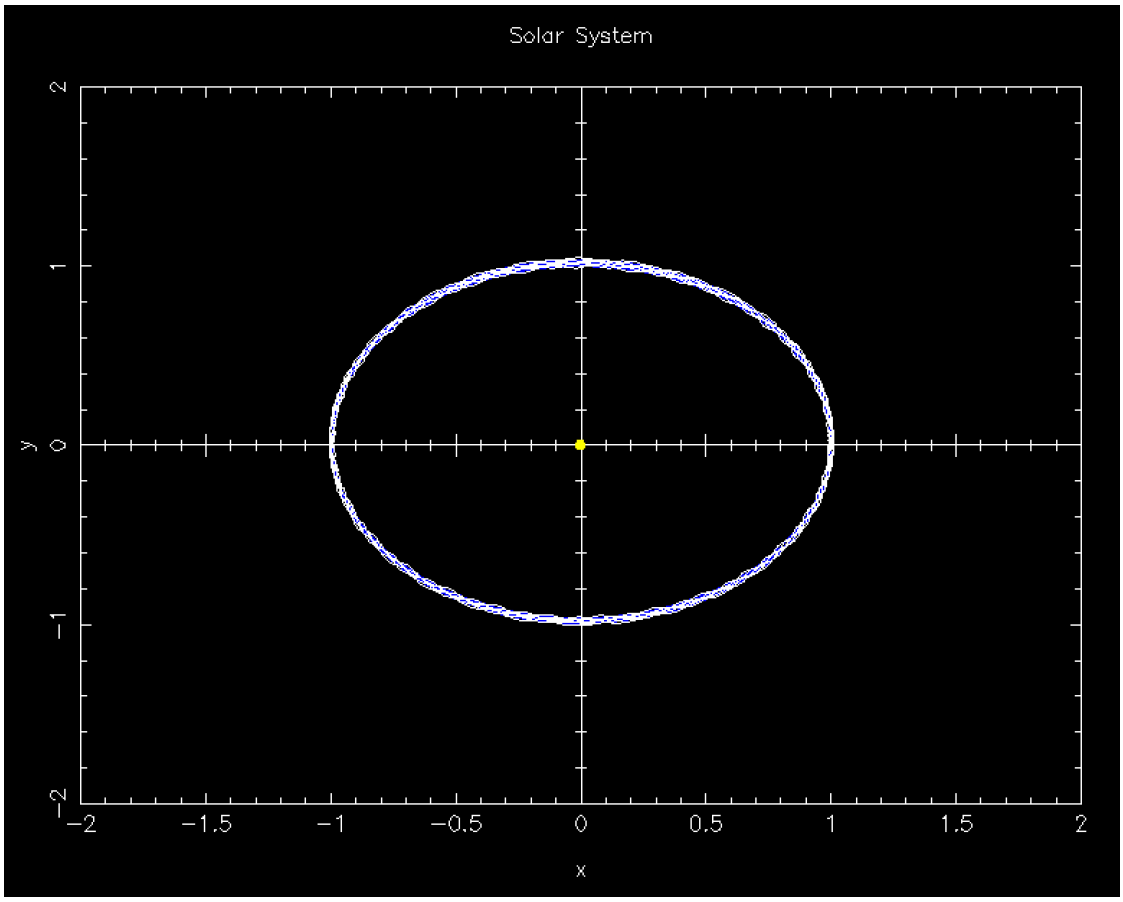

Figure 4: Sun-Earth-Moon Model (Sun: Yellow, Earth: Blue, Moon: White)

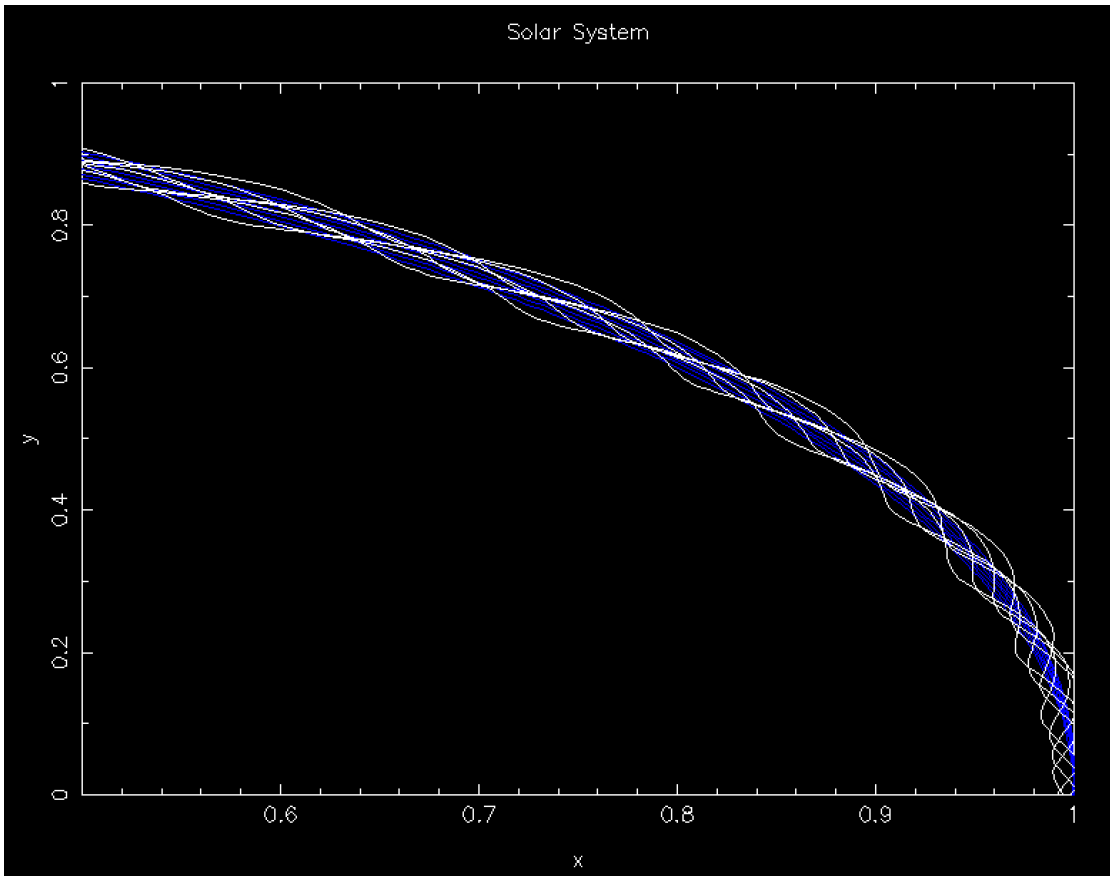

Figure 5: Close-Up of Sun-Earth-Moon Model (Sun: Yellow, Earth: Blue, Moon: White)

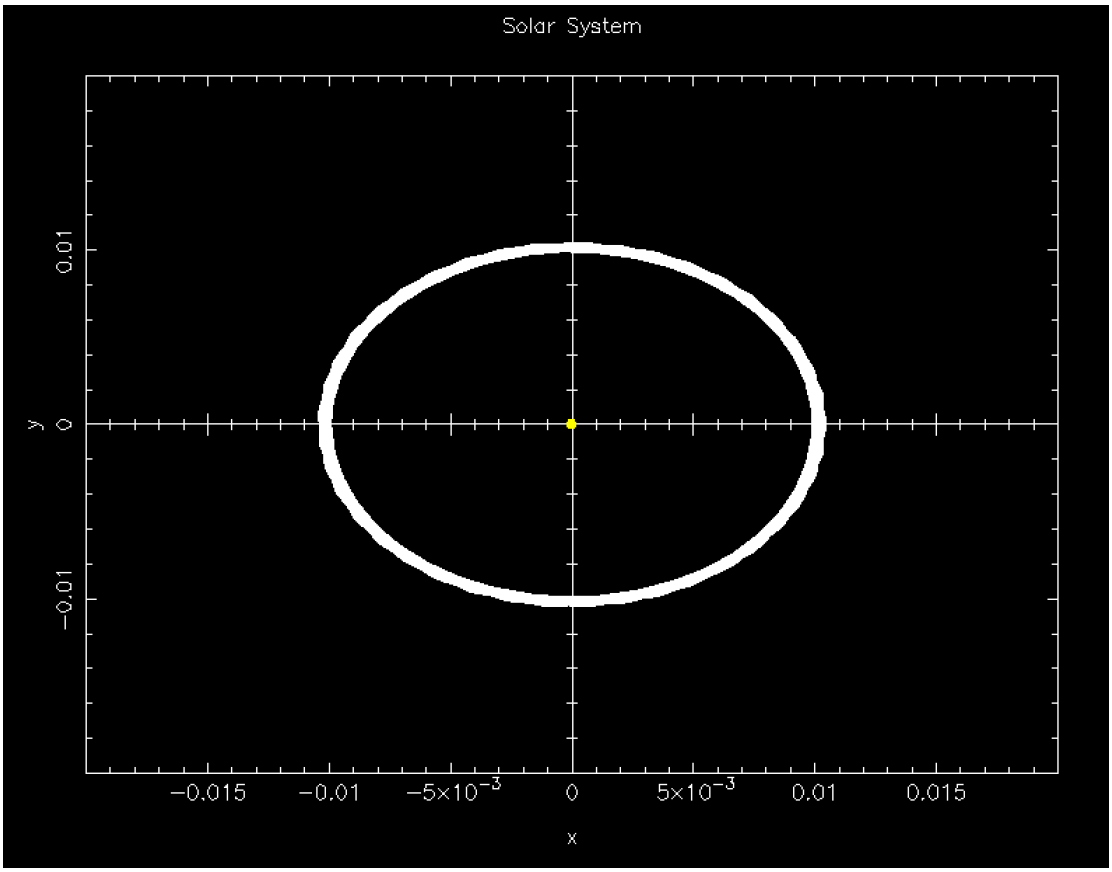

Figure 6: Close-Up of the Moon (Sun: Yellow, Earth: Blue, Moon: White)

Even though it is hard to see in Figure 4 that the Moon is orbiting the Earth when zoomed-in for Figure 5 and 6, we can clearly see it orbiting the Earth while going around the Sun.

Another extension I added to the model is that the user can add their own data file in which they can edit the values for Sun, Earth and Moon and add another planet as well to see what changes it makes to the Sun-Earth-Model. The way to format the file is

1. Number of Planets
2. Sun Mass Sun x-value Sun y-value Sun vx-value Sun vy-value
3. Earth Mass Earth x-value Earth y-value Earth vx-value Earth vy-value
4. Moon Mass Moon x-value Moon y-value Moon vx-value Moon vy-value
5. Planet Mass Planet x-value Planet y-value Planet vx-value Planet vy-value

The 4th and 5th lines are optional. So let's take an example,

1. 4
2. 1.00 0.00 0.00 0.00 0.00

3. 0.001 1.00 0.00 0.00 1.00
4. 0.000 1.01 0.00 0.00 1.31622776601
5. 0.000 0.80 0.00 0.00 1.05

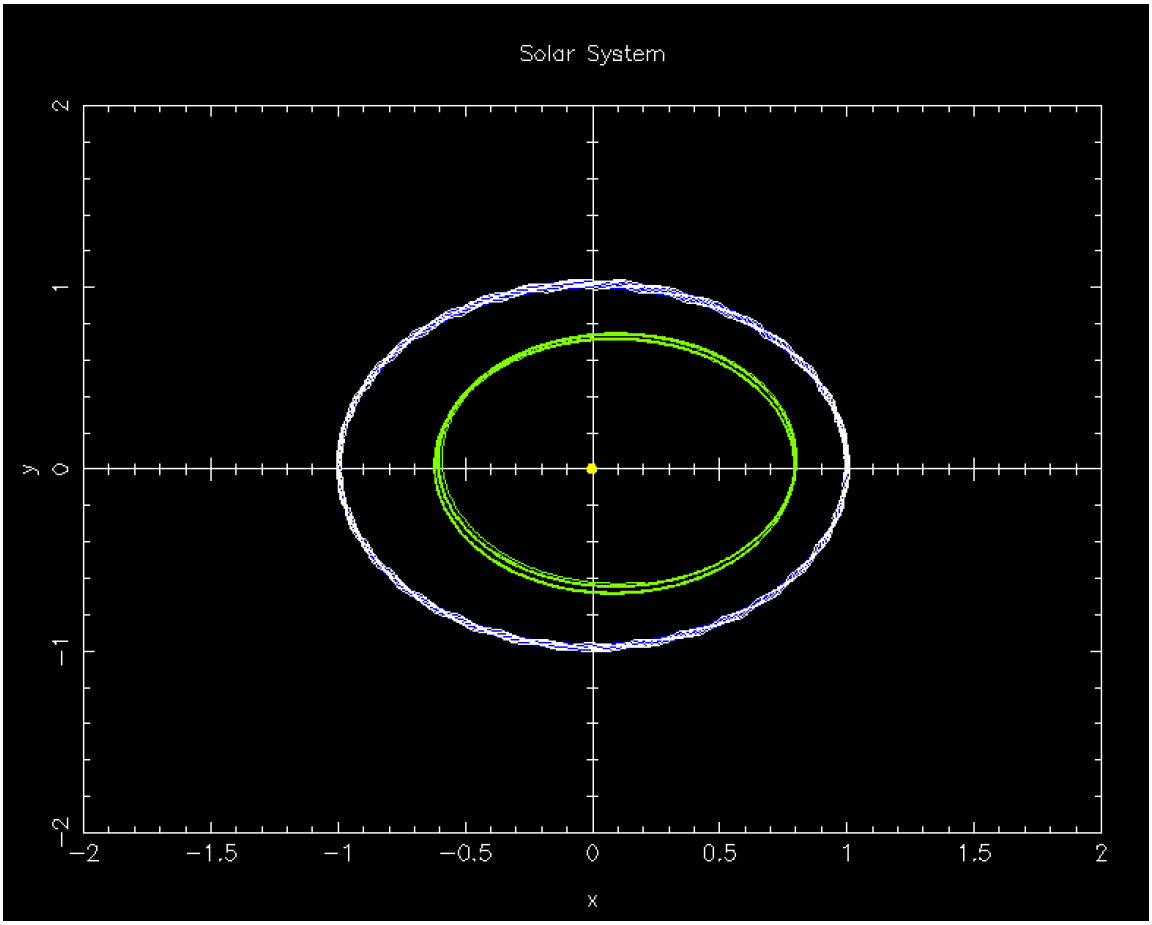

Figure 7: Solar System Model (Sun: Yellow, Earth: Blue, Moon: White, Planet: Green)

As we can see this planet is much close to the Sun and orbits much quicker than Earth. This is a good extension as the user can see when the asteroid will hit or how other planets will affect the model.

Lastly, I added the calculation of counting the years. As we know one year is completed when the Earth has gone all the way around the Sun once. So a way to count the years is to count the number of times the Earth crosses the x-axis. As we know that the orbits do not stay the same, so we can not just say when the initial x value is equal to any x value, it is a year. Rather we just compare when the x value changes from negative to positive because the initial x value of Earth starts in quadrant 1 so it should also end in quadrant 1. So for example, if we take 4000 iterations over the leapfrog integrator, there will be 6 years.

Validation: Comparison of the model vs. the real outcome

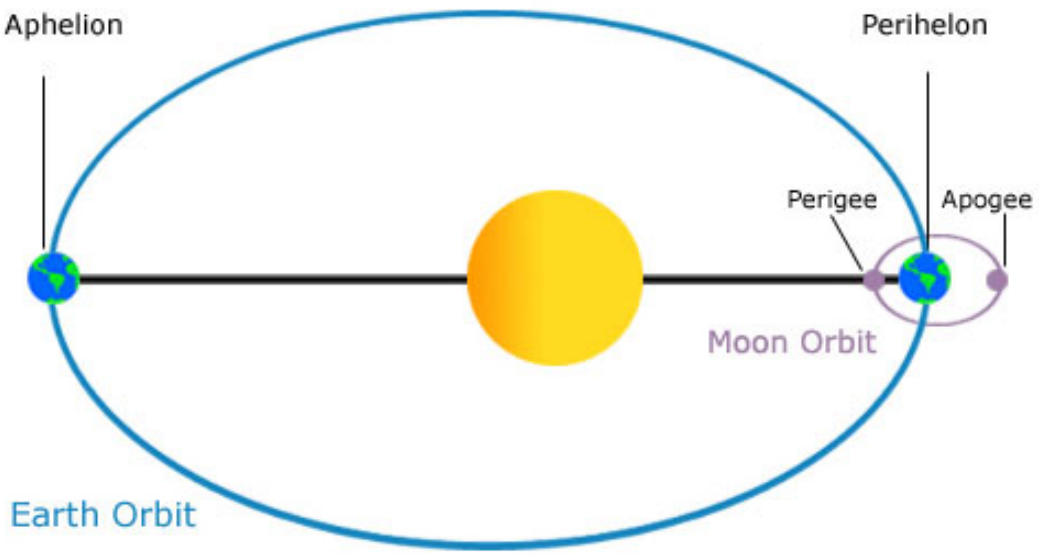

Figure 8: Realistic Model of the Solar System

As we can see the Sun-Earth-Moon model has a really close resemblance to the real model of the Solar System. This is because I used the formulas that are defined to calculate the velocity. I used the formula $\frac{GM}{r} = v^2$ to calculate the velocity of each planets.

First I set the Sun's mass as 1.00 and all the other values as 0 so I can calculate the velocity of Earth without altering the vx value. I used the gravity constant which in this case is 1, the Sun's mass and radius, which is just the x value of Earth that I set to 1.00 $\sqrt{\frac{1(1)}{1}} = 1$. So the velocity of Earth is 1.00 which is the vy value. Then, I set the value of Earth's mass to 0.001, the x value was already 1.00 and the rest of the values I set as 0.

Lastly, to calculate the Moon's velocity I used the Earth's mass, gravity constant and radius of the Moon which is just the x value of the Earth subtracted by the Moon's x value. $\sqrt{\frac{1(0.001)}{0.01}} = 0.31623$. Since the Earth's velocity is 1.00 and the Moon goes around the Earth, we need to add Earth's velocity to the get the Moon's velocity. Therefore, the velocity of the Moon is 1.31622776601.

Even though the values have been scaled down for this model it still holds all the properties as it follows the same formula as the real model.

V. Conclusion

Through this project, I learned how gravity and velocity affect the orbits of the Earth around the Sun. Also, I learned that we could apply the realistic model of the Solar System for various applications such as finding out how long will an asteroid take to hit the Earth, when will Sunar or Lunar eclipse will happen, what time should we launch the spaceship so it takes less time to get to the international space station and etc. There are various ways to implement a realistic model of the Solar System like using Runge-Kutta, however, the leapfrog integrator has the best approximation for orbits. In the end, if I had extra time I would add all the planets in the Solar System and plot the orbits in a 3D graph using gnuplot. Overall, I believe this project was a success as I was able to implement a realistic model of Sun-Moon-Earth.

VI. References

http://www.physics.drexel.edu/~steve/Courses/Comp_Phys/Integrators/leapfrog/