Planet Collider 1.0: A Computer Program for Plotting the Trajectory of a Gravitationally Bound Body around a Stationary Center of Mass

Kyle Drury

January 16, 2023

Abstract

The orbital differential equations are powerful tools that can be used to accurately predict the motion of systems of interacting celestial bodies around a center of mass. The simplest-case system, which is what v1.0 will be modeling, is one where there are two bodies, and where one of the objects dominates the system in terms of its gravitational influence (ex. the Earth and the Moon).

1 Important Definitions and Parameters

1.1 The Equations

Planet Collider is based on the following two second-order differential equations that describe functions r and θ of t:

$$\frac{d^2r}{dt} - r(\frac{d\theta}{dt})^2 = -\frac{GM}{r^2} \qquad\qquad r\frac{d^2r}{d\theta} + 2\frac{dr}{dt}\frac{d\theta}{dt} = 0 \tag{1}$$

If we let $\frac{d\theta}{dt} = u$ and $\frac{dr}{dt} = v$, then we can rewrite the above system of equations as the one seen below:

$$\frac{dv}{dt} = ru^2 - \frac{GM}{r^2} \qquad\qquad \frac{du}{dt} = -\frac{2uv}{r} \tag{2}$$

We say that, in general, $y_{i+1} = y_i + dt\frac{dy}{dt}(y_i, t_i, parameters)$, so we will devise a suitable algorithm to solve for θ as a function of t, and then convert the solution into $y = f(x)$ form.

1.2 Initial Conditions

The program is working to solve for angular velocity, radial velocity, angular position, and radial position all as a function of time. We can solve all of these by using the Euler method in an iterative manner, but we need to determine all of the initial conditions.

$$\theta_0 = \theta_{initial} \qquad r_0 = r_{initial} \qquad v_0 = \frac{dr}{dt}_0 = v_{parallel} = vcos\theta_0 \qquad u_0 = \frac{d\theta}{dt}_0 = \omega_0 = \frac{vsin\theta_0}{r_0} \tag{3}$$

In the above equations, θ_0 is the angle between the velocity vector $\vec{v}$ and the radial displacement vector $\vec{r}$, which can be easily calculated (see the depiction in figure 1, pg 2).

1.3 The Algorithm

See figure 2 (pg 2) to see the code written to solve for θ and r as a function of t, then x as a function of y.

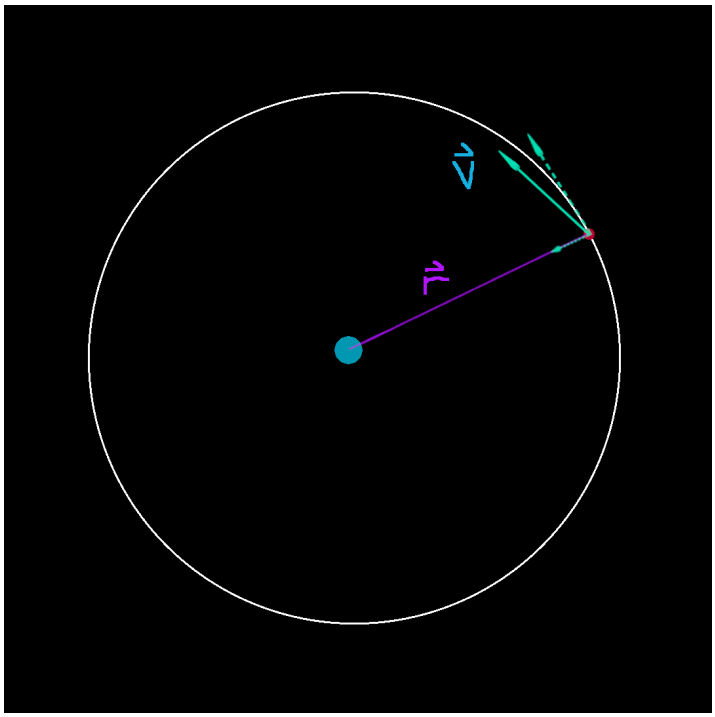

Figure 1:

```
// Solve equations
for (i=0; i < (n-1); i++) {
    dvdt_array[i] = dvdt( r: r_array[i],  u: u_array[i], G, M);
    dudt_array[i] = dudt( u: u_array[i],  v: v_array[i],  r: r_array[i]);
    v_array[i+1] = v_array[i] + dt*dvdt_array[i];
    u_array[i+1] = u_array[i] + dt*dudt_array[i];
    r_array[i+1] = r_array[i] + dt*v_array[i+1];
    theta_array[i+1] = theta_array[i] + dt*u_array[i+1];
    x_array[i] = r_array[i]*cos( X: theta_array[i]);
    y_array[i] = r_array[i]*sin( X: theta_array[i]);
    t_array[i+1] = t_array[i] + dt;
}
```

Figure 2:

2 Testing the Program

2.1 A Stable Orbit

We shall test AsteroidCollider to model a system that we are especially familiar with; that of Earth and the Moon. The top orbital velocity of the moon at the perigee ($r = 357,000$km) is roughly 3960km/h. As such, our initial conditions may be $\vec{r} = (x, y) = (357000, 0)$, $\theta = 0$, and $\vec{v} = (v_x, v_y) = (0, 3960)$ (see figure 3).

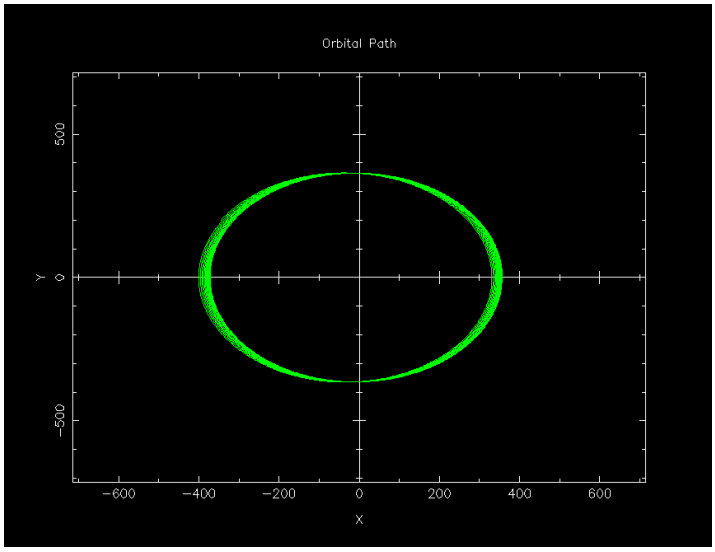

Figure 3:

Excellent. The orbit seems more or less stable, although it is deviating slightly with each full revolution. Let's try experimenting with the initial parameters to see what happens.

2.2 Unstable Orbits

Let the actual perigee orbital velocity of the moon be $= v_{orbit}$. Figure 4 depicts the orbital path of the moon starting at the same location but being given an initial speed of $0.75v_{orbit}$, $0.63v_{orbit}$, $0.58v_{orbit}$, $0.48v_{orbit}$.

So, if you were wondering, the point at which we *might* have to worry about the Moon crashing into the Earth if it suddenly decelerated to half of its regular velocity.

What if Moon started at the same location, but the velocity vector was titled 45 degrees to the left? Figure 5 shows this.

2.3 Arbitrary Orbits

The program is also capable of mapping the trajectory of an orbiting object that starts at an arbitrary location with a chosen speed. Below are the plots of some such trajectories (figure 6) associated with $\vec{v} = (-1300, 2500), (-2000, 4000)$, and $(1300, -2500)$, respectively.

3 Conclusions

The first version of *PlanetCollider* works well to model already-known, stable orbits. Finding stable orbits by hand is tedious and difficult, but it is easy to tell from the provided plots if an orbiting body will fall into the gravitational well it is bound to, or if it will be flung away. And, of course, the program does perform the task it's named after; it is able to tell whether an oncoming asteroid would be a threat to Earth.

It seems intuitively peculiar that given less kinetic energy, the moon would be able to escape the Earth's gravitational pull. To understand what's happening here on a deeper level, the next step

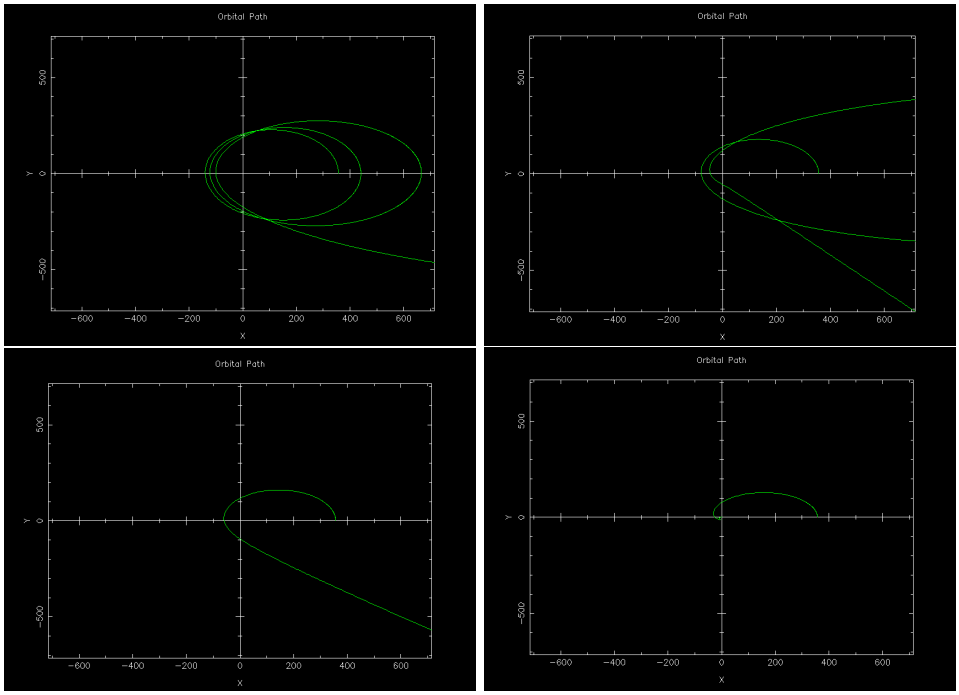

Figure 4:

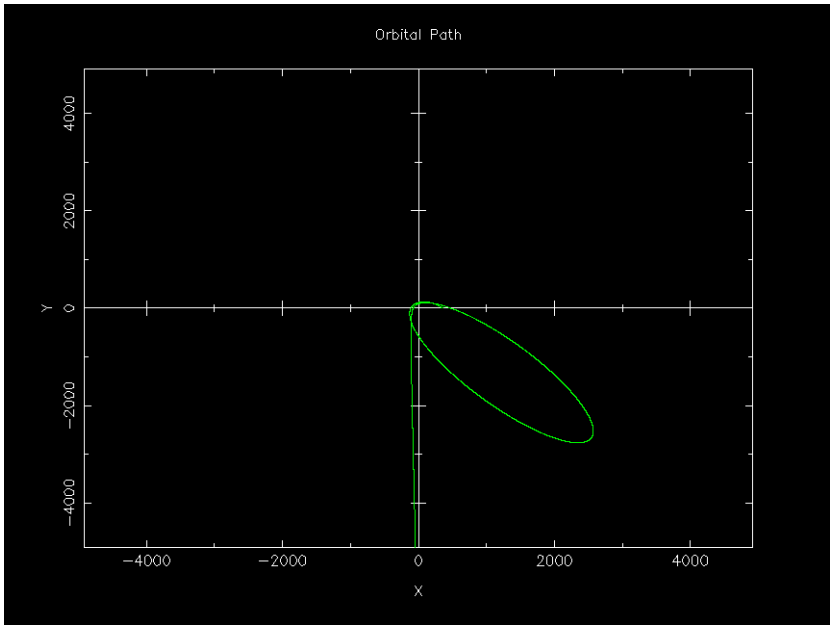

Figure 5:

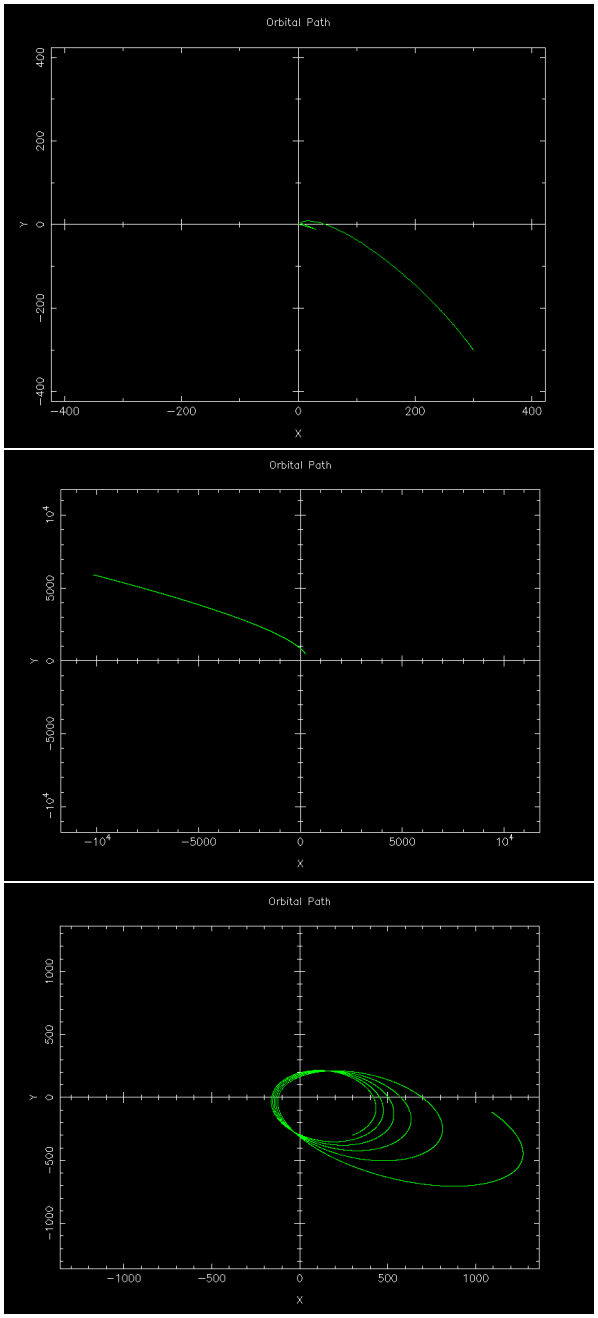
Orbital Path
400
200
0
−200
−400
Y
−400
−200
0
200
400
X
Orbital Path
10^4
5000
0
−5000
-10^4
Y
-10^4
−5000
0
5000
10^4
X
Orbital Path
1000
500
0
−500
−1000
Y
−1000
−500
0
500
1000
X

Figure 6:

would be to calculate the gravitational potential energy along with the kinetic energy of the system as a function of time to see if mechanical energy is remaining conserved.

Another next step that I am going to take to more accurately describe binary planetary systems is incorporating a properly calculated center of mass, where neither object is absolutely fixed. The fact that this wasn't already the case may also account for some shortcomings of the program.