Physics 307 Homework 5
Due Thursday, 18 October, at 5 PM

In this homework assignment, you will simulate the Earth going around the Sun, and then generalize your work to other orbits. There is a hints section at the end.

To review the discussion in class: For an orbit in the x-y plane, you have four dynamical variables x, y, v_x, and v_y. The four differential equations that govern them are

$$\dot{x} = v_x \tag{1}$$
$$\dot{y} = v_y \tag{2}$$
$$\dot{v}_x = -\frac{GMx}{r^3} \tag{3}$$
$$\dot{v}_y = -\frac{GMy}{r^3} \tag{4}$$

where M is the mass of the Sun.

1. First we will pretend that the Earth is in an exactly circular orbit; this is approximately the case. If we measure distance in astronomical units and time in years,
   - what initial conditions for the Earth's position and velocity vectors will give a circular orbit with the correct radius, and
   - what is the numerical value of GM?
2. Write code that simulates the Earth's orbit around the Sun using the leapfrog algorithm. (If you like, you can code Euler-Cromer-Aspel first.) Animate your simulation.
3. Now, let's check to see if the simulation is accurate. Since you put in initial conditions that are explicitly designed to simulate a planet orbiting in a circular orbit with a period of one year, you can check a couple of things:
   (a) Does your planet stay at a fixed distance from the origin?
   (b) Does your planet complete an orbit in one year?
   (c) Does your planet's orbit conserve energy?

   Remember, you will see *some* deviation from these ideals because of "finite timestep error". That's okay, as long as those errors get small as $dt \to 0$.

   I'll let you test these things in any way you want.

The total energy per unit mass ("specific orbital energy") is given by the sum of the potential and kinetic energy

$$E_{\text{spec}} = \frac{1}{2}(v_x^2 + v_y^2) - \frac{GM}{r}. \tag{5}$$

Modify your code to display the energy on the screen as the animation runs. You can do this with the `T` animation command.

4. Modify your initial conditions to simulate a non-circular orbit, changing the initial velocity and position of the Earth. (Note that if the total energy becomes positive, the Earth will escape the Sun's gravity.) What shape do these orbits take? At what part of the planet's orbit does it move faster? Spend at least half an hour playing around with different initial conditions and seeing what happens, and write about that in your report. In particular, how does the planet move if the total energy is very negative? Small and negative? Small and positive?

5. With initial conditions appropriate for an elliptical orbit, modify Newton's law of gravity a bit. Specifically, change the force law to $F_g = \frac{GMm}{r^{2\pm\epsilon}}$, where ϵ is some small value. Simulate your orbit again, ensuring that the planet leaves trails behind it. What happens?[1]

6. Finally, consider what happens if you move the Sun away from the origin. Suppose you put the Sun at coordinates (0.2, 0.3). Change your code to simulate orbits around a star not at the origin.

[1]It turns out there is some history here. In regions of very strong gravity, Einstein's general relativity modifies the force from gravity in a similar way. The orbit of Mercury is close enough to the Sun that GR causes its orbit to precess about one percent of one degree every century; this was known to astronomers and unexplained at the time of Einstein. Einstein was aware of this, and concluded his paper introducing GR by pointing out that his correction to the law of gravity explains exactly the anomaly in the precession of Mercury's orbit.

1 Hints

- **Computing GM:** Remember your freshman physics; what is v^2/r? I know some of you took or taught Physics 211 with me or Physics 215 with Prof. Laiho; treat it as a 215 problem! If you're not sure how to go about this, discuss with other students, but tell me how this goes on your report.)
- **Debugging:** If something is wrong with ...
  - Animation? Run your program without `anim` and see what it is printing out
  - The physics ("planet disappears"): print out everything your algorithm is doing, and examine the very first step in detail. (You can see output one page at a time with `./myprogram | less`.
  - Elliptical orbits: is your total energy positive or negative?
- **Animation:**
  - You can make your planet leave behind a "trail" so you can visualize its orbit with the `ct3` animation command. Its syntax is `ct3 ⟨trail index⟩⟨x⟩⟨y⟩⟨z⟩⟨radius⟩`, where:
    - `⟨trail index⟩` controls which trail you are adding to in this case you only want to leave behind one trail, so just set this to 0
    - `⟨(x,y,z)⟩` is the location to draw the object. Since you are simulating orbits in the xy-plane, just set z to zero.
    - `⟨r⟩` is the radius of the ball to draw.
  - Since the Sun doesn't move, you can draw it at the origin with the `c3` command.
  - If your simulation runs too fast, reduce the timestep; if it runs too slowly, you can draw only one animation frame every 10, 100, etc. timesteps. To do this you might write something like:

```
int steps=0, stepsperframe=10;
...

for (loop over time...)
{
  steps++;   // note that this means the same as "steps=steps+1"
  if (steps % stepsperframe)
  {
    <do anim things>
  }
  <do physics update (Euler-Cromer-Aspel, leapfrog, etc.)>
}
```

- **Displaying the energy (or whatever else) on the simulation:** An example of an elaborate way to do this would be:

```
printf("T 0.9 -0.9\n");
printf("Energy (U + V = E): %e + %e = %e\n",potential,kinetic,potential+kinetic);
```

(Notice that this is a *two-line* `anim` command: the first line (`T 0.9 -0.9`) instructs the computer to print some text at a specified point in the window, in this case near the top left. The second line contains the text to be printed.) Remember, the more information you have about what your code is doing, the more physics you can learn from it, and the easier it will be to fix bugs.