Name

Recitation Assignment # 1

Sep. 26, 2005

You should consider this a trial by fire. If you've never done any programming before, don't worry. We're here to walk you through it. However, you should always have a general idea of what your program is doing. Also, if you've never used linux before, don't worry. You'll ge the hang of it.

You may complete this in class. However, if you are unable to do so, it is expected that you complete this for recitation next time.

Where a box appears, call over Travis to check over your progress. When the sheet is complete, you will hand it in. Also, you are expected to email your final programs to Travis.

If you have any questions, please ask.

Today, we are going to make a working model of the earth going around the sun!

1. Log into a workstation, pull up firefox, make yourself comfortable, etc.
2. In firefox, go to the webpage *http://vpython.org/*, which will serve as a handy manual for programming in visual python (the language that we will be using throughout this course).
3. Open up a "terminal" or "shell," and, if you have not already done so, create a directory called contemporary1. e.g.:
   `% mkdir contemporary1`
   change to the new directory
   `% cd contemporary1`
   check to make sure you are in the correct directory. Typing
   `% pwd`
   should produce:
   /home/newton5/*yourusername***/contemporary1**
4. Now you are ready to start! Run
   `% emacs prog1.py &`
   This will pull up the emacs file editor. (The "&" allows you to keep using the shell while emacs is running).

□ 5. Write the following simple program and run it:

```
from visual import *
sun=sphere(radius=7e8,pos=vector(0,0,0),color=color.yellow)
```

This simple program has two commands. The first tells the python interpretor to include all of the visual libraries (the commands that draw on the screen).

The second command tells it to create a sphere called "sun" with a radius of 7×10^8. In our case, we are generally going to work in mks units (meters, kilograms, seconds), and so we

want to be consistent. There's nowhere to put units into a program, so you have to make sure that the units all cancel. We are also saying that the sun should be drawn yellow.

The part with "pos=vector(0,0,0)" tells the program to "place" the sphere at the origin.

Save your program, and run it by typing,"python prog1.py" on the command line.

6. The earth has a radius of 6.4×10^6m, and is a distance of 1.5×10^{11}m from the sun. Create a sphere called "earth" (make it blue) which lies on the x-axis, and has these specifications.

   What's that? You can't see the earth, and you can only barely see the sun? Don't panic. This is how the solar system would actually look on these scales. In reality, space, even within the solar system, is very, very empty. Change your code so that the sun is 20 times larger than "normal" and the earth is 1000 times larger.

□ 7. Add the following lines to your code:

```
scene.range=2e11
scene.autoscale=0
```

   The first line makes the viewscreen large enough to see a bit beyond the "orbit" of the earth, and the second command tells the code not to automatically resize the picture if the earth flies out of the field of view.

8. Hold down various buttons on your mouse and move the mouse within the display window. See what happens! Left, right, and both. Try them all!

9. Here's where we start to add physics. The earth has a mass of $M_\oplus = 6 \times 10^{24}$ kg. You can set this in the code by writing:
   `earth.m=6e24`
   Similarly, the sun has a mass of 2×10^{30} kg. Put the sun's mass into the program.

   Give the earth a velocity of 35 km/s=35000m/s in the y-direction.

   This can be done by entering:
   `earth.v=vector(0,35000,0)`

   Since we generally deal with momentum, you want to express the momentum in terms of the velocity:
   `earth.p=earth.m*earth.v`

   Make sense?

   Now, run the code! What happens?

   Nothing, eh? That's because you haven't yet told the program how to move the earth around. It doesn't know any physics, after all.

□ 10. Remember the relation we discussed in class:

$$\Delta\vec{r} = \frac{\vec{p}}{m}\Delta t$$

   Well, at each timestep, you have to evolve the position of the earth, according to this formula. Put in the following additions to your code:

```
yr=3.15e7
dt=0.02*yr
t=0

while (t < 2*yr) :
    rate (10)
    earth.pos=earth.pos+dt*earth.p/earth.m
    t=t+dt
```

The first three commands tell the code how many seconds there are in a year. The second tells it what the value of Δt is. The third tells it to "start the clock" at t=0.

The "while" statement tells the code to keep running for 2 years. The "rate" statement says to evolve everything 10 times a second (otherwise it would run too fast for you to see!). The last two commands update the position of the earth and the clock every timestep.

□ 11. As we saw, the earth just flies in a straight line, when, in fact, we know that it moves around the sun. What makes it move? Gravity! The force of gravity can be expressed as:

$$\vec{F} = -\frac{GM_\odot M_\oplus}{r^3_{sun-earth}}(\vec{r}_\oplus - \vec{r}_\odot)$$

where $G = 6.67 \times 10^{-11} \frac{Nm^2}{kg^2}$ is the gravitational constant.

Remember, the momentum principle states that:

$$\Delta\vec{p} = \vec{F}\Delta t$$

So, before the "while" loop, put in the definition of the gravitational constant. Within the loop, calculate the force, and update the momentum. Only 3 lines need to be added to your code.

□ 12. To prove you've mastered the code, do the following:

- Adjust the initial velocity of the earth so that it moves more nearly in a circular orbit.
- Run the code for 5 years rather than 2.
- **Extra Credit:** Put Jupiter into your system. Find the mass, distance from the sun, and orbital velocity of Jupiter by using google.
- **Extra Credit:** Use the online tutorial to follow Earth's path.