PHYSICS 113 – Recitation Assignment 3

Name

Recitation Assignment # 3

Oct. 11, 2006

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 a solid, by stringing together a number of simple harmonic oscillators. I will give you considerably less guidance on writing this program than the previous time, so you should definitely have a copy of the VPython website open, and last week's recitation assignment in front of you.

New Physics Concepts: Oscillators, Molecules

1. Log on, start a shell, and using emacs, start editing a program called *prog3.py* in your "Contemporary1" directory.

2. Begin by making two "atoms" appear, at x=0, and x=1, respectively. Both spheres should be white, and have a radius of 0.2. For convenience, we'll work in meters, even though this means that our giant "molecule" is enormous! Connect the spheres using a "curve" (our spring!). The way you want to do this is using code similar to:

```
springs=curve()
springs.append(pos=atom1.pos)
springs.append(pos=atom2.pos)
```

Note: If, at any point, you want to move one point on the curve, simply adjust the position of that point:

```
springs.pos[1]=atom2.pos
```

to adjust the right-hand side of the curve, for example.

□ 3. Now it is time to add the physics. Use your example from last time. Begin by moving (setting the position of) the second atom to (1.1,0,0). Now, the force will look something like:

```
x0=1    # Put this line in at the top
k=10    # You can change this later
m=1     # You can also change this later
atom1.p=vector(0,0,0) # also at the top
atom2.p=vector(0,0,0) # at the top
```

```
...

# Put the beginning of your ‘‘while’’ loop here!

s=mag(atom2.pos-atom1.pos)-x0
F2=-k*s*vector(1,0,0)
F1=-F1
```

After you compute the force, evolve the position and velocity in the same way as before. Use a timestep of 0.001 seconds, a rate of 100, and run while $t < 5$. You should see your "atom" oscillating back and forth!

□ 4. In some sense the previous method is tiresome. Imagine if you wanted 10 or 20 atoms in your solid – You'd have to evolve each one separately! This is why arrays are useful. Arrays are essentially a list of objects. Begin by saving your new draft as: prog3b.py.

You can create an empty array of objects with the command:

```
atoms=[]
```

Each time you want to add something to your array, you can do so with the command:

```
atoms.append(sphere(pos=(1,0,0),radius=0.2))
```

for example. Then, if you wanted to change, say, the position of the 3rd atom later on, you could write:

```
atoms[2].pos=vector(1.1, 0, 0)
```

Note that the 3rd particle uses an index of 2. That is because we start counting at 0.

In order to make the setup efficient, we don't want to do all of that manually. We need a "for loop." A for loop runs through a list of numbers, and does something at each number. For example:

```
for i in range(10):
     print i
```

would print out the numbers 0-9.

So, near the top of your code write a line like:

```
natoms=4
```

and then, further down, write:

```
for i in range(natoms):
     atoms.append(sphere(pos=(i,0,0),radius=0.2))
```

This will create *natoms* atoms, each spaced at 0,1,2 and 3.

5. Start writing a program which evolves this system when the rightmost atom is displaced, say, 0.1m to the right. You will once again use a "for loop" to compute the forces on each atom, and evolve the positions and momenta.

   Note that he for loop should be run once per timestep. This is an example of a nested loop.

   You can use indices to compute the forces between adjacent atoms. For example:

```
for i in range(natoms):
       dx=atoms[i].pos.x-atoms[i-1].pos.x-x0
```

   computes the distance from equilibrium from the i-th particle. Be careful, though. The code I just wrote won't really work. Why not? Because the 0th particle has nobody to the left of him. In order to correct this, you'll need the if statement:

```
for i in range(natoms):
     if (i>0):
          dx=atoms[i].pos.x-atoms[i-1].pos.x-x0
```

   You should notice your lattice of atoms vibrating back and forth.

□ 6. Once you've got that working, you're ready to compute the speed of sound in your material. Basically, since $v = d/t$. Write an "if statement" to figure out the time at which particle 0 starts vibrating (say, when it is a distance d0/2 from the initial). Print out the speed of sound. Compare this to the interparticle spacing (x0) times the natural frequency $\sqrt{k/m}$.