Name

Recitation Assignment # 4

Oct. 17, 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.

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 Concepts: Physical

- Oscillators
- Molecules

Computational

- New VPython Object: Helix

1. Log on, start a shell, and using emacs, start editing a program called *prog4a.py* in your "Contemporary1" directory (or whatever you have chosen to call it).
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 "helix" (our spring!). The way you want to do this is using code similar to:

   ```
   spring=helix(pos=atom1.pos,axis=atom2.pos-atom1.pos)
   ```

   You should also set it so that the helix has 4 coils.

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

   ```
   spring.pos=atom1.pos
   spring.axis=atom2.pos-atom1.pos
   ```

   Now, set the scene.autoscale to 0 so that the scene doesn't constantly resize.
3. Now it is time to add the physics. The fundamental equation is, of course, Hooke's law:

   $$F = -k(x - x_0) \tag{1}$$

   where x_0 is the unstretched separation between atoms. Begin by moving (setting the position of) the second atom to (1.1,0,0). Now, the force between the two can be implemented by:

```
x0=1    # Put this line in at the top
k=10    # You can change this later
m=1     # You can also change this later

...
(within the loop)
...

s=norm(atom2.pos-atom1.pos)*(mag(atom2.pos-atom1.pos)-x0)
F2=-k*s
F1=-F2
```

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. Save your working 2 atom model as prog4a.py, and edit a new version of code as prog4b.py.

   Create an array of atoms and springs

   ```
   atoms=[]
   springs=[]
   ```

   each of which should be a sphere.

   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.

   Similarly, make $natoms - 1$ springs (helixes) connecting the atoms.

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.

   At each step in the time loop, you will want to start by zeroing out your forces. Why? Because each atom will have a contribution from the atom to its right and to its left. So, you'll want a bit of code along the lines of:

   ```
   for i in range(natoms):
        atom.F=vector(0,0,0)
   ```

After zeroing the forces, you'll want to compute the forces between adjacent atoms. Here's a hint, if there are 10 atoms (for example), only compute the forces between the first 9 and the atoms to their right.

For example:

```
for i in range(natoms-1):
       s=norm(atoms[i+1].pos-atoms[i].pos)*(mag(atoms[i+1].pos-atoms[i].pos)-x0)
       F1=-k*s
       atoms[i].F=atoms[i].F-F1
       atoms[i+1].F=atoms[i+1].F+F1
```

which computes the distance from equilibrium from the i-th particle.

Run the entire loop, and then put in a 3rd loop (the first zeros the forces, the second computes the forces), still nested under the time loop. In this loop, you should update the positions and momenta of the atoms, as well as updating the positions and stretches of the springs.

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 $\sqrt{k/m}$.

7. Make sure you email both programs (with your name in a comment at the top) to Travis.

8. **For fun** Take the following code:

```
while init == 0:
    if scene.mouse.events:
        m1 = scene.mouse.getevent()
        for i in range(natoms):
            if m1.drag and m1.pick == atoms[i]:
                j=i
            elif m1.drop:
                atoms[j].pos=m1.pos
                d0=atoms[j].pos.x-j
                atoms[j].y=0
                init=1
```

before your time loop. This will allow you to drag an atom to start the oscillations. If you include this, make sure you take out the statement explicitly displacing the rightmost atom.

9. **E.C.** Create a triangular solid. That is, make 4 atoms on the bottom row, 3 on the second to bottom, etc. This is 2-d.