Physically Based Modeling
CS 15-860 Spring 1994
Assignment 2: Particle/Spring System
Due February 8

In this assignment, you'll build a system that simulates collections of particles attached together with springs. The particles will move about under the influence of the spring forces between particles, gravity forces, and an interactive mouse force.

What to Simulate

You can wire a particle/spring model into your code, or read model descriptions from files, or build them interactively. If you choose to hard-wire your model, make it contain four particles, three of which are connected by springs to form a triangle, with the fourth attached to one of the first three by an additional spring. (or do something more complicated.)

The springs between particles should have damping and nonzero rest length. Both the stiffness of the springs and the amount of damping should be specified by constants that you can change (either as part of the interface, or using environment variables if you're not into interfaces).

Collisions and Contact

To keep the particle-collection from drifting off the screen, you should bound the screen by four planes, which the particles cannot pass through. If a particle is found on the "wrong" side of a plane, displace the particle along the normal to the plane, and then change the particle's velocity v with an impulse normal to the plane, as discussed in class. Implement collisions with the coefficient of restitution $\epsilon = 0$, and $\epsilon \neq 0$. (As a special case, and to make sure you're doing impulses correctly, simulate a single particle, colliding with one of the walls, and set $\epsilon = 1$. Verify that the particle's kinetic energy $\frac{1}{2} v \cdot v$ before and after the impulse is applied is the same. To simplify your life, you can use Euler's method if you want. (If you want to use the Midpoint or Runge-Kutta method, you'll need to modify the solver to deal with the discontinuities caused by collisions. Ask if you have a question about this.)

Also, if you detect a particle in contact with a wall, and its approach speed v_N is zero, compute and apply an appropriate constraint force normal to the wall to prevent the particle from moving through the wall. (If you find that v_N is positive, don't apply a constraint force: the particle is moving away from the wall and doesn't need any more help from you.) You should be able to watch your particle collections collide with a wall, compress, and then rebound away from the wall, using $\epsilon = 0$ collisions and constraint forces. In detecting

contact, you must allow for some numerical tolerances. That is, declare any particle within some small distance δ of the wall to be "in contact." Likewise, declare any particle with v_N close to zero to be "at rest" on the plane (requiring a constraint force). The choice of suitable tolerances is up to you.

Interactivity

Lastly, to make your simulations interactive, implement a mouse-spring as follows: each time the mouse button is pressed, find the particle closest to the mouse at that moment. Apply an attractive spring force (damped, with zero rest-length) between that particle and the mouse until the button is released. Use the mouse to throw your particle creature at the walls.

There should also be a gravity force that acts on all the particles.

Be sure to indicate which particles are attached to which particles by drawing lines between them (and draw the particles themselves—something preferably larger than a single pixel). Also, draw the walls that the particles cannot pass through. (You can implement more than four walls if you wish: the legal space for the particles can be a convex polygon, bounded by planes at arbitrary angles if you like.)

Optional Stuff

Add friction to particle/wall contact; try to make the walls sticky.