CS176, Project 6

Careful: Two deadlines! (2/18/11 and 2/25/11)

Part I

For this project you are asked to implement a cloth simulator based on Baraff and Witkin's paper. However, we are going to be using a different expression for the energy than the one mentioned in the paper. In order to make your coding easier, we will ask you to take some time to derive the gradient and Hessian of the energy by yourself. This part of the assignment is due Friday 2/18 (return the write-up to Mathieu's office (or in electronic form)) This means that you need to do the derivations started in class—but it will be easier than in the paper.

We will use a fairly simple energy which is expressed as a sum of energies over each triangle and edge in the mesh. For each edge we have a clear notion of the change in length and for each triangle the change in area. The expression we will use for the energy is:

$$\begin{aligned} E = \alpha E_e + \beta E_t &= \alpha \sum_{e \in E} (1 - \frac{|e|}{|\bar{e}|})^2 |\bar{e}| + \beta \sum_{t \in T} (1 - \frac{A_t}{\bar{A}_t})^2 \bar{A}_t \\ &= \alpha \sum_{e \in E} (|\bar{e}| - |e|)^2 \frac{1}{|\bar{e}|} + \beta \sum_{t \in T} (\bar{A}_t - A_t)^2 \frac{1}{\bar{A}_t} \end{aligned}$$

where the overbars refer to quantities in the original configuration.

e is an edge $(x_i - x_j)$.

A is the area of triangle (x_i, x_j, x_k), i.e. half the magnitude of the cross product: $\frac{1}{2}|(x_i - x_j) \times (x_k - x_j)|$

Notice that this corresponds to specific constraints ($C_e(x)$ on edge lengths, and $C_A(x)$ on triangle areas), different from the Baraff & Witkin's ones.

Recall from Baraff and Witkin paper that when solving the linear system we need the forces and their partials.

The force at each vertex is the negative gradient of energy with respect to x_i (position of vertex i):

$$-\left(\alpha \sum_{e \in N(x_i)} \frac{\partial E_e}{\partial x_i} + \beta \sum_{t \in N(x_i)} \frac{\partial E_t}{\partial x_i} \right)$$

where we sum over the edges and triangles that are adjacent to vertex i.

Now you need to take each component of the energy and differentiate it with respect to x_i. Try to think of geometric properties when you are doing the derivation. Use vector calculus to carry out your derivation (do not expand out into square roots of sums of squares, for example), and make heavy use of the chain rule and product rule. If you like, you can check yourself in Mathematica or Maple. (However, this check can only be numerical; the expressions you get from them will have all the length and area computations expanded out.)

Next, for the partial derivatives of the forces $\frac{\partial \mathbf{f_i}}{\partial \mathbf{x_j}}$, you need to derive the Hessian matrix $\mathbf{K}$. You will need to find the entries of the matrix (remember that the Hessian is symmetric, but you might want to check to convince yourself):

$$K_{ij} = \frac{\partial^2 E}{\partial x_i \partial x_j},$$

$$K_{ii} = \frac{\partial^2 E}{\partial x_i \partial x_i},$$

As you are working on the Hessian also keep your work for E_e and E_t separate until you have results for both parts.

Part II

Now that you have your matrices, on to the implementation. Take a look at equation 15 in the Baraff and Witkin paper. This is the linear system we're solving:

$$\left(\mathbf{M} - h\frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\Delta\mathbf{v} = h\left(\mathbf{f} + h\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{v}\right)$$

You should read the paper to make sure you understand what all these terms mean and where this equation came from.

First of all you have your state vector $\mathbf{x}$ of size $3n$, where $(x[0], x[1], x[2])$ is the position of vertex 0, $(x[3], x[4], x[5])$ is the position of vertex 1, etc. Similarly for $\mathbf{v}$.

Next we need the forces. First of all, set β from above to 0 and α to 1; in other words, *do not implement the area term.* We're also not going to be doing any bending forces. You already have $\mathbf{f}_e = \frac{\partial E_e}{\partial \mathbf{x}}$, which you derived — again 3 components per vertex — and $\frac{\partial \mathbf{f}_e}{\partial \mathbf{x}}$ is your $3n \times 3n$ Hessian matrix $\mathbf{K}_e$. (Since this part doesn't depend on velocity, $\frac{\partial \mathbf{f}_e}{\partial \mathbf{v}} = 0$.)

Next we need damping forces. Create $\mathbf{f}_{damping}$, $\frac{\partial \mathbf{f}_{damping}}{\partial \mathbf{x}}$, and $\frac{\partial \mathbf{f}_{damping}}{\partial \mathbf{v}}$ as described in section 4.5 of the paper, using $(\ C(\mathbf{x})\ )_i = (|e_i| - |\bar{e}_i|)/|\bar{e}_i|$. Remember to drop the non-symmetric term.

To those you must add your external forces, gravity and wind.

Gravity is easy: just add a user-defined constant to the y component of the $\mathbf{f}$ vector, and since it's constant it doesn't contribute to the terms with derivatives. For wind, come up with some analytic function in $\mathbb{R}^3$ to use as your wind vector field (try a tensor product of some sinusoids, for example). Fill in your $\mathbf{f}_{wind}$ and $\frac{\partial \mathbf{f}_{wind}}{\partial \mathbf{x}}$ vectors by evaluating this function and its derivative at each vertex position. (We recommend doing everything with just gravity first, and then adding wind only after everything else is working.)

Then add your external forces to the internal forces, $\mathbf{f} = \mathbf{f}_e + \mathbf{f}_{damping} + \mathbf{f}_{gravity} + \mathbf{f}_{wind}$.

With all that set up, now you can finally solve for $\Delta\mathbf{v}$. And then $\mathbf{v}_{new} = \mathbf{v} + \Delta\mathbf{v}$, and $\Delta\mathbf{x} = \Delta t \cdot \mathbf{v}_{new}$, *except* for the vertices which you have constrained; for those, simply don't update the position.

We will be providing you with two cloth meshes to use (both the same, one low-resolution and one high-resolution). Implement the following test cases:

1. No wind. The patch starts horizontal. Constrain two adjacent corners of the cloth patch, and let gravity do its thing.

2. Flag. The patch starts vertical. Again, constrain two corners of the patch, but this time apply your wind force along with the gravity.

3. Custom. Make something pretty. Some suggestions:

   - Time-varying wind.
   - Add an air-drag force (opposes velocities along the triangle's normal direction).
   - "Seams/Pleats/Weaves": Try varying the cloth density across the patch according to some pattern.
   - "Waving Handkerchief": Move one of the vertices around, either through a random walk or some preprogrammed path, and watch the rest of the patch follow.
   - "American Beauty": An upward wind keeping the cloth afloat (without any vertex constraints).

   Document what you did.

Provide a way, either through the command line or the UI, for us to set up the initial state according to case number (1, 2, or 3).

Please submit your implementation by Friday 2/25 11:59pm.

Extra Credit

There are lots of opportunities for extra credit on this one. Implement any of the many things we skipped over in the paper for extra credit.

- The biggie is bending energies. Add forces to your system corresponding to the bending energy defined in section 4.3 of the paper.

- You could try object-cloth collisions. See section 6 of the paper.

- Mass modification. Add the ability to constrain a vertex to lie on a line or a plane, and then do a shower curtains setup or something similar.