CS838 Topics in Computing: Advanced Modeling and Simulation Fall 2011

Programming ssignment #1

Due Wednesday 26 Oct 2011, 1:00pm

For this assignment, you are asked to generate a computer simulation of a simple cloth model (a waving banner, suspended from a moving horizontal rod).

A video illustration of the general look you should try to recreate is shown in the following movie:

`http://pages.cs.wisc.edu/~cs838-2/media/cloth.avi`

You are free to deviate from the simple waving motion demonstrated in this movie, as long as you script a motion scenario that creates an interesting animation. You are also free to adjust material parameters (stiffness, damping), simulation settings (frame rate, etc) and cloth model resolution, although you should demonstrate that your system is able to handle cloth resolutions of at least 20×20 (the movie shown is at a resolution of 20×40).

1. Implement the described cloth simulation using either a Forward Euler integration scheme (see our Sep 28th lecture), or a semi-implicit scheme (e.g. as described in our Oct 3rd lecture). Code samples have been posted on the CS838 website along with lecture notes for both of these dates. If you choose to use the driver/layout scheme as discussed in class, you should be able to implement this part by almost exclusively limiting your changes to the files `SIMULATION_LAYOUT.h/cpp`, without changing the driver at all. If you choose to structure your simulation software in a different fashion, feel free to do so.

   The cloth model should be a triangulated surface. You are free to generate the cloth geometry of your choosing (it does not even have to be a rectangle). However, if you choose to model a simple rectangular banner, it is recommended that you tessellate it using what is called a herringbone mesh: The rectangle is subdivided into small squares/rectangles in a Cartesian lattice, and every square/rectangle is triangulated by drawing a diagonal in *alternating* directions, in order to limit mesh anisotropy. You can find helpful code for generating a herringbone mesh in the `TRIANGLE_MESH` class. You can use the example below as a guideline:

```
cloth_surface=TRIANGULATED_SURFACE<T>::Create(particles);
cloth_surface->mesh.Initialize_Herring_Bone_Mesh(10,20);
particles.array_collection->Add_Elements(200);

particles.array_collection->Add_Elements(n);
for(int p=0,j=1;j<=20;j++) for(int i=1;i<=10;i++)
   particles.X(++p)=TV((T)(1-i)*0.1,(T)(1-j)*0.1,0);
```

You should generate a network of masses and springs, aligned with the *edges* of the triangulated surface described above (you will need to generate the mesh of line segments corresponding to springs, by initializing the appropriate "derivative" mesh of the triangulation). You do not need to employ altitude springs or other elaborate spring connectivity.

2. Implement a *fully implicit, Backward Euler* time integration scheme for the cloth simulation you set up in the previous step. If you have followed the driver/layout methodology, the files `SIMULATION_LAYOUT.h/cpp` should not need to be changed at all (with the exception of `Maximum_Dt()` which can be much more aggressive, when using Backward Euler, and other simulation settings that you may need to adjust).

   You may also consider, for ease of development and debugging, to implement the Backward Euler solver on top of the simulation of the single strand, that was used for the lectures (and accompanying code) of 9/28 and 10/3. If done right, you should then be able to directly combine your modifications to the driver (for the purposes of supporting Backward Euler) with the layout changes of part 1, to have a functional Backward Euler cloth simulator.

For the needs of this assignment, you may have to change the interface of the driver/layout classes. For example, when setting boundary conditions for the linear system of equations that determines the position chance $\delta X = X^{n+1} - X^n$, you may want to add a different, more convenient function to the layout class. Make all such design choices according to your best judgement.

Deliverable: Source code and (optionally) a compressed archive of the simulation output directories. Include a short report (1 page should suffice) describing the high-level changes you had to implement, and the maximum *dt* that you have been able to take using Backward Euler, or the alternative technique tested in part 1. You can email the code to the instructor directly. If output files exceed 2-3MB, it is best that you send a download link for them.