MASS–SPRING CLOTH SIMULATION

Computer Graphics Final Project | CS 180

Christie Wang *Daniel Kluzner* *Susan Wu*

Overview

For the mass-spring cloth simulation, we follow the framework provided in Assignment 8 and use Implicit Euler to update the positions. Our cloth is represented as a 16×16 grid of masses and springs, with diagonal springs in each grid cell. In order to solve for the change in velocity dv and update the positions based on Implicit Euler, we use the following derived equation:

$$(M - dt \cdot \frac{df}{dv} - dt^2 \cdot \frac{df}{dx}) \cdot dv = dt \cdot (f_0 + dt \cdot \frac{df}{dx} \cdot v_0)$$

We are solving the equation $A \cdot dv = B$. To do this, we first compute the Jacobians $\frac{df}{dv}$ and $\frac{df}{dx}$ for each spring. Then the function `multiplyDfDx()` multiplies a $3n$ vector (where n is the number of masses) with $\frac{df}{dx}$. The similar function `multiplyA()` multiplies a $3n$ vector by matrix A. We use `multiplyA()` and `calculateB()` to perform conjugate gradient descent in `CG()`, solving for dv. Once we have dv, we update the positions in the same manner as in Assignment 8 with Implicit Euler and update positions of the masses in `simulateEuler()`. This particular method was used because it incorporates damping into the Euler method.

Libraries used

`<fstream> <string> <array> <algorithm> <numeric> <vector>`

Responsibilities

- Christie Wang
  - `Cloth(), multiplyA(), outputOBJ(), dot_helper()` + lambda func.
  - Adapted Assignment 8 for our purposes (e.g. 3D vectors)
  - 8 hours of rendering .obj files in Houdini Apprentice
- Daniel Kluzner
  - `~Cloth(), ~Spring(), calculateForces(), computeJacobians(), simulateEuler(), multiplyDfDx(), multByScalar()`
  - Figured out how to use `std::transform()`
- Susan Wu
  - `calculateB()`
  - Composed Works Cited
- Christie and Daniel collaborated on `CG()`
- Susan and Daniel both provided voice-over for YouTube video

Compilation and Execution

Unzip the .zip file, and from inside the project folder, run the following terminal commands:

```
mkdir build
cd build/
cmake ..
make
./ropesim
```

The executable is indeed `ropesim`. This was done to not unnecessarily interfere with the original structure. A new window will be generated with the simulation. You will look at the (x, z) plane towards the $-y$ direction. Watch the video for a clearer 3D viewing experience. In `Application::render()`, the line that outputs the sequence of .obj files is commented by default. Uncomment to test it out for yourself.

Implementation Details
Regarding the details of the code base, we readapted the code from Assignment 8 for mass-spring cloth simulation. In doing so, we changed the 2D vectors to 3D vectors and float types into double types for precision. The code directory/structure is the same as Assignment 8 with the following notable exceptions:

`spring.h`

- Jx and Jv which are Jacobian matrices: $Jx = \frac{df}{dx}$, $Jv = \frac{df}{dv}$
- `n[2]` are the indices of the two masses that are the endpoints of the spring
- Damping constant $kd = 0.8$

`cloth.h`

- `cloth(Vector2D start, Vector2D end, int num_nodes, float node_mass, double k, vector<int> pinned_nodes)`

  Given a corner of the cloth (`start`) and its opposite corner (`end`) in the (x, z) plane, we construct a `num_nodes` cloth where each mass has node_mass and each spring has stiffness constant `k`. The cloth is represented as a vector of masses, a vector of springs, and a vector of faces, where each element of the vector of faces is a vector of ints. The vector of ints specifies the indices of the masses that make up the face of the cloth.

- `void multiplyA(std::array<Vector3D,256> src, std::array<Vector3D, 256>& dst, double delta_t)`

  Takes in `src`, multiplies it by A, and outputs the result to `dst`

- `void outputOBJ(string file_name)`

  Creates a new file called `file_name` and outputs the object file version of the cloth

Works Cited

Baraff, D. & Witkin, A. Large Steps in Cloth Simulation. In Proceedings of SIGGRAPH, 1998

Baraff, D. Implicit Methods for Differential Equations. SIGGRAPH course notes, 2001

Bridson, R., Fedkiw, R. and Anderson, J. Robust Treatment of Collision, Contact and Friction for
Cloth Animation. In Proceedings of SIGGRAPH, 2002

Choi, K.-J. & Ko, H.-S. Stable but Responsive Cloth. In Proceedings of SIGGRAPH, 2002

Volino, P. & Magnenat-Thalmann, N. Implementing Fast Cloth Simulation with Collision
Response. Computer Graphics International 2000