Assignment 3 : Cloth Simulation using Mass-Spring System

NAME: TIANYUAN WU
STUDENT NUMBER: 63305667
EMAIL: WUTY@SHANGHAITECH.EDU.CN

1 INTRODUCTION

In this assignment, I implement a simple cloth animation using a mass-spring system. The implementation is in C++11 with OpenGL. This implementation contains 3 parts:
(1) Triangle mesh builder / Cloth Renderer
(2) Mass spring system solver
(3) Constraint systems
A demo of my implementation is shown below:

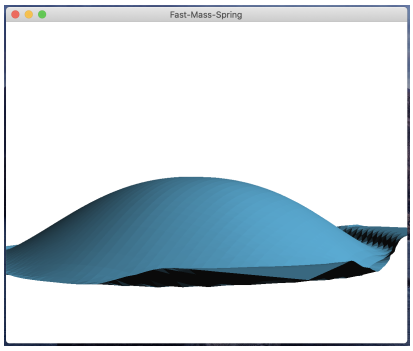

Fig. 1. Demo of cloth simulation

For the solver part, I used the method described in the paper "Fast Simulation of Mass-Spring Systems, Tiantian Liu, 2013" to achieve real-time efficiency with details better preserved animation.
The demo program is compiled under `macOS 10.15` with `Apple clang 12.0` with 3rd party libraries: `OpenGL, glm, Eigen3`, and it's tested that it can preform simple cloth animations correctly.

2 IMPLEMENTATION DETAILS

2.1 Mesh generation

For a given cloth with regular shape, it's simple to implement a mesh generation algorithm. For example, the mesh for a rectangular cloth can be generated as the following figure

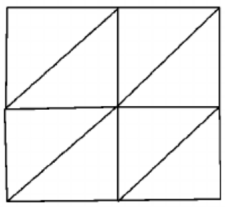

Fig. 2. Mesh of rectangle

In this demo program, I just divide the rectangle into $m * n$ parts, and then divide each part into 2 triangles by its diagonal.
But for more complex shapes, we may need more complex algorithms like Delaunay triangulation (it's not implemented in this assignment).

2.2 Mass spring system construction

For a simple rectangular cloth, the mass-spring system can be constructed as the following figure:

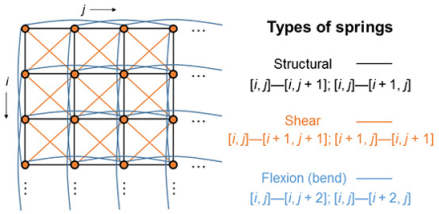

Fig. 3. Mass spring system

There are 3 types of springs:

(1) Structural springs: connect $M(i, j)$ with $M(i, j+1)$ and $M(i+1, j)$
(2) Shearing springs: connect $M(i, j)$ with $M(i+1, j+1)$, connect $M(i+1, j)$ with $M(i, j+1)$
(3) Bending springs: connect $M(i, j)$ with $M(i, j+2)$ and $M(i+2, j)$

When the masses is given, the springs can be simply constructed by the equation shown above.

2.3 Mass spring solver

Up to now, the mass spring system is already done, we just need to solve this system by the method described in the paper. The details and proof of these equations are shown in the paper, so I just

introduce the equations used in my implementation.
First, we calculate the mass matrix $M \in \mathbb{R}^{3n\times 3n}$:

$$M(i,j) = diag(m_1, m_1, m_1, \ldots, m_i, m_i, m_i, \ldots, m_n, m_n, m_n)$$

Then, calculate the matrix $L \in \mathbb{R}^{3n\times 3n}$:

$$L = (\sum_{i=1}^{s} k_i A_i A_i^T) \otimes I_3$$

where $A_i \in \mathbb{R}^n$ is the incidence vector of i-th spring, and I_3 is the 3×3 identity matrix, and s provides the number of springs. In this simple demo, the number of springs $s = (n-1)(5n-2)$, which equals to the number of sum of 3 types of springs.
Then we need to build the matrix $J \in \mathbb{R}^{3n\times 3s}$:

$$S = (\sum_{i=1}^{s} k_i A_i S_i^T) \otimes I_3$$

Here, S_i denotes i-th spring indicator, where $S_{i,j} = \delta(i,j)$, and we denote the external force as $f_{ext} \in \mathbb{R}^{3n}$. Then we can minimize the energy:

$$E = \min_{d\in U} \frac{1}{2} x^T L x - x^T J d + x^T f_{ext}$$

We aggregate the external force and inertia into a vector b and we can rewrite this optimtzation problem as:

$$\min_{x\in\mathbb{R}^{\#>},d\in U} \frac{1}{2} x^T (M + h^2 L) x - h^2 x^T J d + x^T b$$

Starting with an initial guess for x, we can first perform a local step: fix x and compute the optimal d. Second, we perform a global step: fix d and compute the optimal x, repeating this process until a maximal number of iterations reached.
And for the constraints (collision), note that any translation of x can be accomplished by appropriately chosen f_{ext}. Therefore, we can short-circuit this process and instead of computing $f_e xt$, we directly compute by collision response routines and move x to the collision-free state, instead of calculating the collision forces explicitly.
Thanks to `Eigen` provides us a efficient and easy to use API for matrix computation and optimiazation, the solver program can be done within 200 lines of code.

3 RESULTS

In this assignment, I perform 2 different animation in the demo. The first one is a square cloth dropped and collided with a fixed square, and the second one is a square cloth getting a impulse (an straight up external force) on a circle area at its center while dropping.
(1) Results of the first demo:

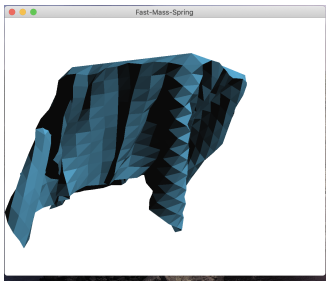

Fig. 4. collision between cloth and a fixed square - 1

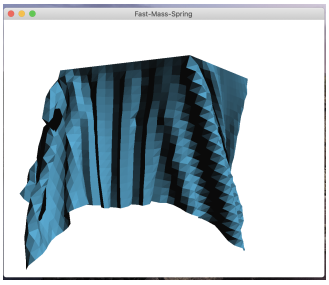

Fig. 5. collision between cloth and a fixed square - 2

(2) Results of the second demo:

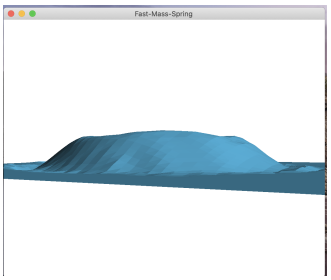

Fig. 6. Circle impulse on the cloth - 1

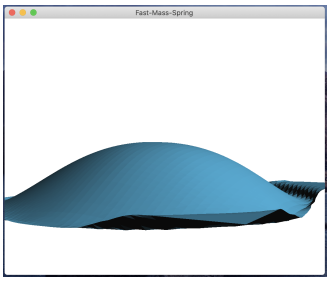

Fig. 7. Circle impulse on the cloth - 2