Assignment 5:
Animation with Cloth Simulation

NAME: HAIRUI YIN
STUDENT NUMBER: 2020533028
EMAIL: YINHR@SHANGHAITECH.EDU.CN

1 INTRODUCTION

In this assignment, we are required to implement a simple cloth simulation based on mass spring system to do a physically based animation. In the mass spring system, cloth is divided into small mass connected with small spring and is effected under the force of a neighboring particle. Since it is an ideal condition in cloth simulation, we should also consider some methods to simulate resistance force of air to make the object has a tend to stable.
The followings are all I have done, consisting must part and bonus part of wind and spheree collision handling.

- Force computation with Hooke's law
- Structural, shear, and bending springs
- Fix the location of two mesh points to stop the cloth falling down
- Real-time and stable animation
- Apply external forces to the cloth to simulate the behavior of wind
- Add a sphere to simulate a piece of cloth falling on a sphere with collision handling

2 IMPLEMENTATION DETAILS

2.1 Hooke's law

Since we are using the model of mass spring, Hooke's law is the core part in calculating the force when the spring is stretched or compressed. Suppose we have two mass points: p and q, the force from q to p accroding to Hooke's Law is

$$F = k(L_0 - ||p - q||) \cdot \frac{p - q}{||p - q||}$$

where k is the coefficient of stiffness, a characteristic of spring, p, q are the Vector3f position of points, L_0 is the zero-force length.
In the code, the Hooke's force computing part is in cloth.cpp ComputeHookeForce(), which is a implement of the above formula.

2.2 Structural, shear, and bending springs

In the spring mass, each mass is affected by its surrounding mass. According to Mass Spring Model defined by Xavier Provot in 1995, the Hooke's force that a mass receives comes from structural, shear, and bending springs. A more intuitive presentation is on the following picture:

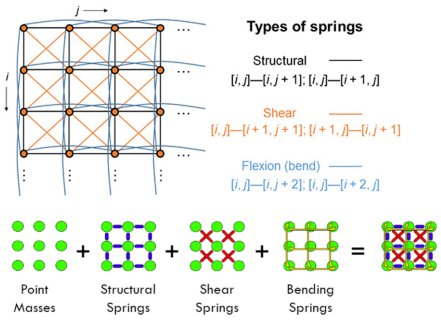

Fig. 1. spring surroundings

My method is to iterate its surrounding mass to compute structural, shear, and bending springs force and add them together in function ComputeSpringForce().

2.3 Fixed points

There are two corners in the cloth that is required to be fixed in world coordinate. For general mass points, after the Spring Force is computed, its acceleration can be calculated, which can help us find its velocity. With the velocity, the mass position in the next time step can be computed and updated in ComputePositions().
However, for fixed points, although they are still affected be force, they should be kept unmoved. So in the step of ComputePositions(), we don not change the position of fixed points to simulate the real world situation that there are other force making the fixed point stable.

2.4 Real-time simulation

In the structure of this project, there are simulation_steps_per_fixed_update_time steps in one display update. Originally, the time doing these number of updates is fixed in 0.02s. However, to make it consistent with real time, we get the real delta time between two mesh update in main loop and transfer this time to cloth.cpp. This collected changing time is used as the time step in velocity and moving distance calculation to ensure Real-time simulation.

2.5 Stable Animation

Considering in real world, everything turns to stable because of the loss of energy. For example, a swinging ball will stop because of resistance to air. In our simulation, since it is too complex to calculate all resistent force in one mesh surface, I directly decrease the velocity by multiply a constant very close to 1 to simulate energy dissipation process. The constant is adjustable to make it look more real.

2.6 Wind Simulation

The wind is a random force added to mass in a specific direciton. Modern C++ random generator uniform_real_distribution is used to control the wind strength. To make a period wind effect, I collect delta time in the real world and generate wind every five seconds.

2.7 sphere collision handling

Collision handling is a difficult job in graphic. In my algorithm, when updating the position of a mass point, I check whether the updated triangle mesh connected to the mass points will intersect of inside of the object. If either two condition above happens, I will get the mass point back to its original value and not update it. The process of checking intersect or inside is done by finding the closest point in the triangle to sphere center and compare it with radius. My algorithm is inefficient but it works in most condition. There remains a condition that my algorithm can not solve, that is when the obeject is too small that my mesh directly go through the object, the collision will not be detected, as described in the following picture:

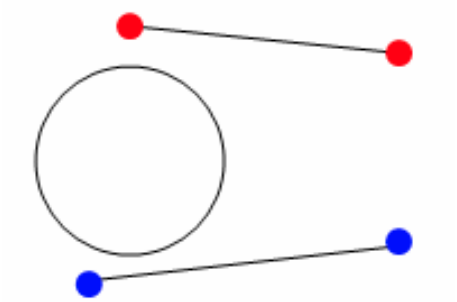

Fig. 2. problem of collision handling

3 RESULTS

The results are pictures following.

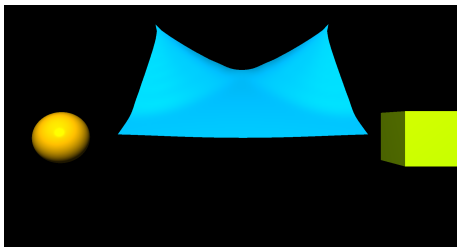

Fig. 3. Falling1

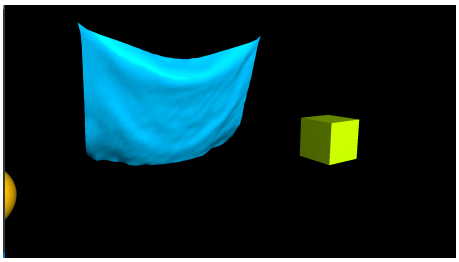

Fig. 4. Falling2

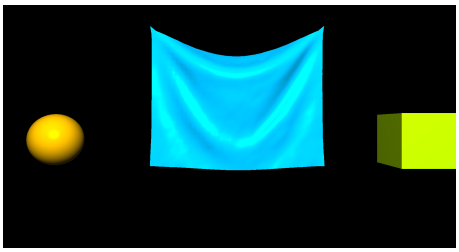

Fig. 5. Stable

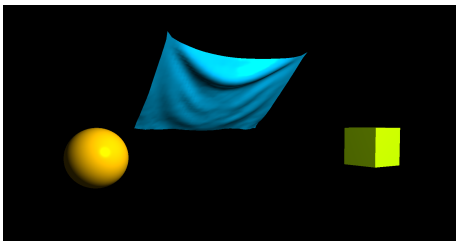

Fig. 6. Wind Simulation

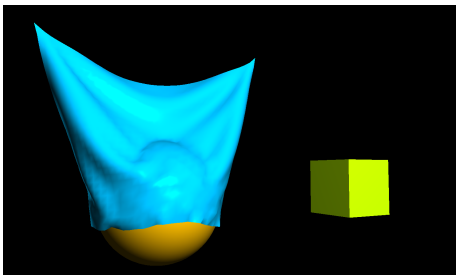

Fig. 7. Sphere Collision