Assignment 5:
Animation with Cloth Simulation

NAME: WEILIANG SUN
STUDENT NUMBER: 2020533010
EMAIL: SUNWL1@SHANGHAITECH.EDU.CN

ACM Reference Format:
Name: WeiLiang Sun, student number: 2020533010, email: sunwl1@shanghaitech.edu.cn. 2022. Assignment 5: Animation with Cloth Simulation. 1, 1 (December 2022), 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This is the report of homework 5 for the computer graphics. In this assignment: we are taught to do the following parts:
(1) Force computation with Hooke's law
(2) Structural, shear, and bending springs
(3) Fix the location of two mesh points to stop the cloth falling down
(4) Real-time and stable animation
Bonus part:
(1) Apply external forces to the cloth to simulate the behavior of wind
(2) Add a sphere or cube obstacle to simulate a piece of cloth falling on a sphere or a cube with collision handling
(3) Drag a mesh point to move the cloth with mouse in real-time

2 IMPLEMENTATION DETAILS

Part 1: Force computation with Hooke's law

In this part, we will calculate the force of the Hooke Force between two meshes by using the Hooke's Law. We are given two masses with the location *iw_this*,*ih_this* and another mass located at *iw_that*,*ih_that*.
According to the Hooke's Law, we get the formula as follows:

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

where p and q are two masses, L_0 is the rest length or named zero-force length. The formula $\frac{p-q}{\|p-q\|}$ stands for the unit vector which is force direction.
First, we deal with some boundary situations: if the location of the two masses is out of our mass dim, we return to the vector $\{0, 0, 0\}$. Then return to the normal situation: we use the function *local_or_world_positions* to get the postion of the two masses, and calculate the final force.

```
Normal = stiffness * (p1_p2 - dx_world *
glm::normalize(p1_p2));
```

Part 2: Structural, shear, and bending springs

In this part, we will calculate the force considering the whole system, including three kinds of forces, structure, shear, and bending. Structure tells us all of the masses are calculated by the four masses right, up, down, left. So we only calculate them one as one as follows:

```
Normal += ComputeHookeForce(iw, ih, iw-1,
 ih, dx_local * structure_x);
Normal += ComputeHookeForce(iw, ih, iw+1,
 ih, dx_local * structure_x);
Normal += ComputeHookeForce(iw, ih, iw,
ih-1, dx_local * structure_y);
Normal += ComputeHookeForce(iw, ih, iw,
ih+1, dx_local * structure_y);
```

Similarly, we can calculate the force generated by shear and bending. The shear tells us that all of the masses are calculated by the four masses right-up,right-down, left-up, left-down. And the bending tells us that all of the masses are calculated by right-two, left-two, up-two, down-two. We simply calculate them one by one and finally return the result Normal.

Part 3: Fix the location of two mesh points to stop the cloth falling down

In this part, we need to deal with the situation that how position of the mesh points change. First we should deal with the coordinate system. While the spring system is calculated in the world coordinate system, the transform matrix MVP is calculated in the local coordinate system. So we must transform the world coordinates to local coordinates and also inverse the transformation part.

The two functions related are: LocalToWorldPositions and WorldToLocalPositions. Since the transform matrix calculated by MVP matrix is given in our function, so we only need to multiply it by the local coordinate to transform to the world position. Meanwhile, in the inverse part, we need to first inverse the MVP matrix given and then multiply it with the world coordinate, then it will turn to the local coordinate.

Author's address: Name: WeiLiang Sun
student number: 2020533010
email: sunwl1@shanghaitech.edu.cn.

XXXX-XXXX/2022/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Of course, the measures we take are not only these two functions, we also need to calculate the speeding formula, which will be showed in the next part.

Part 4: Real-time and stable animation

This part is mostly related to the Part three because it need to deal with the speeding formula. What we need to finish is the three functions as follows: ComputeAccelerations, ComputeVelocities and ComputePositions. We first see if the mass we are dealing with is fixed mass. If it is fixed mass, we won't change it.
Then, when we calculate accelerations, we can use this formula:

$$a = \frac{F}{m} - g$$

What we need to mention is that in this formula we haven't considered the bonus part.

When it comes to velocites, the things are simple: $\Delta_v = a \times \Delta_t$

To calculate the position change, we can use the formula: $\Delta_s = \Delta_v \times \Delta_t$

Bonus Part 1: Apply external forces to the cloth to simulate the behavior of wind

We use wind force minus its speed, then get the minus speed in the direction of the wind and speed. Then we dot it with the normal to put the minus speed on the direction of normal, then multiply with normal, finally get the new normal.

It is obvious that we should apply this fluid force in the force calculating in part four.

Bonus part 2: Add a sphere or cube obstacle to simulate a piece of cloth falling on a sphere or a cube with collision handling

In this part, we will handle the situation that when the cloth intersect with the sphere. First we have a bool function to calculate if the mass is inside the sphere. If it is inside the sphere it returns true otherwise it returns false. Then when intersecting inside the sphere, we make its accelerations zero, its speed zero. And also we renew its position.

How to renew the position? We get the distance of the point and the sphere center, and move it to the outside of the sphere. Then we can get the new position of the points inside the sphere. The code is as follows:

```
1    local_or_world_positions[index] = (
     sphere_r + Float(0.02)) * glm::normalize(
     local_or_world_positions[index] -
     sphere_center) + sphere_center;
```

Bonus Part 3: Drag a mesh point to move the cloth with mouse in real-time

In this part, we will realize the funtion of dragging a mesh point to move the whole cloth which is completed in the camera.cpp function update.

As we have done in the assignment 3 we can get a point's posx, posy, posz through the camera in the near plane. We add all the posx, posy, posz together and we can get a ray generated by the camera. Also the now position is located at posx+posy+posz+camera's position. So after doing this, we can get the generated ray and now position in the near plane.

After this, we suppose a situation: we press the left button but not move the mass, the first task for us is to find the nearest mass point on the cloth plane according to the position we assert by our mouse. In this problem, we first put the length of cloth to camera to our generate ray and then minus them. If the distance is smaller than a specified number, we can get the conclusion that we are pressing this mass point. Of course it is a traverse to find the nearest point.

After that we note the mass point we are pressing, and then renew its fix property. Surely its acceleration and velocities should be zero. Since our location is all on the near plane, we need to turn the scale onto the cloth sphere. We need a scale num to store its similarity scale, and then multiply it with the distance moved on the near plane, we can get its distance change on the cloth plane.