Assignment 5 : Cloth Simulation with the Mass Spring System

STUDENT NAME: XIA KANGJIE STUDENT NO. 2021533071 EMAIL: XIAKJ@SHANGHAITECH.EDU.CN

1 INTRODUCTION

The assignment requires the implementation of a simple mass spring system for simulating a piece of cloth using OpenGL. The goal is to create a cloth simulation that can reach a stable state after a certain period of time. Additionally, there are optional tasks such as applying external forces like wind, adding user interaction, and coupling the cloth with sphere colliders.

And the assignment outlines the following programming requirements:

- Define mass particle properties and initialize particles.
- Define spring properties and create springs connected to particles.
- Implement the mass spring simulation algorithm with gravity, including constraints that fix the left-up and right-up corners of the cloth.
- Ensure that the cloth can reach a stable state after simulation for a certain period of time under the default case.
- Optionally, apply external forces to simulate the effect of wind.
- Optionally, add user interaction to the cloth for interactive control.
- Optionally, couple the cloth with sphere colliders and simulate the cloth falling on a sphere.

2 IMPLEMENTATION DETAILS

2.1 Mass Particle Properties

Each mass particle has been added the following properties:

- 'velocity': A 3D vector ('glm::vec3') representing the velocity of the particle.
- 'mass': A float value representing the mass of the particle.

Since the simulation requires the mass and velocity of the objects to assist in calculations, these properties are necessary.

The velocity is initialized to 0, and the mass is initialized as the total mass divided by the quantity of particles.

2.2 Spring Properties

The following properties have been added to each spring:

- 'stiffness': A float value representing the stiffness of the spring.
- 'restLength': A float value representing the rest length of the spring.

These properties define the behavior of the springs connecting the mass particles. The stiffness determines how resistant the spring is to deformation, while the rest length represents the initial length of the spring when it is in an undeformed state.

These variables allow for fine-tuning the characteristics of the springs in the cloth simulation, enabling control over their elasticity and resting position.

2.3 Simulation Calculation

Step 1: Update particle positions

For each mass particle i, the position update can be represented as:

$$\mathbf{p}_i' = \mathbf{p}_i + \mathbf{v}_i \cdot \Delta t$$

where $\mathbf{p}_i$ represents the current position of particle i, $\mathbf{v}_i$ is the velocity of particle i, and Δt is the time step.

Step 2: Update springs

For each spring j connecting particles m and n, the spring force and subsequent velocity update for particle m can be expressed as:

$$\mathbf{F} = k \cdot (\|\mathbf{p}_n - \mathbf{p}_m\| - L_j) \cdot \frac{\mathbf{p}_n - \mathbf{p}_m}{\|\mathbf{p}_n - \mathbf{p}_m\|}$$

where k represents the stiffness coefficient of the spring, L_j is the origin length of the spring j, and $\|\cdot\|$ denotes the Euclidean norm.

Notice that our springs have a specific direction. Therefore, when calculating forces, we only need to compute forces along that direction.

The velocity update for particle m can be written as:

$$\mathbf{v}_m' = \mathbf{v}_m + \frac{\mathbf{F}_j}{m_m} \cdot \Delta t$$

where m_m is the mass of particle m.

Step 3: Apply gravity and air resistance

For each mass particle i, the velocity update considering gravity and air resistance can be represented as:

$$\mathbf{v}_i' = \mathbf{v}_i + \mathbf{g} \cdot \Delta t - \mathbf{v}_i \cdot \frac{\|\mathbf{v}_i\| \cdot \rho \cdot A}{m_i} \cdot \Delta t$$

where $\mathbf{g}$ represents the gravitational acceleration, ρ is the air density, and A is the effective cross-sectional area of the particle.

In the code, 'airResistanceCoefficient' represents the product of ρ and A.

Additionally, the wind effect is applied to the velocity update as:

$$\mathbf{v}_i' = \mathbf{v}_i' + c_w \cdot \mathbf{d}_w \cdot \Delta t$$

where c_w is the wind coefficient controlling the intensity of the wind effect, and $\mathbf{d}_w$ specifies the direction in which the wind blows.

Step 4: Apply constraints

The constraints are applied to fix the positions and velocities of specific particles. For example, fixing the left-up corner particle can be expressed as:

$$\mathbf{p}_{\text{left\_up}}' = \mathbf{p}_{\text{left\_up\_initial}}$$

$$\mathbf{v}_{\text{left\_up}}' = \mathbf{0}$$

where $\mathbf{p}_{\text{left\_up}}$ and $\mathbf{v}_{\text{left\_up}}$ represent the current position and velocity of the left-up corner particle, respectively, and $\mathbf{p}_{\text{left\_up\_initial}}$ represents the initial position of the left-up corner particle.

2.4 Wind Simulation

In addition to the existing properties, the following variables have been added to enable wind simulation:

- 'windCoefficient': A float value representing the coefficient of the wind effect on the cloth.
- 'windDirection': A 3D normalized vector ('glm::vec3') representing the direction of the wind.

These variables allow for the optional inclusion of wind effects in the cloth simulation. The 'windCoefficient' controls the intensity of the wind effect, while the 'windDirection' specifies the direction in which the wind blows.

And we update the mass particles' velocity by the formula

$$v' = v + c\mathbf{d} \cdot \frac{|\cos\theta| dA}{dm} \Delta t$$

where v' is the updated velocity of the mass particle, v is the current velocity of the mass particle, c is the wind coefficient, $\mathbf{d}$ is the wind direction vector($|\mathbf{d}|$ is the wind strength), dA, dm are the particle area and mass of particle objects, θ is the angle between the normal direction $\mathbf{n}$ and the dind dirction, and Δt is the time step.

The formula describes how the wind affects the velocity of the mass particles in the cloth simulation. By adding the wind effect to the current velocity, the particles are influenced by the wind's intensity and direction, resulting in a more realistic cloth simulation.

2.5 Interaction

To enable interaction between the mouse and objects, we need to calculate the direction pointed by the mouse. Therefore, I added an additional function in 'camera.hpp' to retrieve the mouse ray. Then, we iterate through all the mass particles to find the closest one to the mouse ray and fix its distance at that moment, positioning it at the location pointed by the mouse.

In mathematical notation, the translation would be as follows:

$$\begin{cases} p' = o + d \cdot \mathbf{dir} \\ v' = 0 \end{cases}$$

Where p' is the updated position, o is the camera position, d is the distance, $\mathbf{dir}$ is the direction vector, v' is the updated velocity

3 RESULTS

Fig. 1. normal simulation

Fig. 2. normal simulation

Fig. 3. normal simulation

Fig. 5. interactive simulation

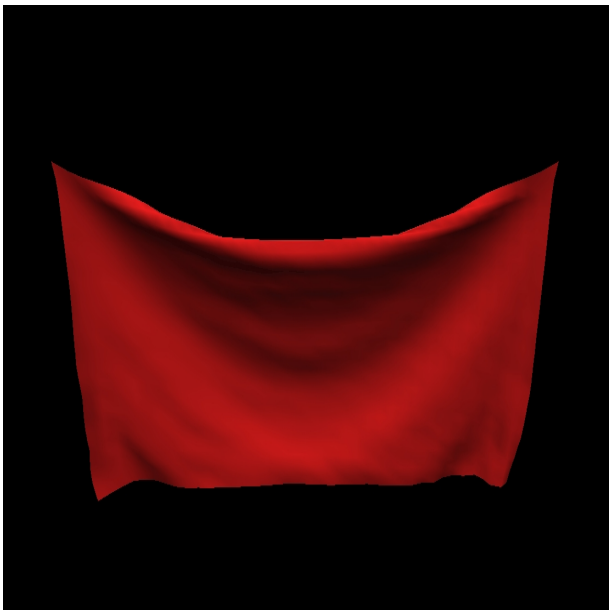

Fig. 4. wind simulation

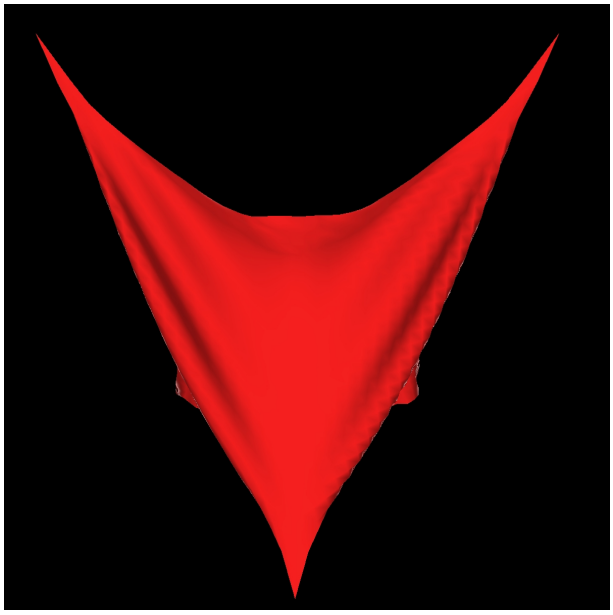

Fig. 6. interactive simulation

4 CONCLUSION

In conclusion, the implementation of the mass spring system for simulating a piece of cloth using OpenGL has been successfully achieved. The cloth simulation was able to reach a stable state after a certain period of time under the default case.

The mass particle properties, including velocity and mass, were defined and initialized to facilitate the calculations in the simulation.

The spring properties, such as stiffness and rest length, were also implemented to control the behavior of the springs connecting the particles.

The simulation algorithm incorporated gravity, constraints, and optional wind effects. The gravity and air resistance were applied to update the particles' velocities, while constraints were used to fix the positions and velocities of specific particles.

Moreover, the optional wind simulation was implemented, allowing for the application of wind forces to the cloth. The wind coefficient and direction could be adjusted to control the intensity and direction of the wind effect.

Additionally, user interaction was enabled through mouse interaction. The direction pointed by the mouse was calculated, and the closest mass particle to the mouse ray was fixed at that moment, following the mouse's movement.

The results of the cloth simulation demonstrated the successful implementation of the system. The stability of the cloth, the wind effect, and the interaction with the mouse were reflected in the rendered images.

Overall, the implementation fulfilled the requirements of the assignment, providing a functional mass spring system for cloth simulation using OpenGL. The optional tasks of wind simulation and user interaction were successfully accomplished, further enhancing the realism and interactivity of the simulation.