CG_8_Mass Spring Report

PB21010479 Caoliwen Wang

April 29, 2024

1 Problem Statement

This assignment involves simulation, specifically using a spring-mass system. The spring-mass system is a fundamental technique in elastic body simulation. Due to its ease of implementation and good results, it has been widely used in applications such as gaming for simulating hair, cloth, and elastic bodies. It has also inspired many subsequent methods for elastic body simulation.

A spring-mass system is essentially a graph composed of nodes and edges between them. Each node of the graph represents a mass, and each edge represents a spring. It discretizes the continuous body. The grid can be a 2D grid, used for simulating objects like cloth or paper, as shown in the figure below. It can also be a 3D grid used for simulating volumetric objects.

2 Propose Algorithms

2.1 Semi-Implicit Method

To get the discrete motion of these points over time, we need to discretize time. Suppose we have n vertices, and we arrange all vertices into a matrix $\mathbf{x} \in \mathbb{R}^{3n\times 1}$.

To initiate the movement of the object, one approach is to assign a velocity to each vertex. If we use semi-implicit time integration, the velocity $\mathbf{v}^{n+1} \in \mathbb{R}^{3n\times 1}$ can be determined as follows, according to Newton's second law:

$$\mathbf{v}^{n+1} = \mathbf{v}^n + h\mathbf{M}^{-1}(\mathbf{f}_{\text{int}}(\mathbf{x}^n) + \mathbf{f}_{\text{ext}})$$

Here, $\mathbf{M} \in \mathbb{R}^{3n\times 3n}$ is the system's mass matrix. Here, we simply set it as a diagonal matrix.

According to the energy perspective, let E be the elastic energy of the system. The internal elastic force $\mathbf{f}_{\text{int}}$ can be defined as:

$$\mathbf{f}_{\text{int}} = -\nabla E$$

If we have an expression for the energy gradient, we can calculate the internal elastic force. By adding external forces such as gravity, we can obtain the total force acting on the vertices. Then, we can initiate the movement of the object!

Now, let's define the energy of a spring i as:

$$E_i = \frac{k}{2}(\|\mathbf{x}_{i1} - \mathbf{x}_{i2}\| - L)^2$$

We define $\mathbf{x}_i := \mathbf{x}_{i1} - \mathbf{x}_{i2}$. Then, the total energy is:

$$E = \sum_i E_i = \sum_i \frac{k}{2} (\|\mathbf{x}_i\| - L)^2$$

Its gradient can be calculated as:

$$\nabla E = \sum_i k(\|\mathbf{x}_i\| - L)\frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}$$

Finally, to simulate damping, we can multiply the velocity by a damping coefficient vel $\times$ damping.

Here, we need to fix some points (Dirichlet boundary conditions). Then, we can simply set the external forces and velocities of these fixed points to zero.

2.2 Implicit Euler Integration

Next, we need to implement implicit Euler integration:

$$\begin{aligned}\mathbf{x}^{n+1} &= \mathbf{x}^n + h\mathbf{v}^{n+1} \\ \mathbf{v}^{n+1} &= \mathbf{v}^n + h\mathbf{M}^{-1}(\mathbf{f}_{\text{int}}(\mathbf{x}^{n+1}) + \mathbf{f}_{\text{ext}})\end{aligned}$$

But we find that this problem needs to be turned into an equation about $\mathbf{x}^{n+1}$.

Here is a commonly used approach to solve it.

Rearranging:

$$\mathbf{x}^{n+1} = \mathbf{x}^n + h\mathbf{v}^n + h^2\mathbf{M}^{-1}(-\nabla E(\mathbf{x}^{n+1}) + \mathbf{f}_{\text{ext}}) \tag{4}$$

Let's define $\mathbf{y} := \mathbf{x}^n + h\mathbf{v}^n + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$.

We can transform equation (4) into:

$$\frac{1}{h^2}\mathbf{M}(\mathbf{x}^{n+1} - \mathbf{y}) + \nabla E(\mathbf{x}^{n+1}) = \mathbf{0}$$

This equation can be viewed as the first-order optimality condition (KKT) for an optimization problem (let $\mathbf{x} = \mathbf{x}^{n+1} \in \mathbf{R}^{3n\times 1}$):

$$\min_{\mathbf{x}} \quad g(\mathbf{x}) = \frac{1}{2h^2}(\mathbf{x}-\mathbf{y})^\top \mathbf{M}(\mathbf{x}-\mathbf{y}) + E(\mathbf{x}) \tag{5}$$

This brings us back to the optimization field that you may have (probably) studied.

The derivative of energy g is:

$$\nabla g(\mathbf{x}) = \frac{1}{h^2}\mathbf{M}(\mathbf{x}-\mathbf{y}) + \nabla E(\mathbf{x})$$

To solve the optimization problem, we can use gradient descent, but its convergence speed is relatively slow (linear convergence rate). In computer graphics, a more commonly used approach is to use Newton's method:

$$\mathbf{x}^{n+1} = \mathbf{x}^n - \mathbf{H}^{-1}\nabla \mathbf{g}$$

So we need to compute the Hessian matrix $\mathbf{H} = \nabla^2 g$ of energy g.

First, let's look at the Hessian of a spring's energy:

$$\begin{aligned}\mathbf{H}_i &= \nabla^2 E_i \\ &= k\frac{\mathbf{x}_i \mathbf{x}_i^\top}{\|\mathbf{x}_i\|^2} + k\left(1 - \frac{L}{\|\mathbf{x}_i\|}\right)\left(\mathbf{I} - \frac{\mathbf{x}_i \mathbf{x}_i^{\mathrm{T}}}{\|\mathbf{x}_i\|^2}\right)\end{aligned}$$

So the overall Hessian $\mathbf{H} \in \mathbf{R}^{3n\times 3n}$ is assembled by stacking the Hessian of each spring $\mathbf{H} \in \mathbf{R}^{3\times 3}$ according to vertex indices. The def of H is shown in Figure 1.

$$\mathbf{H(x)} = \sum_{e=\{i,j\}} \begin{bmatrix} \frac{\partial^2 E_e}{\partial \mathbf{x}_i^2} & \frac{\partial^2 E_e}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \\ \frac{\partial^2 E_e}{\partial \mathbf{x}_i \partial \mathbf{x}_j} & \frac{\partial^2 E_e}{\partial \mathbf{x}_j^2} \end{bmatrix} = \sum_{e=\{i,j\}} \begin{bmatrix} \mathbf{H}_e & -\mathbf{H}_e \\ -\mathbf{H}_e & \mathbf{H}_e \end{bmatrix}$$

Figure 1: def of H

Finally, we write out the Hessian of g:

$$\nabla^2 g = \frac{1}{h^2}\mathbf{M} + \mathbf{H}$$

Now, we can use Newton's method for optimization:

$$\mathbf{x}^{n+1} = \mathbf{x}^n - \mathbf{H}^{-1}\nabla g(\mathbf{x}^n)$$

Here, it actually involves a part of Line Search. The general process for optimizing a problem based on line search methods is: 1. Determine the search direction $\mathbf{p}$ (here, $\mathbf{p} = \mathbf{H}^{-1}\nabla g$), 2. Then determine the step size α to advance (this step is called Line Search), 3. Finally, update $\mathbf{x}^{n+1} = \mathbf{x}^n - \alpha\mathbf{p}$. Since the recommended step size for Newton's method is 1, we will not perform additional Line Search here.

$\mathbf{H}$ is a sparse matrix (only neighboring vertices will have corresponding non-zero elements in the matrix), and we use 'Eigen::SparseMatrix' to store it.

If we want to fix points during the solving process, simply setting the fixed points back to their original positions afterward may lead to excessive stretching of the boundary region. The root of the problem lies in considering hard constraints during the solving process.

We can formulate the problem as a constrained optimization problem:

$$\min_{\mathbf{x}} \quad g(\mathbf{x}) = \frac{1}{2h^2}(\mathbf{x}-\mathbf{y})^\top \mathbf{M}(\mathbf{x}-\mathbf{y}) + E(\mathbf{x}) \text{s.t.} \quad c(\mathbf{x}) = \mathbf{S}\mathbf{x} = \mathbf{0}$$

In a broad sense, this is indeed a constrained optimization problem. However, we don't necessarily need to use the method of Lagrange multipliers to solve it.

Instead, we can follow the approach used in Homework 3 for handling boundary conditions: modify the Hessian matrix while solving the equations, setting the coefficients corresponding to the fixed points to 1. Alternatively, we can use the transformation $\mathbf{H}^{\text{new}} = \mathbf{S}^T\mathbf{H}\mathbf{S}$ to obtain a smaller matrix, where $\mathbf{S}$ is the selection matrix.

2.3 Interactions

Simulating collision and friction is also an important topic in computer graphics. If not handled properly, it can lead to the common "popping" issue in games.

For this assignment, we can also consider the collision between a spring-mass system and a sphere. We can even make this sphere move.

We can use a simple penalty-based contact force as follows:

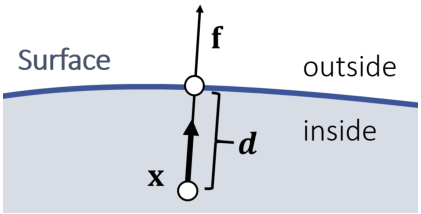

Figure 2: Principle of collision

$$\mathbf{f} = k^{\text{penalty}} \max(s \cdot r - \|\mathbf{x}-\mathbf{c}\|, 0) \frac{(\mathbf{x}-\mathbf{c})}{\|\mathbf{x}-\mathbf{c}\|}$$

Here, $\mathbf{c}$ is the center of the sphere, r is the radius of the sphere, s is a magnification factor of the radius (e.g., let $s = 1.1$), which is used to generate contact force even when actual contact has not occurred to reduce visual penetration (denoted as 'collision_scale_factor' in the program), and k^{penalty} is a tunable parameter.

3 Programming

3.1 Node programming

In this assignment,The node used in this instance is very simple. We adopted the connection method as shown in Figure 3.

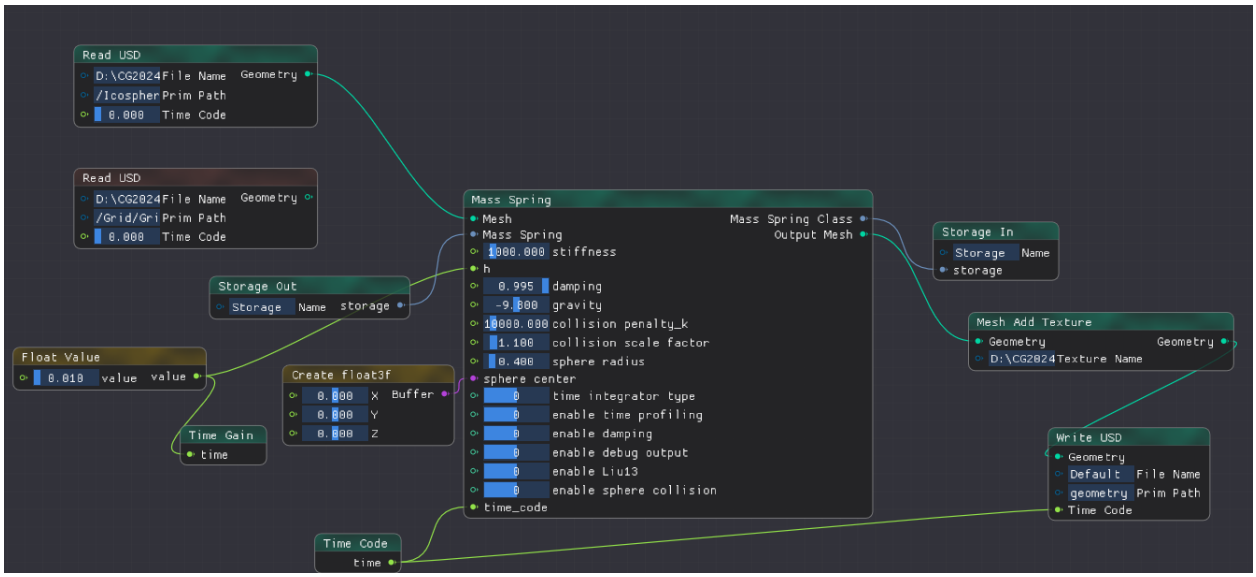

Figure 3: Geometry Nodes

4 Experimental Results

I think the results of this algorithm are fantastic.

We first use the semi-implicit integration method, and the results obtained in this way do explode. The result is as shown in the Figure 4.

Record the video as shown in 1.

Afterward, we use the implicit method, and the resulting video is as shown in 2.

We can observe that the results are still relatively realistic.

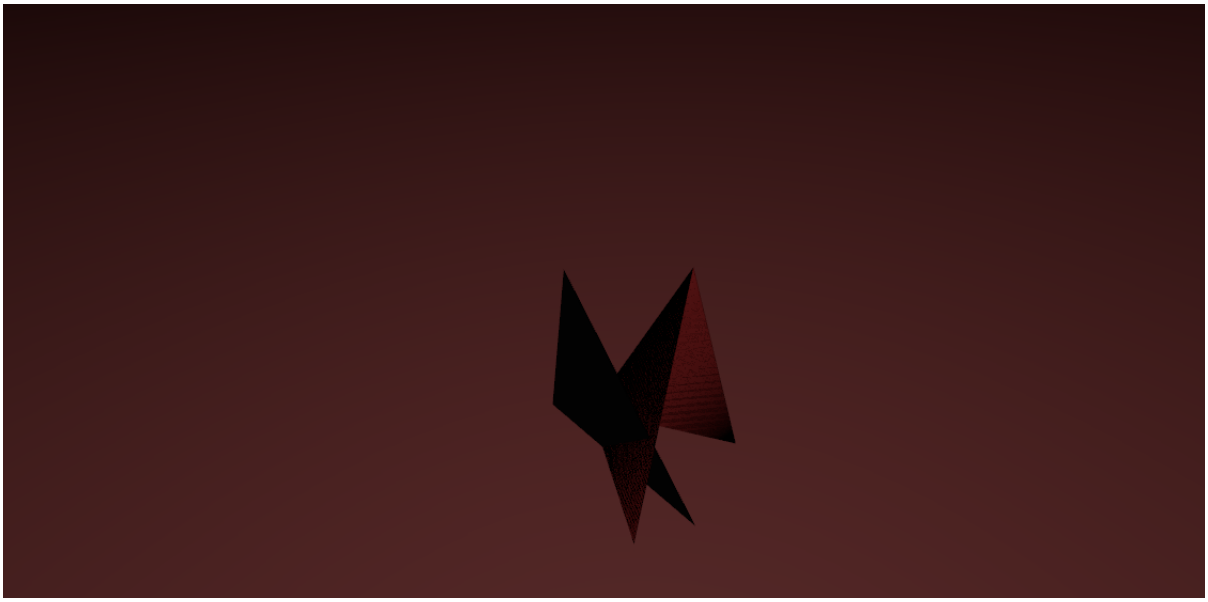

Figure 4: Semi-Implicit Method Outcomes

5 Summary

In this experiment, we successfully implemented a simulation system using a spring-mass model, and overall, the results were satisfactory. However, when I introduced external forces, errors occurred in the solution after running for a while. This is because we did not handle issues such as matrix definiteness well, which requires modification.