Robust Cloth Simulation with Constraints and Collision Detection

Zhiguo Zhang
McGill University
zhiguo.zhang@mail.mcgill.ca

ABSTRACT

For this project, I implemented a 3D cloth simulator that handles collisions and constraints using a Backward Euler integrator. The simulator is tested using a rectangular cloth, an icosphere representing a obstacle meshes, and a spherical constraint. This simulator generally produce a good visual effects of cloth-obstacle collisions and cloth motions.

1 INTRODUCTION

For this project, I extended the idea of particle-spring system, constraints and collision detection into 3D by implementing a cloth simulator in 3D. The code from assignment 1 and 2 forms the backbone of this cloth simulator. The first step is extending the existed functionalities into 3D. Then, using a Backward Euler integrator, I can update the position and velocity of each particle in each time step. Using an explicit integrator generally produces an unstable simulation for large time steps. Since the system in cloth simulation usually contains hundreds of particles, it is impractical to use small time step. Collision and cloth self-penetration can happen during cloth simulation. Then, the next step in this project is the implementation of robust collision detection and response. Lastly, I incorporated constraints into my cloth simulator. The later sections discuss the concepts and implementations in detail.

2 RELATED WORK

Baraff and Witkin [1998] provide a very clear problem formulation for cloth simulation. They show a cloth system that can take large time steps and new technique for handling constraints using implicit method. Bridson et al. [2002] describe a robust way to detect and resolve contacts and collision in cloth simulation. They also introduce friction force that makes cloth simulation visually more accurate. Later work by Goldenthal et al. [2007] discussed methods such as combining relaxation of strain and adopting simple iterative strain algorithms, and Constrained Lagrangian Mechanics. This project focus on and realize some of the ideas from those previous works.

3 MODELING

Particles, springs and triangles are the primitives for building different objects in this project. Triangles can represent a planar surface of a object. The Triangle class uses 3 particles to build a triangle. We can use 3 points in space to compute the normal vector of a triangle and find the distance between an arbitrary point to the triangle. With these building blocks, we can formulate our cloth grid and icosahedron sphere objects in simulation.

COMP559'W18, July 2018, Montreal, Quebec, Canada

3.1 Cloth Grid Modeling

Firstly, I formulate the cloth grid using a $N \times N$ grid. Then, I connect each vertex with its 8 nearest neighbors. Fig. 1 shows a sample cloth grid. However, I only form 2 triangles for every 4 adjacent particles,

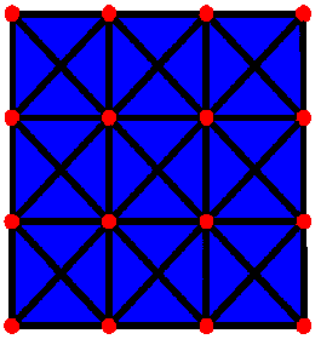

Figure 1: Mass-Spring Model for Cloth. Source: https://graphics.stanford.edu/~mdfisher/cloth.html

namely the top-right and bottom-left triangles. The reason not to form 4 triangles is that the mass is accumulated at the particle at each corner.

3.2 Sphere Mesh Modeling

For the sphere meshes, Cajaraville [2018] shows that icosahedron sphere gives the smallest relative error with the actual sphere. In order to create a icosahedron sphere, I start with 3 rectangles in space and take the 12 vertex as the initial vertex of the sphere, then, keep refining the mesh until the sphere is vivid enough. Fig. 2 to Fig. 5 illustrates how to construct a icosahedron sphere.

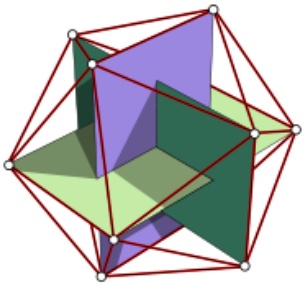

Figure 2: Start with 3 rectangles. Source: http://blog.andreaskahler.com/2009/06/creating-icosphere-mesh-in-code.html

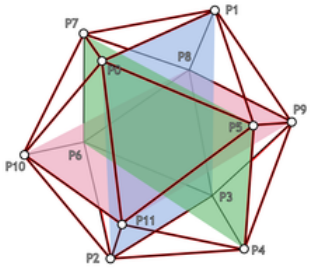

Figure 3: Find the vertex. Source: http://blog.andreaskahler.com/2009/06/creating-icosphere-mesh-in-code.html

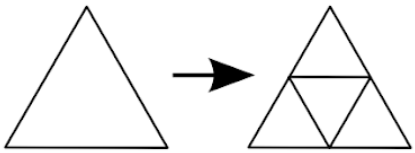

Figure 4: Refinement. Source: http://blog.andreaskahler.com/2009/06/creating-icosphere-mesh-in-code.html

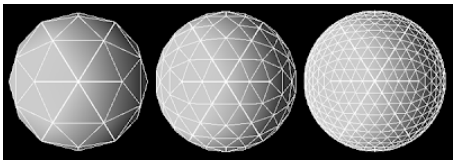

Figure 5: Icosahedron Sphere for level 1, 2, 3 refinements. Source: http://blog.andreaskahler.com/2009/06/creating-icosphere-mesh-in-code.html

Since the number of triangles increases dramatically, 3 level of refinements can produce a visually appealing sphere.

4 IMPLICIT INTEGRATION

Given current position $x(t_0)$ and velocity $v(t_0)$, we need to the new position $x(t_0+h)$ and velocity $v(t_0+h)$ after a time step of h. We can solve the problem by solving the following first order differential equation.

$$\frac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} v \\ M^{-1}f(x,v) \end{pmatrix}$$

As shown by Baraff and Witkin [1998], by using Taylor series expansion to f, this is equivalent to solve

$$\left(I - hM^{-1}\frac{\partial f}{\partial v} - h^2M^{-1}\frac{\partial f}{\partial x}\right)\Delta v = hM^{-1}\left(f_0 + h\frac{\partial f}{\partial x}v_0\right)$$

for Δv. Then, using Δv, we can get $\Delta x = h(v_0 + \Delta v)$. To solve the system, we need to compute f, $\frac{\partial f}{\partial v}$, and $\frac{\partial f}{\partial x}$. Spring class has the necessary functionalities to compute these quantities. It remains to solve the linear system $Ax = b$. Baraff and Witkin [1998] use a mass modification to enforce constraints and modified conjugate gradient method for faster convergence. However, in this project, I have tried to implement mass modification and modified conjugate gradient method but failed. The main reason of failure is that the filter does not have the expected behavior and the system is too complicate to debug. Due to the limited time for this project, I gave up this build and used the original CG solver. What I have not tried is preconditioning on the matrix A to facilitate the convergence.

5 CONTACT AND COLLISION

Contacts and collisions make computer animations vivider. There are two phases in handling contacts and collision, contact detection and collision response. Detection focus more on proximity between meshes and the system before contact. This prevents many visual artifacts from happening such as cloth inter-penetration. Collision response shows what will happen if a collision happens. That is collision response focus more on the effect after collision.

5.1 Detection

There are two possible cases where contacts occur, namely particle-triangle contacts and edge-edge contact. For particle-triangle contacts, it can happen only if the particle and triangle are coplanar and particle falls into the face of triangle, i.e. in terms of barycentric coordinates $u, v, w \in [0,1]$ with $u + v + w = 1$. As described by Bridson et al. [2002], the simulator checks the coplanar condition by solving triple product

$$(x_{21} + tv_{21}) \times (x_{31} + tv_{31}) \cdot (x_{41} + tv_{41}) = 0$$

If $t \in [0,h]$, the simulator registers a collision that needs collision response. A small rounding error is added to boost robustness.

For edge-edge contacts, the simulator checks the coplanar condition in the same way as before, since it only needs the information of 4 particles to perform detection. Then, the simulator uses edge information to check if two edges intersect.

5.2 Response

After detected a possible contact in time interval $[0,h)$, the simulator resolves the collision by adding adjusted impulse response and adjusted repulsion to the 4 particles. We can compute the impulse using the relative velocity at contact point before collision, restitution coefficient, contact normal and mass of each particle. We compute repulsion as described by Bridson et al. [2002]. We compute the adjusted impulse and adjusted repulsion by the same scheme. For particle-triangle contact with triangle $x_1x_2x_3$ and particle x_4, if the barycentric coordinates $w1, w2, w3$ intercept with particle x_4, then we compute

$$I' = \frac{2I}{1 + w_1^2 + w_2^2 + w_3^2}$$

$$v_i^{new} = v_i + w_i(I'/m)n \qquad i = 1,2,3$$

$$v_4^{new} = v_4 - (I'/m)n$$

where I is impulse or repulsion, n is contact normal, m is the mass of particle.

For edge-edge contact with edges x_1x_2 and x_3x_4, if the relative position a on x_1x_2 intercepts with the relative position b on x_3x_4, then we compute

$$I' = \frac{2I}{a^2 + (1-a)^2 + b^2 + (1-b)^2}$$

$$v_1^{new} = v_1 + (1-a)(I'/m)n \qquad v_2^{new} = v_2 + a(I'/m)n$$
$$v_3^{new} = v_3 - (1-b)(I'/m)n \qquad v_4^{new} = v_4 - b(I'/m)n$$

We can also compute friction using Coulomb's model and update tangential velocity in the event of collision. However, the simulator does not have this functionality due to limited time.

6 RESULTS

The demo video shows 2 simulations one with two corners of the cloth pinned and the other with a free falling cloth. The level of detail for both cloth and icosahedron sphere determines the speed of simulation. Generally speaking, with larger than 20×20 cloth grid and sphere more than 3 refinement level, the simulation is extremely slow. The rate determining step in simulation is the continuous collision detection step. A small blue sphere is added to the simulation representing a spherical constraint. The method used to enforce this constraint is using penalty force.

The demo video also shows the unshaded version of the same simulation, which is more clear in mass-spring point of view. Extra features like flat shading also implemented but not shown in the demo video. Flat shading is very useful during debugging.

7 CONCLUSION

This project has extended the content in the first two assignments. The results look very realistic based on my own inspection. Expanding 2D to 3D on basic concepts are pretty easy to deal. However, what is unexpectedly hard when expanding to 3D is actually the rendering. I have to compute normal vector for every vertex and triangle in the system and then render each vertex and each triangle. Computer graphics assignments help me a lot to learn how to do OpenGL rendering. I also tried to use explicit solvers, however, they are unstable even for a small time step with hundreds of particles. Resolving the collision is the second most time consuming part in this project. Basically, I have to redo all the work that I have done in 2D previously. The 3D geometry is more complex than 2D. Another aspect that I have considered but not yet implemented is acceleration techniques for the continuous collision detection process. Overall, this is a fun project. I am pleased with the simulations I have produced.

REFERENCES

Baraff, D. and Witkin, A. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. New York, NY, USA, 43–54. http://doi.acm.org/10.1145/280814.280821

Bridson, R., Fedkiw, R., and Anderson, J. 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*. New York, NY, USA, 594–603. http://doi.acm.org/10.1145/566570.566623

Cajaraville. 2018. 4 Ways to Create a Mesh for a Sphere. https://github.com/caosdoar/spheres. Accessed: 2018-04-04.

Goldenthal, R., Harmon, D., Fattal, R., Bercovier, M., and Grinspun, E. 2007. Efficient Simulation of Inextensible Cloth. In *ACM SIGGRAPH 2007 Papers (SIGGRAPH '07)*. ACM, New York, NY, USA, Article 49. https://doi.org/10.1145/1275808.1276438