Cloth simulation using mass-spring technique

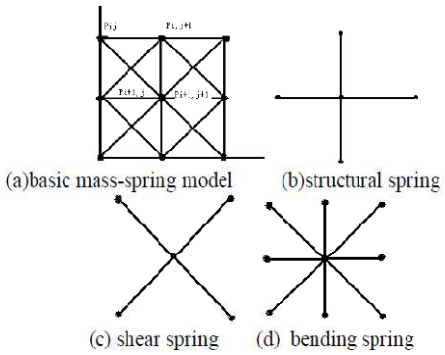
Introduction

Cloth simulation has been a very popular topic of research for many years; having proper looking cloth that reacts to forces and character movements adds realism to games and movies. Many methods for simulating cloth have been investigated upon and widely used throughout the years; from physically correct but complex energy models to simple and fast constraint-based mass spring models. The compromise is always between the complexity of the model, and how fast and reactive it is in interactive situations. The game industry is nowadays a big consumer of cloth simulations, given the constant increase in hardware power that can accommodate many more complex simulation models.

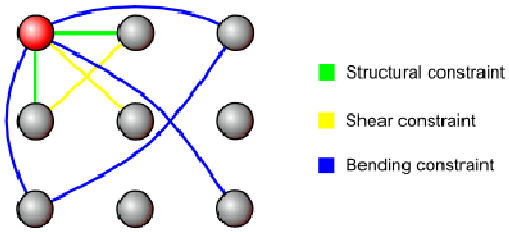
Among all these, the mass spring model is the most widely used, because it is very easy to implement, easy to add in external forces and improve the simulation and it executes quite fast. Although it is quite far from being exact, it does give out some very impressive results.

The Spring Mass Model

This model uses the concept of masses being connected together in a grid style through springs. The model uses 3 kinds of springs, namely, stretch, shear and bend. Each mass is connected to its neighbours through these springs and varying the properties of the latter gives different kinds of cloth behaviour.



(Yu-feng et al., 2010)



(Computer Graphics Labat the Alexandra Institute, 2010)

Thus, the simulation of the cloth comes down to realistic movements of the masses subjected to forces, which can be classified into 2 categories, internal and external.

The internal force arises due to tensions in the connected springs and is determined by Hooke's law of elasticity, as follows:

$$F = -kx$$

where k is the spring constant and x is the displacement in the spring from its rest length. The force F is a restoring one and, thus, has a negative magnitude.

Another component of the internal force is damping, that is essential in a spring system to bring the system back to rest state, without which the springs oscillate to infinity.

The damping force acts against the velocity of the masses to slow them down until they come back to rest. It is given by the following formula:

$$F = -cv$$

Where c is the damping coefficient and v is the velocity of the particle.

The final spring force applied on each particle can, thus, be given by the following:

$$F = -kx - cv$$

External forces vary from simulation to simulation, but basically consist of gravity and wind. Gravity can be easily calculated using Newton's second law of motion, as follows:

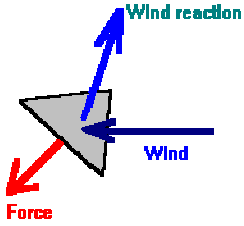
$$F = mg$$

where m is the mass and g is the constant of freefall acceleration.

Wind usually acts on faces instead of individual particles; as such, it uses the normal at the points to affect the face and is given by the following formula:

$$F = c*(n.d)*n$$

Where c is the wind strength, d is the wind direction and n is the normal of the face at the point.



(Windclth, 2011)

The adding of all these forces will eventually give rise to motion in the particles, any of which will add more tension in the springs, kicking in Hooke's law and giving rise to expansion and contractions of the particles to create movement in the cloth.

Integration

Once we calculate the net force on each particle, the next step is to calculate the velocity and the position to which the particle will move, under the action of the force. The acceleration can be calculated using Newton's second law of motion,

$$a = F / m$$

Where m is the mass of the particle and F is the net force on the particle.

The velocity is derived from the acceleration using a first order derivation from the acceleration. The displacement is obtained by a second order derivation from the acceleration, as follows:

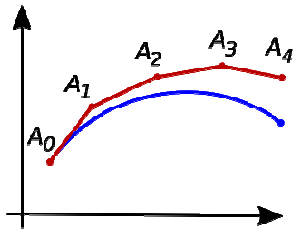
$$v = d(a) / dt, \text{ and } x = d(v) / dt$$

Determining the next velocities and positions from the current one leads into the field of numerical analysis and it comes down to solving ordinary differential equations. Many integration methods are available, ranging from fast, easy but very unstable methods such as the Eular explicit, to complex and time consuming ones such as the Eular backward method. Once again, it's a compromise between computation time and simplicity and accuracy.

The Eular explicit integration method uses the following formulae:

$$\mathbf{x}' = \mathbf{x} + \mathbf{v} \cdot \Delta t$$
$$\mathbf{v}' = \mathbf{v} + \mathbf{a} \cdot \Delta t,$$

Eular suffers from stability problem with big time steps, because it diverges gradually away from the actual result (BlueThen, 2011); while keeping the time step tiny helps, it increases the amount of calculations done per frame.



Verlet integration is another method that has emerged lately to be very fast and more stable than Eular Explicit because it bypasses the calculation of velocity and derives the displacement directly from the acceleration. The following is its formulae:

$$\mathbf{x}' = 2\mathbf{x} - \mathbf{x}^* + \mathbf{a} \cdot \Delta t^2$$
$$\mathbf{x}^* = \mathbf{x}$$

Other more stable methods are used, such as RK4 and backward Eular, but these are computationally expensive and not simple to implement.

Implementation

1. Simulation Algorithm

The following is the algorithm for a single step through the cloth simulation

Evaluate Force

Foreach particle, add in gravity
Foreach triangle in the cloth plane, calculate and add in wind
Foreach spring, calculate and add in spring force to particles

Integrate using Eular-Forward or Verlet for next velocity and position

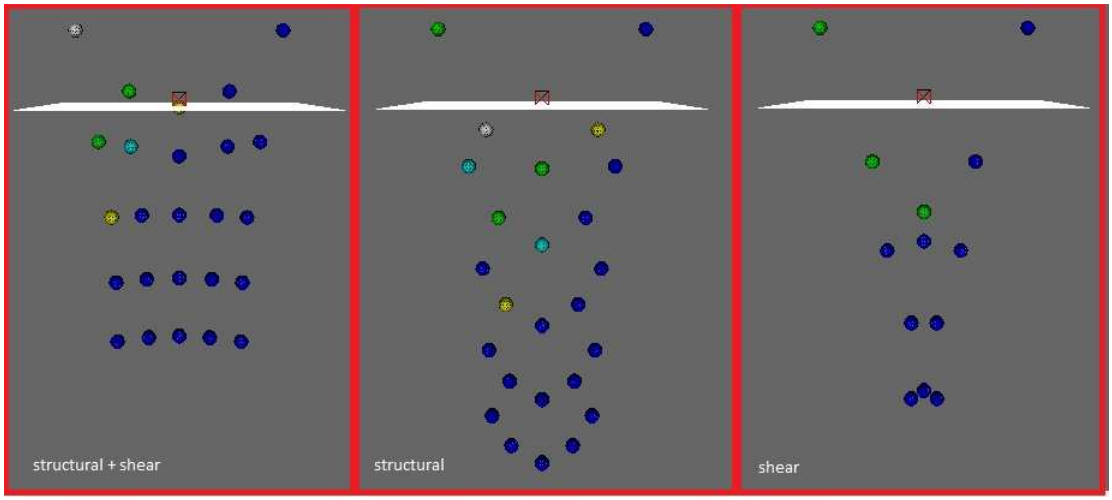
Check Collision

If collision exists, resolve for a new position and integrate again

Draw particles

2. Initial Implementation test

Initially, I setup the particles in a grid fashion, added the 3 kinds of springs and did some preliminary tests to assess the effects of each of them. The following is a screenshot of my preliminary tests:

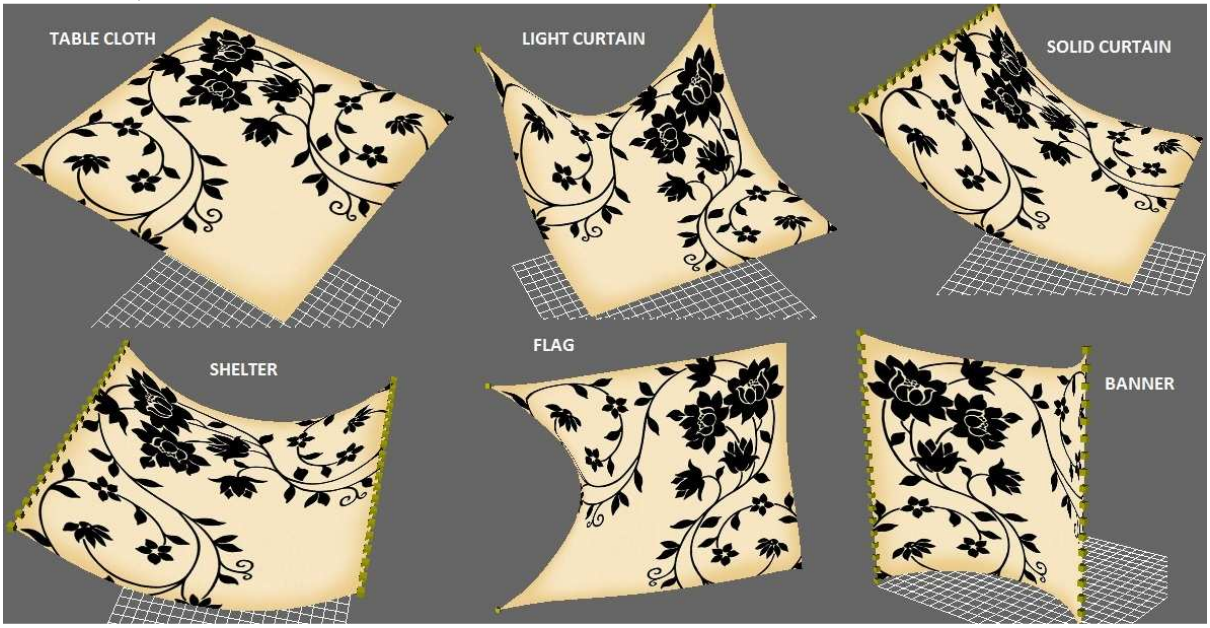


3. Pinning and constraints

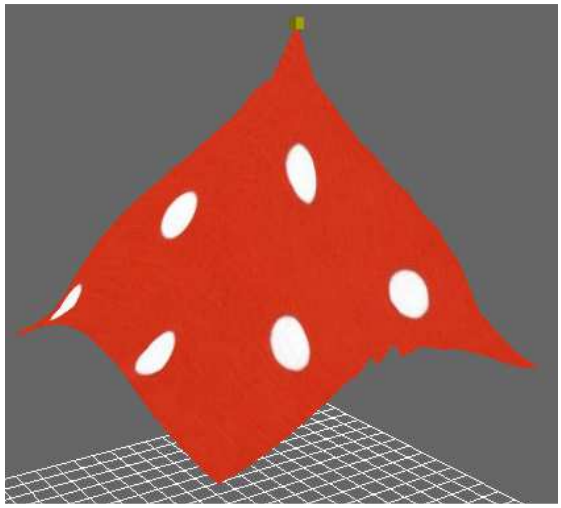
I have created the Particle class to store information about a spefic particle in the simulation. Each particle has properties like mass, position, velocity and also a property called “moveable” which is set to false initially.

Different options (tablecloth, flag, and etc) allow the modification of that property for a specific particle or a group of them (row or column) to make it moveable or not. This property is then used in the “EvaluateForce” and “Integrate” method, whereby the particle movement is ignored if it is not moveable.

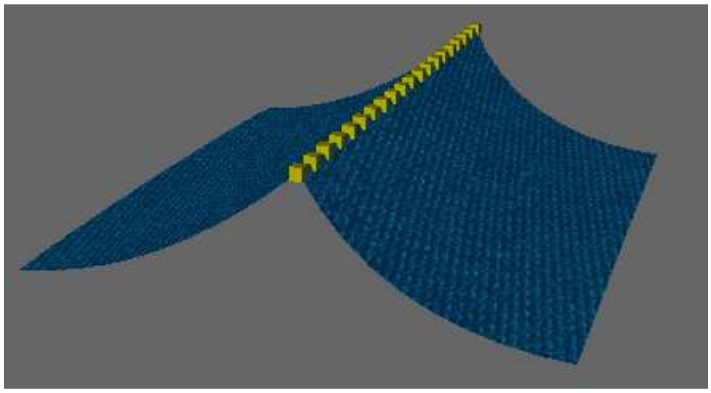
Using different arrangements of these pinned particles, I have implemented several kinds of constraints, as follows:



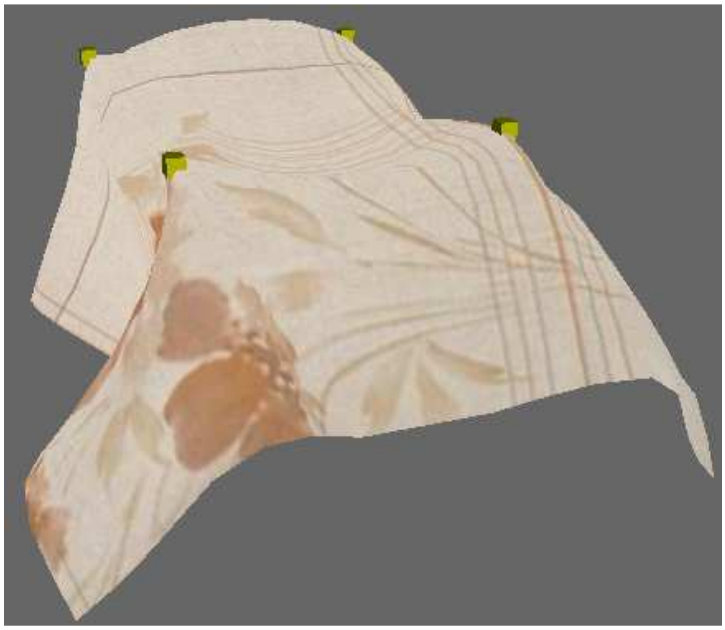
I have also provided functionality in my code to pin individual particles and thus get some very fine control to simulate very specific situations, such as the followings:



Single pin at the center



Row pin at the center



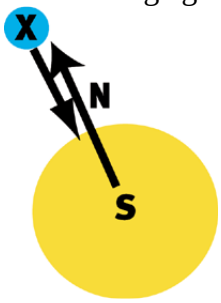
4 center pins

4. Collision

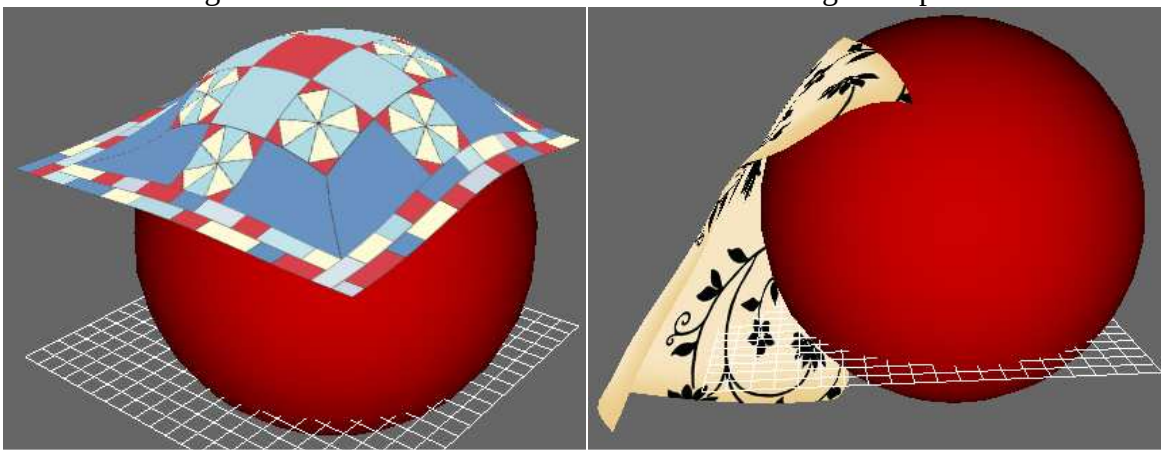
I have implemented collision detection and resolution for 2 specific objects, a sphere and a cube. At each movement of the particles, they are checked against collision with these 2 objects; in case collision is detected, the appropriate resolution method is called (each object has its own specific resolution method) and the particle is moved back to the surface of the obstacle.

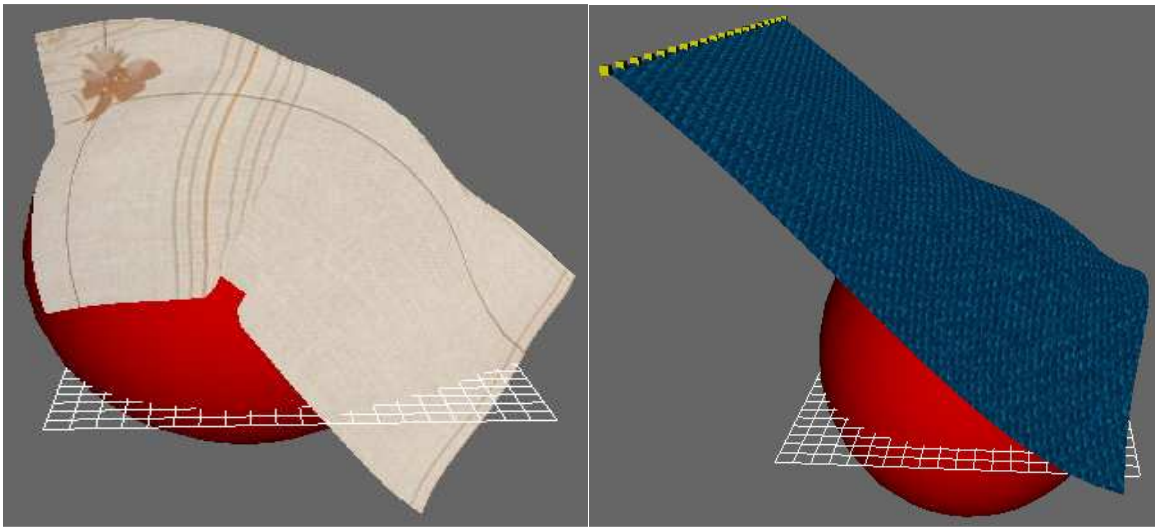
- *Sphere Collision*

Given the following situation, where X is colliding against S (Jeff L., 2010),



A normal N is calculated; X is then moved to the surface of S by (N*Sradius) in direction of N. The followings are some screenshots of collision resolution against sphere obstacle:





- *Cube Collision*

The cube has had very specific collision detection and resolution which is impractical with other polygons, but it does work very well here. The following is the algorithm:

For each particle

Check whether the particle is inside the box

If no

Then no collision

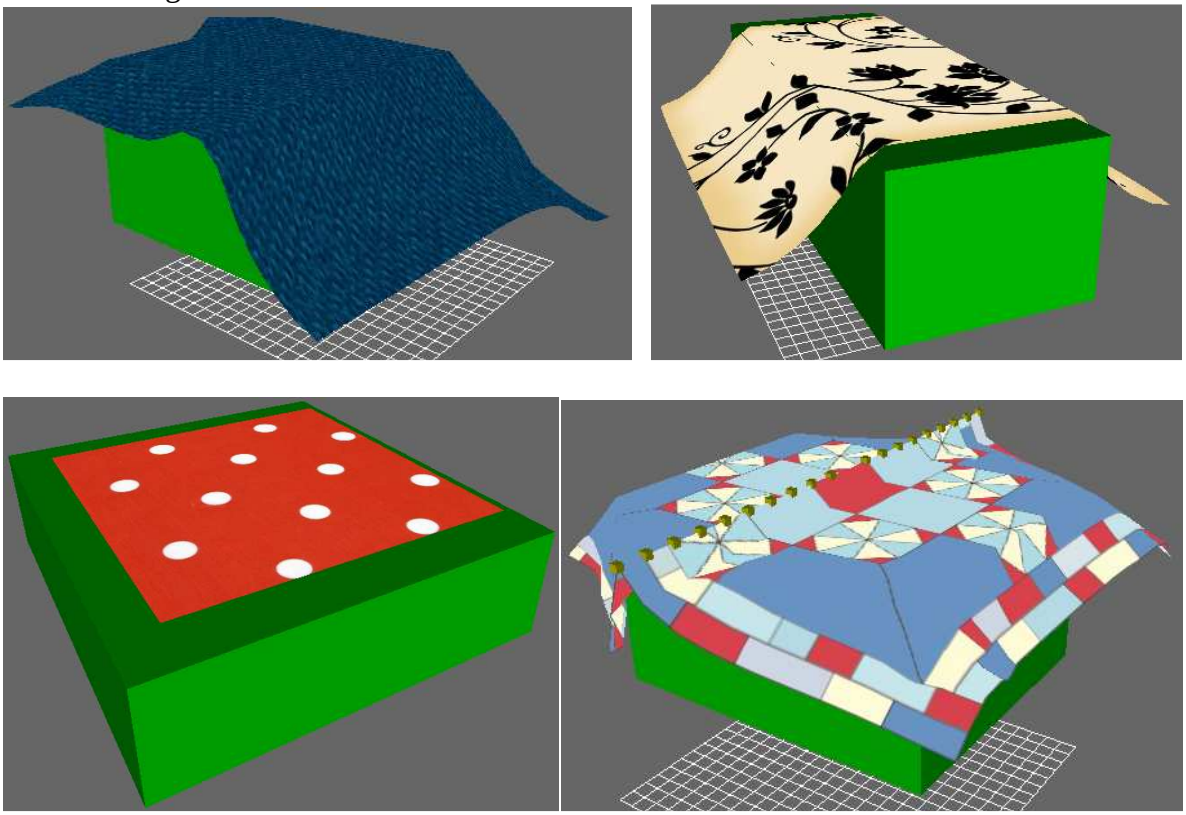
Else

//there is collision -> need to resolve now

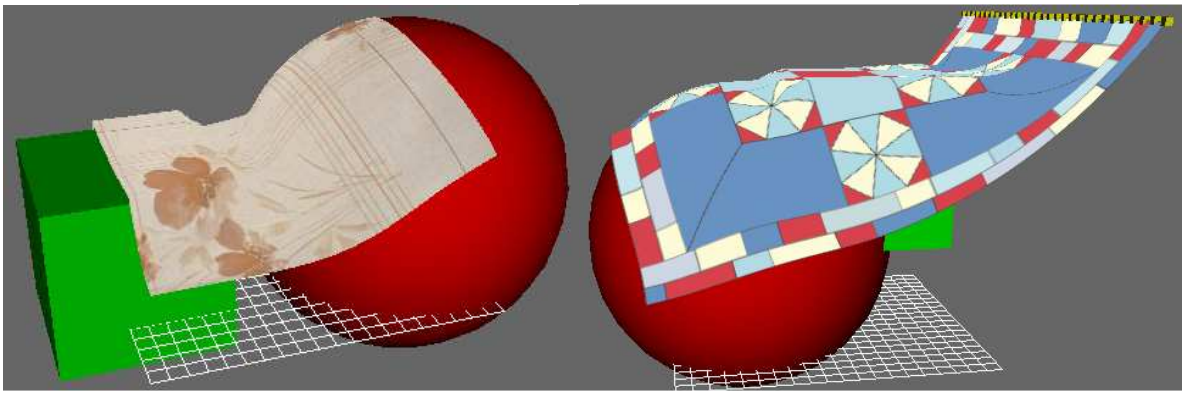
Get the direction of movement and compare with each face to determine Which face has been collided against

Move particle back to the surface of collided face

The following are some screenshots with cube collision and resolution:

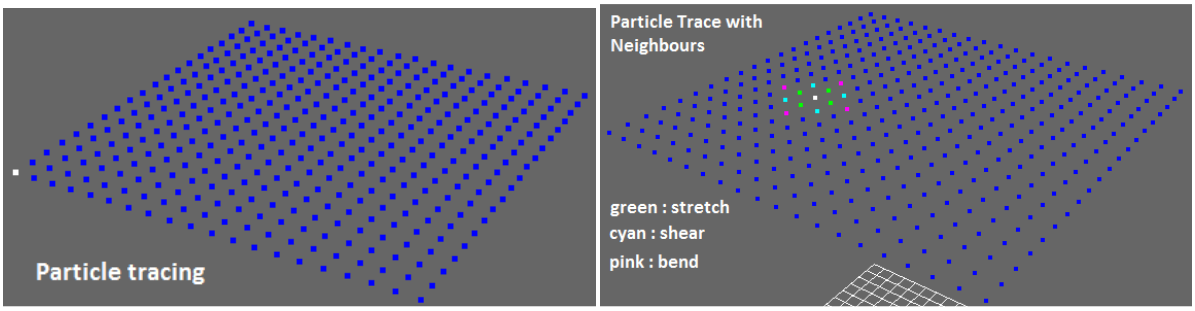


The followings are some screenshots of box and sphere collision resolution:



5. Internal Trace

I have added the functionality to be able to view each particle and each neighbour, in colour-coded fashion. Using the GUI also makes it easy to pin that particle. The following are screenshots of the internal trace enabled:



Challenges and Difficulties

While the result of the cloth simulation is good, stable and reacts well to collision in most cases, I faced numerous difficulties in the project and I had to compromise many features.

- Tiny Timestep

Given the timeframe, it was not possible to implement RK4 or implicit integration method and as such I had to make use of Eular Explicit and Verlet. As such, to have a good stable cloth movement with collision and varying damping and stiffness, I have to keep my timestep to very small, going in the order of 0.001 to 0.007; this resulted in a very slow simulation, doing many calculations per frame, and very limited timestep variation.

- Limited Damping

While using small timesteps gave a quite stable simulation with Eular explicit and verlet, it resulted in explosions of particles in many cases when I would try to increase the damping coefficient or even the stiffness. After some analysis, I found out that the force due to damping was actually getting out of range, greater than 1 and above. To cope with this, I limited the damping force/magnitude to 0.8. This arbitrary value helped keep the simulation stable at any damping coefficient. While this looks good on the interface, it has actually clamped the damping of the simulation to some value, thus I lost damping control in the simulation. As such, all the materials appear to be light and I was not able to get really heavy cloth simulation.

- Collision Detection and Resolution

Once the cloth is in contact with the obstacle sphere, it stays on it, but then eventually falls off. While this is logically not correct, it is due to the rounding errors that occur as we calculate the normal along which to displace particles away from the sphere center. The accumulation of rounding errors eventually move the particles on one side and causes a unbalance in forces, thus leading causing the cloth to fall off the sphere. A similar problem is solved in Maya ncloth using friction and stickiness properties and given more time, I would investigate on the implementation of stickiness property to solve the problem.

As for the cube obstacle, I am using a bounding box. However, the cloth usually goes a little bit through the sides of the cube. This is mainly due to the limited number of particles making up the cloth; increasing this will cause a finer collision resolution. Another solution could be to check collision with the springs also, besides just the particles.

Conclusion

This project has been a very good learning experience, since particle-based simulation is at the heart of many other systems, such as water, fire, and etc. I started with no prior knowledge of the mass spring technique and cloth simulation, and by the end of it, I got the cloth simulation to work.

Overall, I am very pleased with the results. While the Eular-explicit and verlet integration method forced me to keep the timestep very small and get a quite slow simulation, the latter is quite accurate and works well with a controlled variation in stiffness and damping coefficients. I was also able to implement collision detection and resolution against a sphere and a cube and the results were very convincing. I was also happy to implement solid shading and give out a choice of textures, in the form of materials, each one having a specific stiffness and damping coefficient. This has been a really challenging project but overall, a very rewarding one.

References

BlueThen, 2011, "Eular Method". Available online at :
http://upload.wikimedia.org/wikipedia/commons/a/ae/Euler_method.png
Last Accessed : 20.05.2011

Computer Graphics Labat the Alexandra Institute, 2010. Available online at :
http://cg.alexandra.dk/files/cloth.png (Last Accessed : 20.05.2011)

Jeff L., 2010, "Collision with a sphere". Available online at :
http://www.gamasutra.com/features/20000327/lander_08.gif (Last Accessed : 20.05.2011)

McCarthy J., "Comparison of Simulation Techniques using Particle Systems".

Available online at :
http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/F05/assignments/final_projects/mccarj7/index.html (Last Accessed : 20.05.2011)

Windclth, 2011. Available online at : ttp://freespace.virgin.net/hugo.elias/models/windclth.gif
Last Accessed : 20.05.2011

Yu-feng Y., Kai-jian, Jin-cheng Z., "Cloth Deformation Simulation Based on a simplified mass-spring model". (IJCNS) International Journal of Computer and Network Security, 2010, 2[9].

Bibliography

Baraff D., Witkin A., "Large Steps in Cloth Simulation", COMPUTER GRAPHICS Proceedings, Annual Conference Series, 1998 (SIGGRAPH 98, Orlando, July 19–24).

Dochev V., Vassilev T., "Efficient Super-Elasticity Handling in Mass-Spring Systems". In International Conference on Computer Systems and Technologies - CompSysTech'2003.

Martin S., 2003 , "Stable and Responsive Cloth", U.C. Berkeley.
Available online at : http://stevezero.com/eecs/cs294proj3/
Last Accessed : 20.05.2011