Cloth Simulation

Amitha Arun
Computer Generated Imagery Techniques Report
MSc Computer Animation and Visual Effects, 2014-15

November 20, 2014

Abstract

This report discusses the major topics involved in real-time cloth simulation. After comparison of the techniques used in cloth modelling, integration and collision detection, we choose the most appropriate ones to proceed with. The mass-spring model is discussed as the primary model for the construction of the cloth. The spring forces, external forces, damping forces etc are accumulated to give a value for the acceleration. The Verlet method is appropriate for integrating the acceleration value to obtain the position after force is applied. Finally, we can use sphere-plane collision detection concepts and self collision in order to make the simulation more realistic.

1 Introduction

Cloth simulation is a constantly evolving field of research in computer graphics. Although the fundamental concepts and mathematical equations have remained the same, the algorithms have improved consistently and there is still much scope for improvement. Many animated films and games require the generation of cloth in real-time. This is often a challenge because the process is slow and CPU-heavy. So, the algorithms should strike a balance between ensuring that the cloth has realistic properties and ensuring that it is efficient.

Certain areas in the model can be tweaked to provide a faster and a better quality of the cloth. One proposition is - moving the data onto the GPU. In doing this, we can ensure that the data transfer between GPU and CPU is reduced, hence decreasing the time to simulate the cloth. We may also use frameworks such as OpenCL which provide parallel computing in order to speed up the process. If we modify the code according to the requirements of a parallel program, the simulation time can reduced considerably.

2 Related Work

One of the first cloth synthesis ideas was conceived in the 1980s by Weil[15]. The result of this method was a cloth hanging on several constraint points in three

dimensions. After computation of the interior surface to the constraint points and creating an approximation of the shape of a cloth using these surfaces, the final cloth was rendered using a ray-tracer.

The earlier cloth simulation techniques were based on the elasticity theory. One such method was deriving a differential equation based on the theory of elasticity by Terzopoulos et al [13]. These were used to dynamically create shapes of non-rigid bodies. One of the attempts to improve upon this was to use implicit numerical integration to achieve larger time steps by Baraff et al[2]. According to Fischer [6], the elasticity model has several advantages over the basic Mass-Spring model. The elasticity model is based on integration of energy across the cloth. The various forces that act on the cloth result in a change in the enery leve of the cloth. Under this assumption, we may calculate the net force at a position (x,y,z) as:

$$F_{net} = \frac{\partial E(s)}{\partial x}, \frac{\partial E(s)}{\partial y}, \frac{\partial E(s)}{\partial z} \tag{1}$$

where,
$E(s)$ = Energy at current state of the system.

If we find the value of energy at a particular point, it must be integrated over all the vertices, faces and edges. This technique is not easy to implement although it provides a more accurate representation of the cloth.

The Mass-Spring is a simpler and a less elaborate model for dealing with cloth simulation.[16] It is useful in hair simulation, elastic solids and other such applications. This method uses Hooke's law and Newton's second law of motion as the foundation for the algorithm. For cloth simulation in particular, a network of springs can be created. This can also be used to demonstrate the working of strings and jelly etc.

The simulation of this mass-spring system can be approximated to the behavior of a cloth (See Figure 1). The force due to the springs is calculated for each of the nodes and integrated to obtain the final value of the force [11]. The force can be calculated as per Hooke's law:

$$F = -k \times D_x \tag{2}$$

where,
F = force,
k = Spring coefficient,
D_x = Displacement ($Dx = x_{final} - x_{initial}$).

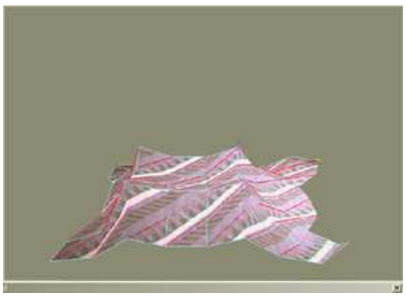

Figure 1: Cloth generated using the Mass-Spring Model. [11]

When the two ends of a spring are separated by a distance greater or less than its resting length, a force is applied and it's value is given by the above equation. If the distance is greater than the spring's resting length, the force is applied in the inward direction and vice versa. To find the positions of the nodes after the application of the force we can use Newton's Second law of motion:

$$F = M \times a \tag{3}$$

where,
M = mass,
a = acceleration.

Attempts have been made to improve the accuracy and the resolution of the cloth by using GPU-based kernels and data structures [12] (see Figure 2).

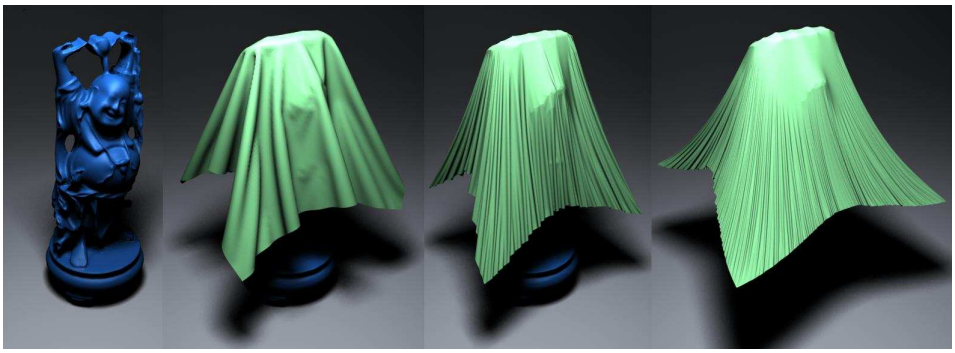

Figure 2: Simulations results of different cloth mesh resolutions. [12]

3 Mass-Spring Model

There have been several methods for cloth simulation, namely - physical, geometric, particle-based etc. The most common method for cloth simulation is

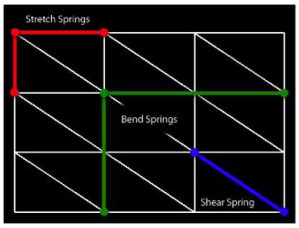

Figure 3: Placement of stretch, bend and shear springs in cloth nodes. [7]

called the Mass-Spring model. The mass-spring model consists of a set of nodes laid out to form a grid-like arrangement. We can assume this to be a polygon mesh that is composed of a collection of quadrilaterals or quads. The nodes in this mesh are connected to each other by a network of springs (as in Figure 3). The springs are of three types, namely

1. Stretch springs - These springs connect a node to its left/right and top/bottom neighbours. This type of spring is responsible for elasticity of the cloth.

2. Bend springs - These springs connect alternate nodes in a network. They control the amount a cloth can be bent. For higher values, the cloth exhibits more rigidity.

3. Shear springs - These springs are used to connect non-adjacent nodes in the same quad. They can control the cloth's resistance to shearing forces.

   Using these springs in conjunction with the nodes of the mesh, we obtain a simple cloth as shown below.

A representation of Hooke's law as presented by Langyel[9] is:

$$F_{spring} = k_{spring}(||Q - P|| - d)\frac{Q - P}{||Q - P||} \tag{4}$$

where,
P and Q are immediately neighbouring nodes connected by a spring and d is the resting length of the spring.

We may also add dampers to the network. Force is applied by the damper if there is a large difference in velocities that are computed for the nodes P and Q.

$$[F_{damper} = k_{damper}(\frac{dQ}{dt} - \frac{dP}{dt})$$

where $\frac{dQ}{dt}$ and $\frac{dP}{dt}$ are the velocities of the nodes.

A general equation by Fisher, considering the different forces acting upon the nodes, is shown below:

$$[F_{net} = Mg + F_{wind} + F_{air-resistance} - \sum k(x_{current} - x_{rest}) = Ma \quad (5)$$

where,
F_{net} = overall force,
M = mass,
g = gravity vector (0,-9.8,0),
k = Spring coefficient,
$x_{current}$ = current length of the spring,
x_{rest} = resting length of the spring,
F_{wind} = wind vector,
$F_{air-resistance} = -a \times velocity(v)^2$.

With the force that is calculated, we can quite easily arrive at a value for the acceleration that the force results in. This is a result of Newton's second law of motion that has been discussed in equation (3). Since the acceleration is the double differential of the position value, we can determine the final cloth position by integrating this acceleration vector. There may multiple ways of doing this integration and they are elaborated in the next section.

When converting these equations into an algorithm, we may use vectors having the x,y and z components. These vectors have operations such as addition, subtraction, dot product, cross product etc that can be used in order to find the resultant vectors.

4 Integrators

After obtaining the acceleration vector, we are expected to find the position vector by using integration since acceleration is the double differential of the position. There are several methods that can be used for integration and they are explained below. The integration methods chosen depend on various factors - size of problem, accuracy required, time needed for computation, simulation context etc. [14]

4.1 Euler's Integration

Euler's integration can be used to find the value of an integral when given an initial value using the tangent to find subsequent coordinates. A general representation of Euler's equation [1] is written as:

$$y_{n+1} = y_n + hF(x_n, y_n) \quad (6)$$

where,
$yn + 1$ = current y value,
yn = previous value of y,

xn = previous value of x,
h = step size.

We can approximate the curve more precisely if a smaller value of step size is used. This formula can be easily converted to an algorithm and a very small step size can be maintained for the computations without worrying about the complexity of the program. So, if this equation is written in terms of acceleration, velocity and position, we get:

$$v_{t+\triangle t} = v_t + \triangle t a_t \tag{7}$$

where,
v = velocity vector
a = acceleration vector.
The position may be calculated in a similiar manner as:

$$x_{t+\triangle t} = x_t + \triangle t v_t \tag{8}$$

where,
x = position vector.
Hence, using these formulae, we can iteratively arrive at a solution for the position vectors for all the nodes in the cloth simulation example.

4.2 4th Order Runge-Kutta Integration

It is another explicit method that is really an extension of the Euler's integration procedure. It is a more accurate solution to obtaining the integral [8] and it uses the formula explained as follows:

$$y_{n+1} = y_n + hT_4(x_{n+1}, y_{n+1}, h) \tag{9}$$

where,
$T_4 = \frac{1}{6}(k1 + 2k2 + 2k3 + k4)$
$k1 = f(x, y)$,
$k2 = f(x + \frac{h}{2}, y + \frac{h}{2}k1)$,
$k3 = f(x + \frac{h}{2}, y + \frac{h}{2}k2)$,
$k4 = f(x + h, y + hk3)$.
also,
$xn + 1$ = current x value,
$yn + 1$ = current y value,
yn = previous value of y,
h = step size.

This can also be converted into an algorithm without much complication.

4.3 Verlet Integration

Verlet Integration is an explicit integration technique that can be specifically used for the equations of motion.[4] It is a stable method and can be generalised as:

$$x_{t+\triangle t} = 2x_t - x_{t-\triangle t} + \left(\frac{dv}{dt}\right)_t \left(\triangle t^2\right). \tag{10}$$

where,
$\triangle t$ = change in time,
$x_{t+\triangle t}$ = value of x at time $t + \triangle t$,
x_t = value of x at time t,
$x_{t-\triangle t}$ = small change in x between t and $\triangle t$,
$\left(\frac{dv}{dt}\right)_t$ = acceleration.

Verlet integration is a very efficient method, is easy to compute and has several advantages over both the Euler Integration and the Range-Kutta methods. [3]

Although the explicit methods are easy to convert into computer programs due to the ease of computation, they are not very stable as the time steps we assume should be greatly reduced in order for us to get very precise and accurate results. So, there is a trade-off between the stability and the cost of computation for the integration methods. Implicit methods of integration may be used in order to obtain a more stable and accurate result but the ease of computation will have to be compromised. But for the purpose of this cloth simulation procedure, explicit integration would be largely sufficient as accuracy is not critical.

5 Collision Detection

Apart from its many uses in the gaming scene, collision detection forms in integral part of the cloth simulation model. A realistic cloth must respond correctly when it collides with other objects or itself. The concepts of collision detection remain the same no matter what objects are involved. It may be defined as the concept of recognising the interaction or intersection of two or more bodies. After the detection of a collision, the next step is to respond to it and this step is less complicated than the collision detection itself. The response to a collision can range from a body coming to rest to a body being set into motion, a body deforming or a body applying a force on another body and so on.

In order to proceed with collision detection mathematics, it is important to understand the vector operations. We can assume the velocities, position or forces of the colliding bodies to be vectors having 3 dimensions. The distance between the bodies can be calculated by finding the magnitude of the difference

of their position vectors. The magnitude is simply the square root of the sum of squares of the x,y and z component of the vector. Using this, we can determine if a collision is about to occur and if it has occurred, the exact position of the collision.

Let us consider the bodies involved in collision detection to be spheres. Based on this, we may have a sphere-plane collision, sphere-capsule collision, multiple-plane collision, sphere-polygon collision, self-collision etc. The sphere-plane collision phenomenon can be extended to understand the other collision patterns. The sphere-plane collision uses the normal of the plane to determine its distance from the spherical body. If the length of this normal is less than or equal to the sphere's radius, it means that a collision has occurred and the location of this collision can be isolated (Figure 4).

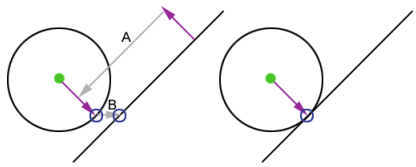

Figure 4: Sphere-plane collision detection using normals [10]

Collision detection can be Discrete or Continuous. Discrete methods need to have information about the distance needed to push the objects apart after the collision has occurred. [5] We can calculate the penetration vector by using the radii of the two spheres in the equation:

$$p = ||A - B|| - (r1 + r2) \tag{11}$$

where, p = Penetration Vector,
A and B are the centres of the spheres,
$r1$ and $r2$ are the radii of the spheres.

We can use this penetration vector to move the objects after the collision.

We must not ignore the cloth on cloth collisions. This could also work in a similiar manner to the sphere-plane collision detection, only the response would change. We could apply a repulsive force on the two nodes involved in the collision in order to effectively simulate the behavior of cloth in a self-collision.

6 Conclusion and Future Work

In conclusion, we can see that cloth may be simulated in multiple ways. We can choose the most efficient and at the same time, the most easy to implement algorithms for our purposes. With regard to the future work, the algorithm proposed could be optimised to improve performance and reduce the load on

the CPU. Post rendering the cloth simulation using OpenGL, the algorithm could also be parallelised in order to achieve the same results. We can do this by identifying parts of the code that are independent of each other and running these parts simultaneously on different threads. Moving the data to the GPU is another alternative although the improvement that is obtained by this is very marginal.

References

[1] Kendall Atkinson, Weimin Han, and David E. Stewart. *Numerical Solution of Ordinary Differential Equations*. Wiley-Interscience. A John Wiley and Sons, Inc., Publication, 2009.

[2] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 43–54, New York, NY, USA, 1998. ACM.

[3] Florian Boesch. Integration by example - euler vs verlet vs runge-kutta. `http://codeflow.org/entries/2010/aug/28/integration-by-example-euler-vs%-verlet-vs-runge-kutta/`, 2010.

[4] Jonathan Dummer. A simple time-corrected verlet integration method. `http://lonesock.net/article/verlet.html`.

[5] Paul Firth. Collision detection for dummies. `http://www.wildbunny.co.uk/blog/2011/04/20/collision-detection-for-dumm%ies/`, 2011.

[6] Matthew Fisher. Cloth. `http://graphics.stanford.edu/~mdfisher/cloth.html`, 2014.

[7] Zaxwerks Inc. Understanding stretch, bend, and shear. `http://zaxwerks.com/online_docs/3D_Flag_AE/2.0/stretch.html`.

[8] Autar K Kaw, Egwu Eric Kalu, and Duc Nguyen. *Numerical Methods with Applications*. autarkaw (2010), 2011.

[9] Eric Langyel. *Mathematics for 3D Game Programming and Computer Graphics*. Boston : Course Technology PTR., 2012.

[10] Paul Nettle. General collision detection for games using ellipsoids. `http://archive.gamedev.net/archive/reference/articles/article1026.html`, 2000.

[11] Gustavo Oliveira. Exploring spring model. `http://www.gamasutra.com/view/feature/131441/exploring_spring_model.php%?page=1`, 2001.

[12] Min Tang, Ruofeng Tong, Rahul Narain, Chang Meng, and Dinesh Manocha. A gpu-based streaming algorithm for high-resolution cloth simulation. *Comput. Graph. Forum*, pages 21–30, 2013.

[13] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 205–214, New York, NY, USA, 1987. ACM.

[14] Pascal Volino and Nadia Magnenat-Thalmann. Comparing efficiency of integration methods for cloth simulation. In *Proceedings of the International Conference on Computer Graphics*, CGI '01, pages 265–, Washington, DC, USA, 2001. IEEE Computer Society.

[15] Jerry Weil. The synthesis of cloth objects. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 49–54, New York, NY, USA, 1986. ACM.

[16] Nico Zink and Alexandre Hardy. Cloth simulation and collision detection using geometry images. In *Proceedings of the 5th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, AFRIGRAPH '07, pages 187–195, New York, NY, USA, 2007. ACM.