Devil in the Blue Faceted Dress: Real-time Cloth Animation

I've been describing methods of dynamic simulation using mass and spring systems for the past couple of months. These techniques excellently increase the realism in your real-time graphic simulation. One of dynamic simulation's key benefits is that it creates a scaleable game experience.

Users with more powerful systems get a more realistic experience, while users with less powerful systems are still provided with a complete experience. It's an analogous situation to the use of levels of detail in your 3D models. Particularly in the PC market, where target systems can vary widely, these techniques have become a crucial weapon in the developer's arsenal.

For a current project, I decided to maximize the use of dynamics to increase realism wherever possible. The project focuses on characters in moody interior environments. It occurred to me that the use of cloth animation in my scenes would be crucial to creating the mood I was trying to establish.

Traditional Cloth Animation in Games

Cloth animation is tricky. Even in the world of high-end computer graphics it's difficult to get right. Most of the time, it's wise to avoid the whole issue. Anyone who has ever created a female character in a skirt is familiar with the problem of the legs poking through the cloth mesh during animation. This is pretty difficult to fix, especially if animation requires a variety of motions. It's particularly tricky if you are applying motion capture data to a character. Unfortunately, it's also a really obvious animation problem that any end user can spot. These cloth animation problems outline the reasons why most digital characters are clothed in tight-fitting gear, such as skin hugging stretch pants.

Most loose clothing doesn't look natural in digital art because it's static. It doesn't move along with the body. It's possible to morph the shape of the skirt to match the motion of the character, but this requires quite a bit of detailed animation work. Likewise, deforming the skirt with a bone system can be effective, but not necessarily realistic.

For my work, I wanted to create realistic cloth in the environments and on the characters. My hardware accelerated graphics rasterization freed the processor power necessary to make this possible. So, I set about creating a real-time cloth simulation.

The Latest "Springy" Fashions

The mass and spring dynamics simulation I developed in my *Game Developer* column ("Collision Response: Bouncy, Trouncy, Fun" March 1999) proved effective for simulating soft body objects in real-time. I thought it should be possible to use these techniques to create a cloth simulation. In fact, several of the commercial cloth animation systems for 3D animation programs such as 3D Studio MAX, Softimage, and Maya use similar techniques. So how do I go about creating a piece of cloth?

I am going to be using the same spring force formulas for the cloth simulation as the ones I used in the March 1999 *Game Developer* column. If you are unfamiliar with the dynamic forces generated by springs, you should go back and read the March column or at least take a look at the March source code on the *Game Developer* website (http://www.gdmag.com).

I start by creating a rectangular grid, and then connected each point to neighboring points with springs, as you can see in Figure 1A. These springs define the rough structure of the cloth and so I refer to them as "Structural

FIGURE 1. *The devil's animated-cloth blue dress.*

Jeff Lander prefers to wear comfortable loungewear when hanging out writing code at Darwin 3D. Drop him a note and let him know what the fashion conscience are wearing this spring at jeffl@darwin3d.com.

Springs." The resulting cloth patch looks pretty good and requires few springs. However, once I run the simulation, problems appear immediately as shown in Figure 1b.

The simple spring connections are not enough to force the grid to hold its shape. Much like the box in the March column, there are simply no springs to maintain the shape. If I held on to only one point, the entire surface would collapse into a single line creating a rope. Not exactly what I wanted, but it points out something I want to address a bit later.

I really want to keep the model from shearing too much. That is, I want the space between diagonal elements of the model preserved. So, I just add a few more springs to the grid along the diagonals creating a group of "Shear Springs," as you can see in Figure 2A. Run this new structure through the simulation and the results are much better, as you see in Figure 2B.

This new form of cloth works pretty well hanging from hooks on the wall. However, if you drop the cloth on the floor, it wads up into a big mass of springy spaghetti. The reason for this failure is that the model is still incomplete. If you look at the structure in Figure 2a, you may see that there is nothing to keep the model from folding along the edges of the structural springs, much as you fold a handkerchief. The fibers that compose actual cloth run the length of the fabric and generally resist folding and bending. In order to simulate this effect adequately, I need to do a little more work.

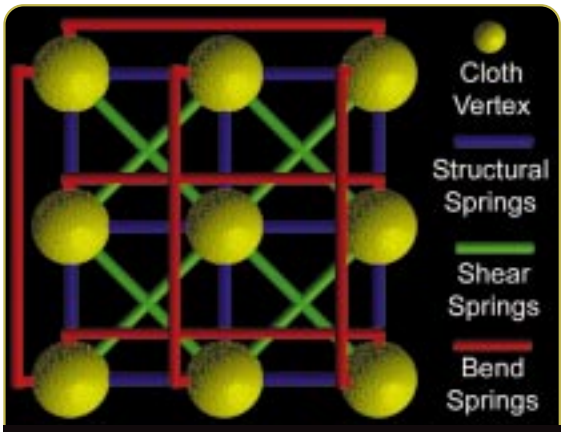


FIGURE 1. *The interconnection of cloth springs.*

My research uncovered two methods for dealing with this problem. The first minimized the bend between two adjacent cells by using the dot product to determine the angle of bend. The second method simply added an extra set of springs called "Flexion" or "Bend" springs to apply the bend force. I created the bend springs by stretching a spring across two cells alongside the structural springs. These springs end up connecting every other cell in the cloth mesh.

I prefer the second method because it works within the existing spring system without the need for a new method for calculating forces. I also get the benefit of having only one calculation to optimize later. For other applications, it's possible that the angle minimization method may work out better.

I now have a sufficient spring network to simulate a variety of different types of cloth. You can see how all the springs are connected in Figure 1.

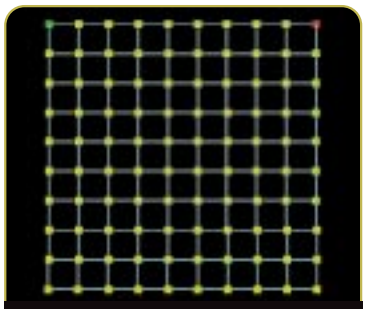


FIGURE 1A. *A simple cloth grid.*

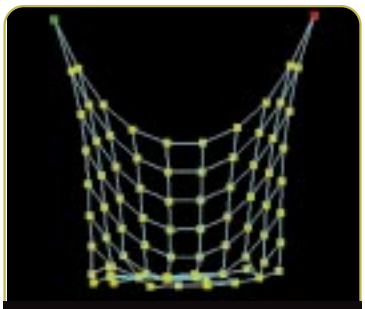


FIGURE 1B. *Stretched cloth grid.*

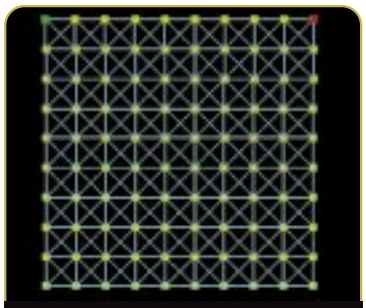


FIGURE 2A. *Added shear springs.*

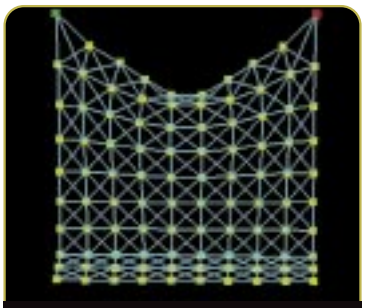


FIGURE 2B. *A better cloth model.*

Stretch without Tearing

These three types of springs make it possible to simulate a variety of different cloth types. By varying the stiffness of the springs, it's possible to simulate anything from stiff cardstock to stretchy nylon. For example, spandex would have very flexible structural and shear springs to allow strong stretching capability. Paper, on the other hand, is very resistant to shearing and stretching, so its springs would be very stiff. When considering how much a material will bend, a surface like cardboard should have the very stiff bend springs to make it resistant to folding. I find experimenting with different values for spring stiffness the only real way to find adequate surface properties.

Stiff springs can make a numerical simulation unstable. To combat this

FIGURE 2. *Collision boxes allow for the draping effect on this tablecloth.*

it's important to use a good numerical integrator. The midpoint method and Runge-Kutta integrators developed last month seem to do the trick nicely.

Making It Move

I already have a simulator from the March column that is capable of handling a cloth patch. I can even apply gravity to it and lock the position of individual vertices. That's pretty interesting, but it needs some improvement to come alive. In March, I also discussed the use of planes for collision. With this same method, I can create collision boxes that enable me to simulate a tablecloth draped over a table, as you see in Figure 2.

This model is interesting and realistic looking but not terribly animated. In fact, in this case it's probably better to freeze the simulation and avoid the constant recalculation. Unless, of course, the wind kicks up or someone pulls on the corner.

For characters, a moving box is not the most realistic way to displace the cloth. Moving bounding spheres allow much more pleasing character animation. Fortunately, this is easy to add to the simulation. Determining if a point is inside a sphere is very easy. If the distance from the point to the center of the sphere is less than the radius of the sphere, the point is on the inside. If a point in the cloth is found inside a sphere, I have a penetrating collision. Just like handling collisions in the March simulator, I need to back up the simulation time to find the point of actual contact. In a sphere, contact takes place when the distance of the point to the sphere center is equal to the radius of the sphere. Now that the contact point is established, I need to resolve the collision. The collision normal, *N*, between a point and a sphere is the vector between the point of contact and the center of the sphere. You can see this in Figure 3. Fortunately for me, the rest of the collision response is handled just like the collision with a plane. This means my existing collision response code works great.

I can now add collision spheres to my simulation. The cloth slides realistically off the spheres. You can see how the simulation looks with two collision spheres in Figure 4. By animating the spheres along with the 3D model, I can get a nice animated hip sway and other alluring effects. The motion of the cloth continues after the animation stops. This creates entertaining effects that are diffiicult to achieve with traditional animation techniques.

Problems to Avoid and Ignore

The simulation has a couple of problems. The first is that the way to simulate cloth realistically is to use a lot of points in the simulation. This takes more computation time. High-end animation programs rely on a great numbers of particle points for realism. Of course, in other fields, hour-long render times are perfectly acceptable. In a real-time game, however, this won't get you on the cover of any game magazines. You have to sacrifice realism for speed. This is another good area for scaling game performance. If the system is running quite fast, subdivide the cloth patches a little more. Game players with a white-hot system should have smooth-looking cloth.

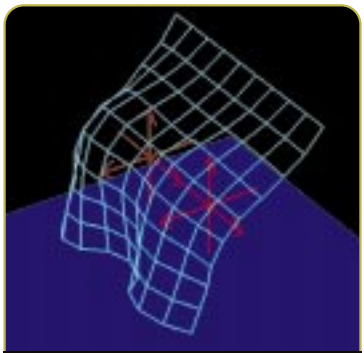
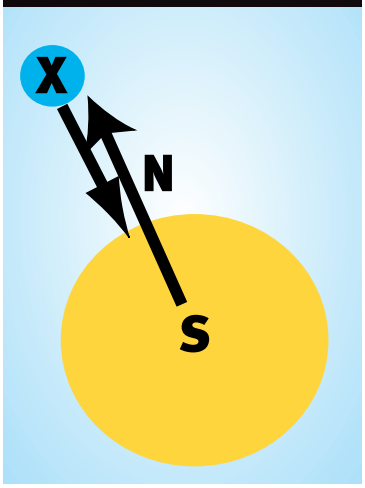


FIGURE 4. *Cloth hanging from a pair of invisible hips.*

FIGURE 3. *Collision with a sphere.*



Another problem is that each spring acts independently. This means that each spring can be stretched to a great extent. In many cases, the amount of stretch can exceed 100%. This is not very realistic. Actual fabric will not stretch in this manner. The problem I have is that I am using linear springs when fabric actually displays a non-linear spring behavior. As the amount of stretch increases, the strength of the spring increases also. The fabric will also stretch to some limit and then if the force continues, it will rip. This is not what I want (at least for now). This issue, which Xavier Provot (see For Further Info) calls "the Super-Elastic Effect," is difficult to handle. Increasing the spring strength dynamically can lead to instability problems just like any other stiff spring problem. Provot suggests checking the amount of stretch in each spring and if it is exceeds a set deformation limit, the springs are adjusted to achieve this limit. While I agree this solves a definite problem, a second pass through the springs is costly. For the effects I have attempted to achieve, I can live with super-elastic cloth.

My collision system is pretty primitive. To make things easy, I only collide the vertices of the mesh with the objects. As it stands, if a sphere is small or the fabric stretches too much, the sphere will pass right through it. I

also don't handle self-collisions. That is, the fabric can pass through itself without penalty. This could be corrected by placing bounding spheres at each vertex. However, applying the sphere collision test between each vertex gets expensive. So, I just limit the situation so either that cloth doesn't pass through itself, or so that it isn't too noticeable.

Taking It to the Limit

Once the system is working, it's fun to see how it can be extended. I mentioned the issue of tearing and ripping after the fabric stretches too far. I can monitor the spring lengths. If they exceed a limit, the spring can be removed from the system, effectively tearing the fabric. I think this would be a great way to simulate a cannonball tearing through the mainsail of a tall ship. This same method of breaking a spring would work for a simulation of a rope as well. After all, a rope is really just a one-dimensional version of the cloth patch.

Another dynamic effect can be achieved by manipulating the flexion springs. With these springs in place, the fabric will resist folding. However, if I selectively delete one of these springs, the fabric will be able to fold nicely where the springs are missing. I don't know where I can use that yet, but I'm sure I can find a way.

The Application

The application this month was actually pretty easy to build. It's the same as last month's, with a few additions. There's a function that creates the cloth patch in a sort of macro fashion. You can set the spring settings for the three types of springs. You can also drop some collision objects around and watch them interact. Get the application and the source at http://www.gdmag.com. ■

SPECIAL THANKS:

Go to Chris Hecker of Definition 6 and Rob Wyatt of Dreamworks SKG for discussing the issue with me.

FOR FURTHER INFO

• Provot, Xavier. "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior," Graphics Interface, 1995, pp. 147-155. Also available in electronic form at http://www-rocq.inria.fr/syntim/research/provot

• Baraff, David and Andrew Witkin. "Large Steps in Cloth Simulation," Proceedings of SIGGRAPH 1998, ACM SIGGRAPH, pp. 43-54.

There are also fabric simulations available for many professional 3D animation packages available either as plug-ins or integrated into the software. I do not know what techniques these products use with the exception of one. Colin Withers of Topix created a fabric simulation for Softimage based on the Provot paper. Graciously, Topix released the source code for this plug-in to the public. See http://www.topix.com for more info.