Flag in the Wind: Using the Fundamentals of Cloth simulation to Achieve a Flag Effect

NCCA, Thesis Portfolio
Peter Smith

August 2011

Abstract

Cloth simulation is an area of ongoing research widely used in realistic simulations for film and games. The following paper is a technical write up for a simple implementation of particle based cloth. This implementation features both Euler and Verlet integration.

Contents

List of Figures

Chapter 1

Simple Cloth Simulation

The following paper is a technical write up for the implementation of a cloth simulation written in C++ using OpenGL. It contains the details of implementing the spring-mass model as well as a couple of variations of explicit integration types. The technical write up will also briefly assess other integration methods and advanced topics of cloth simulation.

The cloth sim was made over a period of 2 days and was aimed as a demo of a flag blowing in the wind.

1.1 Cloth as Particles

In order to render an approximation of cloth, a set of points must be given to OpenGL. These points can be used to rasterize several triangles. Defining where these points are, is fundamental to cloth rendering. In addition to this, in simulating cloth, defining a lattice of inter-connected mass points is also important. Due to these two similar paradigms, defining the cloth structure as a set of particles is justified. This is because particles changing over time supply the 3D positions needed by OpenGL whilst also being useful for our cloth simulation with attributes like velocity and mass.

As such the particle states are used in numerical integration to calculate their positions over time. In most conventional particle simulations forces like gravity are applied, and the accumulated force is used with other state variables in integration. The key difference, as with other variations of particle based simulations is the method in which force is injected and extracted from the system. This is where using the mass-spring technique is important.

1.2 Springs

There are 3 types of spring outlined by Parent[1]; bend, structured and shear. Each of these springs work in fundamentally similar ways. Two of the cloth particles are connected to the spring, at either end. Each spring has a rest-length, which is the

distance that the spring will always attempt to return to. In other words, in addition to the external forces being applied to the cloth particles, the springs will apply a force to each particle, based on spring length and direction. The spring direction is simply the normalised vector between each of the springs particles. Spring forces can also be referred to as internal forces; they are generated by the system itself. These internal spring forces are used in reaction to the external forces of the cloth, like wind and gravity.

1.2.1 Structured Springs

Structured springs are the backbone of the cloth, which ensure stability and connect every adjacent point. Figure 1.1 shows a diagram of how these springs are setup.

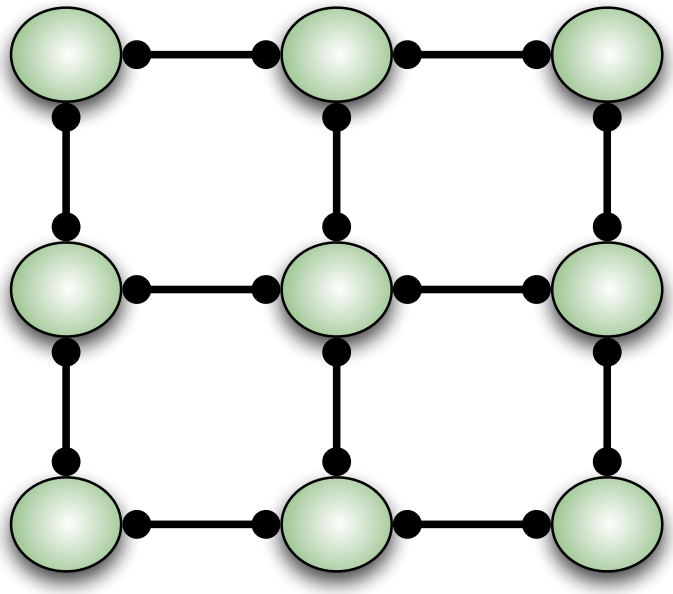

Figure 1.1: Structured Springs Layout

They are constructed in a set of rows and columns around quadrilaterals in the cloths points. These springs help to maintain the cloths structural integrity when affected by outside forces. For example, if a point is constrained and its neighbour is pushed downwards by gravity, the spring will lengthen and attempt to contract to its rest length.

1.2.2 Bend Springs

Broadly, bend springs are intended to stop the cloth from folding in on itself. They join connecting sets along the edge of the cloth. Figure 1.2 shows this.

These springs act to moderate external forces applied perpendicular to the longitudinal direction of the cloth. The figure shows this on a small patch of cloth, in larger

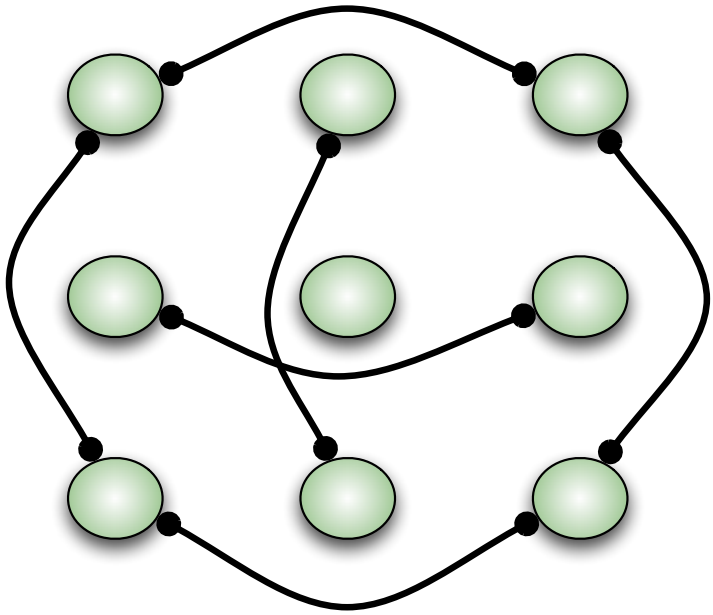

Figure 1.2: Bend Springs Layout

patches the outer connections are distributed in even intervals along the edge points.

1.2.3 Shear Springs

Shear springs traverse diagonally between points to moderate and create lateral movement. See figure 1.3.

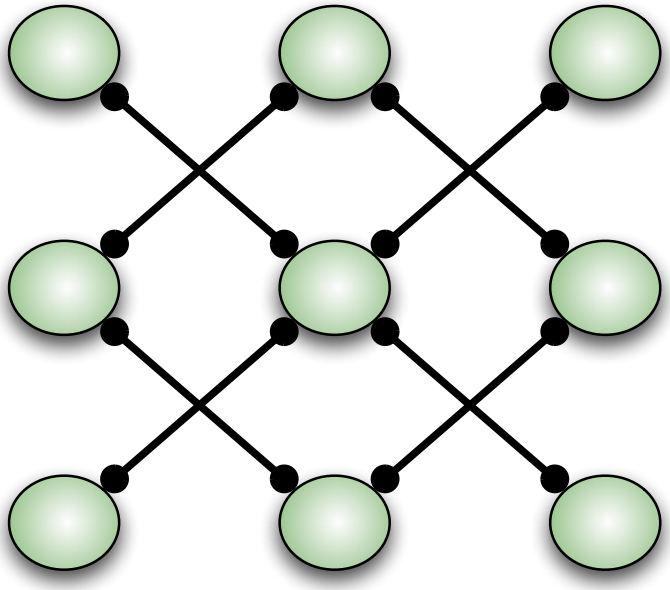

Figure 1.3: Shear Springs Layout

They specifically moderate against force perpendicular along the latitudinal direction of the cloth.

1.3 Integration

The two different methods of integration in this application are Forward Euler and Verlet. Integration, in this context, is a function of time and is used to move the particles in space based upon their current states. This usually involves the variables acceleration, force, velocity and of course, position. Both the methods implemented for this application are explicit methods. Put simply, they calculate new state values based on a given time step and have no knowledge of the future state of the system. Implicit methods are usually more complex and work in the opposite way; working out the next time step starting from the final state in the simulation.

1.3.1 Euler

Forward Euler integration gets its name from the type of numerical differencing it uses. An estimation of the derivative is made by using forward differencing. The new position can be found based on the current position and delta of this position. In this case the delta is the velocity multiplied with the time step. The formula for this is:

$$P = P + V * \delta t$$

Where P is Position, V is Velocity and δt is the time step. The derivative of the velocity is acceleration and this is used in a similar way to calculate a new velocity for the next time step.

$$V = V + a * \delta t$$

Where V is Velocity, a is acceleration and δt is the time step.

1.3.2 Verlet Integration

In contrast, Verlet integration uses second order central differencing for estimating its derivative. This means more samples are taken and higher detail achieved. This makes for a more stable and accurate simulation. The method takes the previous position into account along with the forward difference of the position with respect to time. [1] This method can be improved by using time-corrected Verlet (REF) which takes the delta time into consideration. The formula for this implementation however, is:

$$P_i = P_i + (P_i + P_{i-1}) + a * \delta t * \delta t$$

Where P_i is current position, P_{i-1} is previous position, a is acceleration and δt is the time step.

[1] calculated using the acceleration

Chapter 2

Conclusion and Other Features

2.1 Other Features

In addition to the core features I also constrained some points in the application demo. This essentially means that certain points will never move regardless of external or internal forces. In the case of this implementation this was used in order to simulate a flags fastenings.

A pseudo-random wind gust system was also made to give more aesthetic to the motion of the flag. This was done by taking the wind vector and multiplying each component of it by a coefficient close to 1.0. This allowed for subtle variations in direction.

Finally, normal re-calculation was added. This was done by taking the vector between the current vertex and two connecting vertices and using the cross product to find the orthogonal vector; the vertex normal.

2.2 Conclusion

Overall, the implementation of a simple cloth simulation was successful and allowed for the fundamentals of the process to be explored. The desired effect of a flag blowing in a squall was achieved. However it is noted, that Euler integration did not hold up under aesthetic scrutiny and did not supply a convincing look and feel. Verlet, however was far more pleasing and technically robust due to its use of central differencing and second order accuracy. Figure 2.1 and 2.2 show Euler and Verlet integration respectively. These figures also display the other features of the demo application like normals and texture co-ordinates.

In future, Runge-Kutta 4th (RK4) order integration will be explored as well as implicit methods. These methods are the ones most commonly found in film production software and are considered the most robust.

Figure 2.1: Euler Integration

Figure 2.2: Verlet Integration

Bibliography

[1] Parent, R. 2008. Computer Animation: Algorithms and Techniques. 2nd Edition. Massachusetts : Morgan Kaufmann.

[2] Fielder, G. 2006. Spring Physics. Gafferon Games. Available from http://gafferongames.com/game-physics/spring-physics/ Last Accessed 17 May 2011

[3] Dummer, J. 2007. Verlet Integration Method. Available from: http://www.lonesock.net/article/verlet.html Last Accessed 14 May 2011