Real Time Cloth Simulation

Sebastian Olsson (81-04-20)
Mattias Stridsman (78-04-13)

Linköpings Universitet
Norrköping
2004-05-31
## Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Spring Systems</td>
<td>3</td>
</tr>
<tr>
<td>Theory</td>
<td>3</td>
</tr>
<tr>
<td>Implementation</td>
<td>4</td>
</tr>
<tr>
<td>Collision Detection</td>
<td>4</td>
</tr>
<tr>
<td>Rendering</td>
<td>5</td>
</tr>
<tr>
<td>Results</td>
<td>5</td>
</tr>
<tr>
<td>References</td>
<td>5</td>
</tr>
</tbody>
</table>
Introduction

The aim of this project was to learn how to simulate deformable objects in real time, especially cloth. We chose to use a very simple spring system approach which could later be extended and added too. We decided to implement it in C# (C-Sharp) using OpenGL. In order to be able to use OpenGL in C# we used a graphics library called CsGL (C# Graphics Library). CsGL caused some minor problems, and appears to have a few bugs. C# is also an interpreted language, and said to be a bit slower than C++, and this is another reason we wanted a simple model to start with.

Because of the time limits (both implementation wise and the limited calculation time for each time step of the simulation) the cloth does not collide with itself. This is quite a complex problem and it is also computational heavy, which made it an easy decision to leave this out. It does collide with other objects in the scene.

In this report you will find a brief explanation of the theory, followed by a description of our implementation and the results.

Spring Systems

Theory

The idea of a spring system is to approximate deformable objects. A spring system consists of a network of nodes, connected by springs. Each spring has a resting length. When the distance between the two nodes the spring is connected to is equal to the resting length, the spring is not affecting the two nodes with any force. When the distance between the nodes is greater than the resting length, the spring will apply a force to them, thus trying to move them closer, and vice versa. The spring force is linear to the displacement, and a spring coefficient controls the stiffness of the spring. The higher the coefficient is, the stiffer the cloth will be. You make it stiff enough to simulate a sheet of paper. A higher spring coefficient will make the system more unstable.

A spring system can be seen as a discrete approximation of a real continuous object. One node’s movement will propagate through the whole system as neighbouring nodes will move and affect their neighbours and so forth, just like a real cloth would do if you were to pick it up by the corner. Since we are simulating cloth, the nodes are placed in a plane. You can place the nodes in any formation you like, and depending on the formation you will get different results. The springs between the nodes can also be connected in many ways.
Implementation

We used the spring and node structure as seen in Fig. 1. This will keep the nodes in a grid formation. The diagonal springs will prevent the structure from shearing (without the diagonal springs the lower three nodes could align vertically with the middle three nodes and no springs would do anything about that).

This form works pretty well, but the cloth has no stiffness, so there is nothing to avoid it from folding into a small flat pile. To prevent this we connected two additional springs for each node; one from the left neighbour to the right neighbour and one from the neighbour below to the one above.

The spring calculations consist of four steps:

1. For each node, calculate the resultant force from all the springs it’s connected to.
2. For each node, we calculate an acceleration (based on the force) and a velocity.
3. For each node, we loop through all polygons and check if they collide.
4. Place nodes which should have a fixed position to that position (for example, we could make edge nodes fixed to a curtain rod.)

Collision Detection

The collision detection was implemented by checking each polygon in the scene against all nodes. This is potentially a very CPU intensive operation, but our scene is very simple and only consists of a small number of polygons. For larger scenes with more objects a number of optimization tricks would be necessary (bounding boxes, quad trees).

Before we move a node, we check for collisions. The collision algorithm is described below.

For each polygon:

1. We first check to see if the node is moving towards the polygon plane. If not, no collision is possible, so we move this node to its new position and exit.
2. We check if the collision point on the polygon plane is within reach of the current time step.
3. We check if the collision point on the polygon plane lies within the polygon.
4. If the distance to this collision point is the closest collision point we have found, we set the closest collision point to the current one.

If we reached step 4 above we have found a collision point. We then move the node to the collision point and cancel the velocity in the polygon normal direction.
Rendering
The cloth is rendered by creating quadrangles between 4 nodes. When the cloth is resting on a surface, OpenGL sometimes draws the cloth and sometimes the surface because of numerical errors. This results in flickering. To resolve this, we used `glPolygonOffset()`, which lets you change the depth values.

When the cloth is resting on a surface it is constantly moving (because the spring system is never in rest). The vertex normal is constantly changing values because of this, and it can be seen as a flickering in colour intensity. We solved this by averaging the normal over a small period of time.

Results
The spring system we implemented works, but it is quite unstable, especially for large number of nodes and when it is resting on a surface. We had to use a low spring coefficient to prevent the whole system from “exploding”. We also used a low gravity and a high damping to minimize the unstable behaviour. This gives the simulation a sense of going in slow-motion. Since we want the simulation to run in real time we cannot solve this.

We have created a couple of scenes with a cloth and some simple objects, like a plane and a sphere. The cloth can be moved using the mouse, although this functionality is somewhat limited you can drag the cloth over the objects and it behaves reasonably realistically.

The next step to improve this simulation would be to move from explicit Euler to implicit Euler integration. This will make the need for low gravity and high viscous damping obsolete.

References