Cloth Modelling - A Literature Survey

By: Gavin Hayler

Due Date: 31/05/2004

Abstract

Cloth modelling is a highly researched field with many different algorithms to choose from. The algorithms one chooses to solve the cloth modelling problem relies heavily on the application or desired purpose. This review is written from the perspective of choosing an algorithm for an offline animation rendering application, and attempts to provide some insight into the advantages and disadvantages of the various algorithms. It will also explain why certain algorithms were chosen for the afore mentioned purpose.

1 Introduction

Cloth modelling refers to the process involved in simulating the behaviour and look of cloth, and has been an interest of animators, programmers and engineers for some time. However, in more recent years, it has become a highly researched area because two industries have exploded: gaming and computer generated imagery (CGI). Each has different needs from the models that they employ. Game developers need a model which is fast enough that it can be implemented into a game, used in real time, and a user is still able to play the game without any latency or slowing down of the system, and which still makes the cloth look believable. In contrast, CGI animators require the model to perform as close to real cloth as possible but the time taken to process the cloth is not of primary importance, since the actual rendering is done offline on a render farm (a large number of computers dedicated to the task of rendering a scene). Since there are two areas to a cloth simulation, dynamics and collision detection, we will be investigating each area separately. For the dynamics of the cloth model, many different types of integration have been used in the past, but they usually fall into the category of either explicit integration or implicit integration. We will be examining the way the integrations used have migrated from explicit to implicit methods as the area of cloth modelling has matured, and why this has occurred. Collision detection is a relatively separate field to cloth modelling, so we will be examining the different techniques used in the field, not just the techniques that are used in cloth modelling, and discussing the advantages and disadvantages of each.

2 Cloth Dynamics

2.1 The Physical Model

Previously, a number of different physical models have been used to model cloth, such as:

- a continuum, as described by Amirbayat et al. [Amirbayat et al., 1989];
- Eischen et al. [Eischen et al., 1996] use a non-linear shell model;
- and Baraff and Witkin [Baraff et al., 1998] use a system of interconnected flat triangles that are treated as a continuum.

These models have been superseded by a system of particles which is used to represent the cloth. This model is defined well by Choi et al. [Choi et al., 2002], who draw their model from an idea that started with Breen et al. [Breen et al., 1994] who were first to apply a particle system to the simulation of cloth. Choi et al. [Choi et al., 2002] describe the model as a mesh of interconnected particles which approximate the cloth. These particles are connected by a system of springs with certain attributes assigned to them, depending on the type of interaction that it is dealing with (stretching or compression). In general, variations of this physical model are now used throughout the cloth modelling field - by game developers and CGI animators.

2.2 Integration

As mentioned before, there are two types of integration used in cloth modelling - explicit integration and implicit integration. Explicit integration dominated the field for some time, until Baraff and Witkin [Baraff et al., 1998] introduced the idea of implicit integration into cloth modelling. We shall now examine these two integrations.

2.2.1 Explicit Integration

The most commonly mentioned examples of explicit integration are the Euler and Runge-Kutta methods. These methods are fast and easy to implement, but do require a small time step to maintain stability. Because of this fact, this type of integration is usually used in real-time applications such as games where the time step is small and the regularity of the system cannot be determined [Rudomin et al, 2002].

Explicit integration attempts to solve the system based on the information available at time t_0, and has no idea of what the system will be like at the end of the time step. It takes no notice of wildly changing derivatives as it continues to solve the system. This is the reason that the time step has to be small to maintain stability, so that there will not be any huge changes that could corrupt the results. [Baraff et al., 1998]. Each step of an explicit method is faster than that of an implicit method, however the more particles that exist in the system,

the smaller the time step needed to keep the stability of the system [Oshita et al., 2001]. Therefore, providing the particle system used to approximate the cloth has relatively few particles in it, explicit integration works well. In general, game developers try to keep the number of vertices for each object to a minimum, making it an excellent application for explicit integration.

Most real-time applications do, in fact, use explicit integration for their cloth engines. One example of this in industry is the game Hitman: Codename 47. The paper written by the author of its physics engine [Jakobsen, 2001] discusses the methods used to create the cloth dynamics in their game. An variant of the integration called Verlet integration was used which is an explicit integration technique, and as the paper shows, very easy to implement. Other papers whose aim is real-time simulation from an academic point of view also use explicit methods, and obtain good results [Rudomin et al., 2002],[Oshita et al., 2001]. For example: Rudomin et al. [Rudomin et al., 2002] achieved no less than 30 frames per second and often up to 60 frames per second.

2.2.2 Implicit Integration

Baraff and Witkin [Baraff et al., 1998] introduced the idea of implicit integration into the cloth modelling world. This new method allows large time steps, and are able to handle large numbers of particles without much variance in the running times (ie: 5%). This method is largely used for accurate simulation of high resolution models, where the application is not intended to be real-time, namely animations.

Baraff and Witkin [Baraff et al., 1998] describe their integration as "backward" Euler, because it starts from the output state having the values for the position of the particle: $(x_0 + \triangle x)$ and $(v_0 + \triangle v)$ where x_0 and v_0 are the position and velocity at the start of the time step and $\triangle x$ and $\triangle v$ are the change is position and velocity during the step. It then uses a forward Euler step to run the system backward in time. This forces one to find an output state whose derivative points back to where the system started from at the beginning of the time step, which brings an extra level of consistency to the integration.

Choi and Ko [Choi et al., 2002] implement a variant of Baraff and Witkin's implicit integration, and demonstrate that they are able to use a time step of up to 100 seconds and still keep the system stable, even though the animation is choppy. However, when they use a more realistic time step, they are able to achieve most convincing results using a clothed animated character. Etzmuß et al. [Etzmuß et al., 2001] also use an implicit integration for their system, and are able to achieve realistic results.

3 Collision Detection

There are numerous collision detection algorithms in existence, but not all algorithms are suited to cloth modelling. Largely, collision detection algorithms have been designed for rigid body objects, whose geometry is fixed - meaning

the vertices will not move in relation to each other over time. Due to this fact, special considerations need to be taken to achieve an algorithm that will work with cloth or other deformable objects. Because of the dynamic nature of the geometry of cloth, an algorithm needs to be able to deal with the arbitrary object deformations that are present in deformable objects [Etzmuß et al., 2001].

3.1 OBB Trees and AABB Trees

Gao [Gao, 2001] describes an OBB-Tree an a hierarchical representation using oriented bounding boxes (OBBs). An OBB is a rectangular bounding box set at an arbitrary orientation in the 3D world. The ideal would be for the OBB to be oriented in such a way as to enclose an object an tightly as possible, so that the bounding box is as small as it can possibly be while still encapsulating the object it is supposed to be surrounding. An Axis-Aligned Bounding Box (AABB) is also a rectangular box in the 3D world, but, as its name suggests, instead of having an arbitrary orientation, the box is aligned to the axes of the 3d world. Logically, if the object that the AABB is trying to encapsulate is not aligned to the world's axes, the box is not going to be able to bound the object as tightly as it could if it was able to change its orientation. In general, AABBs have to use more bounding boxes in order to surround an object than OBBs, but are easier to implement.

OBBs and AABBs are known as bounding volumes (BVs), and are used to surround the objects that need to be tested for collisions. The way the trees work is that one BV is created to encapsulate the whole mesh. From this BV, smaller child BVs are created inside the parent, encapsulating different parts of the mesh the parent is surrounding. This process is repeated a number of times, and the tree is created. This means that a node in the upper level of the tree encloses the geometry covered by its child nodes in the lower levels. To test for a collision, all one needs to do is to check if any of the bounding boxes overlap. If no overlaps occur at a certain level of the tree, there is no need to check for overlaps in any of its children [Li et al., 1998]. When an overlap is found, a collision is found.

Bridson et al. [Bridson et al., 2002] use AABBs enlarged by the thickness of the cloth for their collision detection. These boxes are recalculated after each time step, and the hierarchy is recalculated every iteration of the collision detection algorithm. Although this creates a large processing overhead, real-time application was not their focus and their results are realistic and consistent.

Etzmuß et al. [Etzmuß et al., 2001] use Discrete Orientation Polytopes (k-DOPs) for their algorithm. k-DOPs are just another type of bounding volume like OBBs and AABBs, but are able to be adjusted to memory limits by the choice of the variable k. Their results are, while still realistic, not as good as those achieved by Bridson et al.

3.2 Octtrees

Baker [Baker, 1998] defines an octtree as a data structure which can define a shape in 3 dimensions. The object is surrounded by a box which is divided up into 8 smaller boxes. Each box contains a certain number of the object's vertices. Where more detail is needed in a box, that box is then divided up into eight more boxes. Then process can be repeated as many times as desired to provide the level of detail the object is wanted to be broken down into. This can then be represented as a tree where each node either has eight or no children.

Jakobsen et al. [Jakobsen et al., 2001] use an octtree approach for their collision detection. The results produced in their game Hitman: Codename 47 are believable if not completely realistic.

3.3 The Vironoi-clip (V-Clip)

This algorithm was invented by Mirtich [Mirtich, 1998] who claims it as a successor to the Lin-Canny closest features algorithm. The Lin-Canny algorithm had certain limitations, and the motivation for designing the V-Clip algorithm was to overcome these limitations. Mirtich claims that his algorithm is robust, significantly easier to code, and deals with the penetration case, where the Lin-Canny algorithm falls short. The algorithm works by tracking a pair of closest features (vertices, edges or faces) between two convex polyhedral objects moving through space. It uses the fact the the current closest features are probably near the previous closest features to achieve a near constant query time in practice. The distance between two polyhedral objects is easily found when the closest features are known. A collision is detected when this distance drops below a certain level or value. Since the V-Clip handles the penetration case, it is able to be extended to non-convex polyhedra when they are represented as groups of convex polyhedra [Goa, 2001].

4 Conclusion

Cloth modelling has been highly researched, and this survey has only touched the tip of the iceberg with regards to the amount of material that is available. However, a lot of the material consists of reworkings and minor tweakings of previous algorithms, and very few ground breaking algorithms have recently emerged. Therefore, one could make a decision on which algorithms to use and be fairly certain that (providing enough research has been done) the algorithms settled upon are going to be near the front of the pack.

All the research that I have done has revealed that the physical model that is most widely used is the particle system linked by springs (in one form or another). Thus, this is the model that I would recommend. Since the project that the decision for algorithms rests upon is intended for non-real-time application, performance will not be the most important criteria in the choice. With regards to what integration to use, as previously stated, implicit integration is

well suited for offline applications such as animation rendering, so I will be using the implicit integration described by Choi et al. [Choi et al., 2002] in their paper. Collision detection is a bit more tricky, since there are so many option available, but after examining the results produced by the various methods, I will be using the algorithm designed by Bridson et al. [Bridson et al, 2002], which is a variant of the AABB tree.

References

[1] Amirbayat, J., Hearle, J. (1989). *"The Anatomy of Buckling of Textile Fabrics: Drape and Conformability"*, Journal of Textile Institute, Volume 80, Issue 1, Pages: 51-70. Baker, M. J. (1998). "3D Theory - Octtree", Homepage. 30 May 2004. <http://www.euclideanspace.com/threed/solidmodel/spatialdecomposition/octtree/>.

[2] Baraff, D., Witkin, A. (1998). *"Large Steps in Cloth Simulation"*, International Conference on Computer Graphics and Interactive Techniques. Pages: 43-54.

[3] Breen, D. E., House, D. H., Wozny, M. J. (1994). *"Predicting the Drape of Woven Cloth Using Interacting Particles"*, Proceedings of SIGGRAPH 1994, ACM Press. Pages: 365-372.

[4] Bridson, R., Fedkiw, R., Anderson. J. (2002). *"Robust Treatment of Collisions, Contact and Friction for Cloth Animation"*, ACM Transactions on Graphics (TOG), Volume 21 Issue 3, July 2002. Pages: 594-603.

[5] Choi, K. and Ko, H. (2002). *"Stable but Responsive Cloth"*, ACM Transactions on Graphics (TOG), Volume 21, Issue 3, July 2002. Pages: 604-611.

[6] Eischen, J. W., Deng, S., Clapp, T. G. (1996). *"Finiter-element Modelling and Control of Flexible Fabric Parts"*, IEEE Computer Graphics and Applications, Volume 16, Issue 5. Pages: 71-80

[7] Etzmuß, O., Hauth, M., Keckeisen, M., Kimmerle, S., Mezger, J., Wacker, M. (2001). *"A Cloth Modelling System for Animated Characters"*, Universität Tübingen. 02 March 2004. <http://www.gris.uni-tuebingen.de/publics/paper/Etzmuss-2001-ACloth.pdf>

[8] Goa, J. (2001). *"Collision Detection"*, Stanford University. 25 May 2001. 30 May 2004. <http://www.stanford.edu/~jgao/collision-detection.html>.

[9] Gottschalk, S., Lin, M. C., Manocha, D. (1996). *"OBBTree: A Hierarchical Structure for Rapid Interference Detection"*, SIGGRAPH 1996.

[10] Jakobsen, T. (2001). *"Advanced Character Physics"*, Games Developers Conference (GDC). 24 March 2004. <http://www.ioi.dk/Homepages/tj/publications/gdc2001.htm>

[11] Li, T., Chen, J. (1998). *"Incremental 3D Collision Detection with Hierarchical Data Structures"*, ACM Symposium on Virtual Reality Software and Technology 1998.

[12] Mirtich, B (1998). *"V-Clip: Fast and Robust Polyhedral Collision Detection"*, ACM Transactions on Graphics, Vol. 17, No. 3. Pages: 177-208.

[13] Oshita, M., Makinouchi, A. (2001). *"Real-time Cloth Simulation with Sparse Particles and Curved Faces"*. 28 May 2004. <http://www.db.is.kyushu-u.ac.jp/~moshita/paper/ca01.pdf>.

[14] Rudomin, I., Castillo, J. L. (2002). *"Real-time Clothing: Geometry and Physics"*. 28 May 2004. <http://wscg.zcu.cz/wscg2002/Papers_2002/E31.pdf>