Interactive Animation of Cloth including Self Collision Detection

Arnulph Fuhrmann
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
afuhr@igd.fhg.de

Clemens Groß
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
cgross@igd.fhg.de

Volker Luckas
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
luckas@igd.fhg.de

ABSTRACT

We describe a system for interactive animation of cloth, which can be used in e-commerce applications, games or even in virtual prototyping systems. In order not to restrict the shape of the cloth we use a triangulated mesh, which serves as basis of a mass-spring system. For the animation of the particles internal and external forces are considered. Since cloth is a very rigid material when stretched, extremely large forces occur in such a system. Several methods have been described in the recent years to solve the underlying differential equations efficiently. We describe an algorithm which replaces the internal cloth forces by several constraints and therefore can easily take large time steps. These constraints can be parameterized to generate different behavior of the simulated cloth. Furthermore we provide an algorithm for an efficient avoidance of self collisions. For these reasons very plausible animations of cloth at interactive rates are possible with our system.

Keywords

real-time animation, cloth modeling, physically based modeling, interactive application

1. INTRODUCTION

Several cloth models have been proposed in the last decade. Some of them aim to reproduce the physical behavior of cloth [Bre94, Ebe96, Vol95, Bar98, Vol01]. Even more accurate ones are [Ebe00, Hau01, Cho02]. Others are able to generate approximate, but plausible animations in the context of interactive VR applications or games [Mey01, Kan01, Jak01, Kan02]. Some recent work [Cor02] combines fast geometric with physical methods for interactive animation of dressed humans. The model presented in this paper belongs to the category of approximate methods, since the computational efforts needed for accurate simulations are too large for real-time or interactive applications.

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972 *WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.*

In particular, when the piece of cloth should be able to move freely through 3D-space, some simplifications, which are possible, when animating dressed humans, can not be made.

One problem of cloth modelling is the efficient solution of stiff differential equations. These equations result from the formulation of internal forces, which are very large for most cloth, since it resists strongly against stretch. With explicit integration methods, like the one described in [Ebe96], the time step must be very small for the system not to diverge. This results in a large computational effort. Larger time steps can be taken by using an implicit method like the ones described in [Bar98, Hau01, Vol01]. But these methods are not able to compute real-time animations of cloth with a sufficient number of particles, because they require the solution of a linear system. Although this system is usually sparse, its solution with a modified conjugate gradient method takes nevertheless about $O(n^{1.5})$ time where n denotes the number of particles [Bar98].

In [Mey01] an approximate implicit method is described, which is able to animate systems composed of several hundred particles. The method uses a precomputed filter matrix of size $O(n^2)$ which is applied

at every time step. Although extremely fast for a small number of particles the advantage of this method against implicit methods vanishes for a larger number of particles. While experimenting with this method we noticed that after moving the cloth it lasted several seconds for it to reach its final position. This means, that their method is perfectly stable but the evolution of cloth motion is below real-time. This may be due to damping caused by the approximate implicit integration scheme. Similar experiences were also made by [Kan00, Kan01]. Therefore, this research group developed a method, which improves the work of [Mey01] by replacing the filter matrix with an update formula, which only considers neighboring particles for the filtering. This solves the problem of the $O(n^2)$ matrix, but since only direct neighbors are considered only a few mass points can be used. Therefore the authors added wrinkles and more details by using so-called wrinkled cubic spline curves for calculating in-between nodes. The wrinkles generated by this method are not physically motivated.

Recently, an innovative method was presented, which uses two layers of meshes — a sparse mesh for global motion and a fine mesh for more realistic appearance and details [Kan02]. This method is able to animate thousands of particles in real-time. But it does not handle self-collisions, which is an important part of cloth animation. Additionally, since it is an improvement of [Kan01], it can handle only several hundred particles in the sparse mesh. The particles in the fine mesh are influenced only by internal forces but not by external ones. In our approach internal and external forces are calculated for all particles. Furthermore, since there are only a few particles in the sparse mesh, it will be difficult to animate a highly curved piece of cloth, which is needed when the cloth falls onto the ground and piles up or when the user grabs it on one end only. The later is shown in figure 1.

Another kind of optimization is possible when animating dressed humans since most parts of a garment do not change position relative to the moving body [Cor02, Rud02]. In [Cor02] a system is described which allows real-time animation of complex dressed humans. To achieve this tremendous performance, the particles of the garments are divided into three layers. The ones belonging to the first and the second layer are treated mostly geometrically and only particles of the third layer are animated by physical methods. To get a further speed-up, no tests for self collisions are done. The technique for animating garments described in [Rud02] relies also on the presence of a human body. The particles are moved mainly according to the motion of the human body. Additionally, an explicit euler integration scheme is used for describing the interactions between particles. Because most

Figure 1: A round table top is hold at one end. In (a) and (b) the self collision detection is enabled. In (c) and (d) it is disabled. Self intersections are clearly visible. The view of (b) and (d) is from below.

of the motion of a particle is determined by geometric criteria, the system remains stable even for large time steps and consequently it is capable to animate garments in real-time. As in the system above, no self-collisions are handled. Since our system shall be able to animate cloth without any underlying body or other shape, both techniques can not be applied here.

Until now we did not discuss the problem of collision detection between the cloth itself and between the cloth and rigid bodies. In the area of cloth-object collision detection several methods were proposed in the last years. The most suitable ones for interactive applications are image space or voxel-based techniques [Zha00, Vas01, Mey01]. They tend to be much faster than classical approaches using bounding volume hierarchies [Bar98].

Recently an algorithm for robust treatment of collisions which also models cloth thickness was presented [Bri02]. The animations shown in the paper are really impressive and demonstrate the power of the algorithm. Unfortunately such a proper handling of self collisions is computationally too expensive for interactive applications. As mentioned above, none of the interactive cloth animation systems did implement self collision handling routines yet. Although some speedup techniques basing on the curvature of the cloth surface are known [Vol00, Pro97], self-collision handling is often dropped to get real-time performance.

2. OVERVIEW

In this paper we describe an algorithm, which is able to animate a piece of cloth in real-time. The cloth can be dragged interactively around using the mouse. During the animation, collisions with the environment and self

collision are handled. All particles of the cloth are influenced by internal and external forces. The novel aspect of our algorithm is, that we handle internal forces purely by constraints. They are derived from real cloth behavior and although they give a strong simplification of reality, different types of cloth can be visualized. External forces like gravity are applied normally and are integrated over time. This results in a system, which can be integrated explicitly and nevertheless is stable for large time steps. Additionally, self-collision treatment is integrated easily in this system. Nearby particles are kept away from each other by constraints similar to the ones used for the internal forces.

3. THE CLOTH MODEL

Currently, the standard approach to the physically based simulation of cloth is the use of particle systems. A piece of cloth is discretized into a set of particles. Our goal is the animation of arbitrary shaped cloth. Therefore we have chosen to use triangular meshes for representing the discretized cloth. Moreover it is possible to connect several pieces of cloth to form garments without any topological restrictions. The choice of triangular meshes has some influence on our system, but generally all methods described in this paper can be applied to rectangular meshes as well. The edges of the triangle mesh represent the structural springs and the vertices are the particles or mass points. Additionally bending springs are inserted between the two opposite particles of two adjacent triangles. Since the triangulation is not a regular one, not all particles have equal mass. The mass of one particle is computed by the following formula

$$\mathbf{m_i} = m_{pm} \cdot \sum_{\text{adj. triangles}} \frac{area_j}{3} \qquad (1)$$

where $area_j$ is the area of a triangle and m_{pm} is a cloth parameter which stands for the mass per square meter of the fabric. The idea behind this formula is to assign one third of the area of each adjacent triangle to each particle.

The movement of each particle is governed by the well known Newton's equation of motion

$$\mathbf{f_i} = m_i \cdot \mathbf{a_i} = m_i \cdot \frac{d^2\mathbf{x_i}}{dt^2} \qquad (2)$$

where $\mathbf{x_i} \in \mathbb{R}^3$ denotes the position of the particle, $\mathbf{f_i} \in \mathbb{R}^3$ the force acting on it, $\mathbf{a_i} \in \mathbb{R}^3$ the acceleration and m_i the mass of the particle. To solve equation 2 we have to determine the external and internal forces which are affecting the particle

$$\mathbf{f_i} = \mathbf{F_{int}} + \mathbf{F_{ext}}. \qquad (3)$$

Since the forces can by split into external and internal forces, we will describe first, how we handle internal forces. After that we will show how to deal with external forces and constraints. Finally, we will show, how to efficiently integrate the equation over time in order to yield the new positions of the particles.

Internal forces by constraints

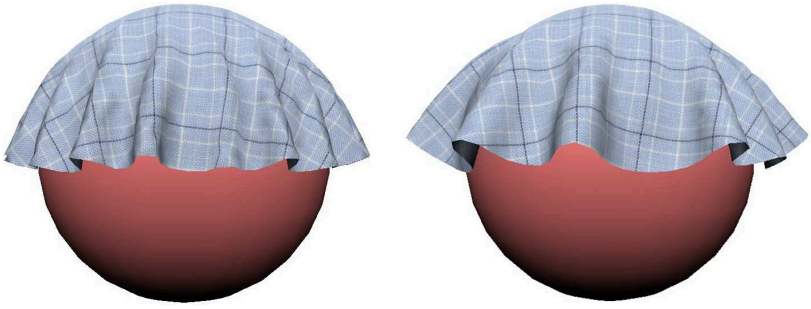
The way the particles interact with each other distinguishes the cloth models described in the literature. Tremendous effort has been done to develop models which reflect real cloth behavior [Bre94, Ebe96, Vol95, Bar98, Hau01, Cho02]. These models consider properties like shear, anisotropic stretch and bending. Our model is much simpler than those, but will reflect approximations to stretch and bending properties. Since we believe, that in most cases shear and anisotropic stretch can be neglected, we did not take them into account. Nevertheless, our model can be extended to handle these by using rectangular meshes instead of triangular meshes.

To justify our model we begin with reviewing shortly the most simple model – a mass-spring system – which is used in current interactive systems [Mey01, Kan02]. Interaction between linked particles is handled by linear springs. The resulting force is calculated by

$$\mathbf{f_{ij}} = k_{ij}(\|\mathbf{x_j} - \mathbf{x_i}\| - l_{ij}) \cdot \frac{\mathbf{x_j} - \mathbf{x_i}}{\|\mathbf{x_j} - \mathbf{x_i}\|}. \qquad (4)$$

where k_{ij} denotes the spring constant and l_{ij} is the rest length of the spring. Using linear springs is unfortunate since the stretch of cloth is nonlinear. It becomes extremely stiff after the elongation exceeds a certain threshold. One solution is to take very large spring constants and use an implicit method for solving the differential equations [Bar98, Vol01, Cor02]. Another, older approach is to use springs with a small k_{ij} and to apply a post step correction for springs which are overly elongated. This idea is after [Pro95]. The algorithm iterates over all springs and moves the two particles together along their common axis. Both particles are either moved by the same distance or, in the case of external constraints, only one of them is moved. This was followed and applied also in the context of an approximate implicit integration scheme [Mey01]. In [Vas01] the post correction of particle positions was converted into a correction of velocities, so that the over-elongation never occurs. Since our model, as described later, does not use velocities at all, we did not follow this approach. Another inspiring idea is after [Jak01], where each spring is modelled as a stick which does not change length at all. This results in a very stiff cloth which can by animated fast and robust in the means of stability.

Our method heads for a similar direction, but takes some steps further than the prior approaches. We simply set the k_{ij} to zero, i.e. we are not using any springs



(a) (b)

Figure 2: A tabletop consisting of 1600 particles draped on a sphere in real-time. The coefficients for the bending constraints are 0.89 for (a) and 0.99 for (b). The coefficients for structural constraints were not changed.

for our computation. Only the post correction as proposed by [Pro95] will take place. But now nothing hinders the cloth to collapse since no springs are active. Therefore we also add an additional constraint for too short springs.

To compute the new particle positions we determine in the first step the direction and length of the correction vector by

$$\mathbf{u} = \frac{\mathbf{x_j} - \mathbf{x_i}}{\|\mathbf{x_j} - \mathbf{x_i}\|}(\|\mathbf{x_j} - \mathbf{x_i}\| - d_{max} \cdot l_{ij}) \quad (5)$$

where $d_{max} \cdot l_{ij}$ is the maximal allowed length. By replacing d_{max} with d_{min} we get the formula for compressed springs. Due to our discretization all particles have different masses. Therefore we can not simply move both particles by $0.5 \cdot \mathbf{u}$ to fulfil the constraint because this would change the center of mass. If the correction vector is weighted by

$$m_1 = \frac{m_i}{m_i + m_j} \quad (6)$$

$$m_2 = \frac{m_j}{m_i + m_j} \quad (7)$$

and added according to

$$\mathbf{x_i} \mathrel{+}= m_2 \cdot \mathbf{u} \quad (8)$$

$$\mathbf{x_j} \mathrel{-}= m_1 \cdot \mathbf{u} \quad (9)$$

the center of mass stays constant. Furthermore it becomes easy to add external constraints by changing m_1 and m_2 to appropriate values. We refer to the next section for details.

Finally, the bending springs are replaced by constraints as well. They are handled like springs which are too short, but another parameter is used because bending is independent of compression. In practice, values from 1.0 to 0.8 give pleasing results. The choice depends on the desired bending rigidity of the simulated material. See figure 2 for a comparison between different materials.

Certainly, a single iteration over all the constraints – bending and structural – cannot be sufficient, since the alteration of the length of one spring affects neighboring springs, which in turn must be adjusted. But fortunately only a few iterations are needed in most cases. This depends mostly on the resolution of the cloth and the time step. Clearly, the adjustments are only local and therefore the information of a changed position is carried only to the direct neighbors in one iteration. To inform further particles another iteration is needed, i.e. the higher the resolution of the particle system the higher the number of iterations must be. We note, that the system remains stable even with high resolution and few iterations but only at the cost of visual quality.

External forces

There are several kinds of external forces which can influence the motion of the cloth. The most important are gravity, user input and collisions with other objects in the scene. The latter is described in section 4.

Gravity is treated as a normal force affecting each particle

$$\mathbf{f_i^g} = 9.81\mathbf{g}m_i \quad (10)$$

where $\mathbf{g}$ is a normalized direction vector which points to the ground. This force is integrated over time as described in the next section.

User input is done by grabbing some particle with a suitable pointing device. By moving it around the particle is also moved. Since it is desirable that the particle sticks to the pointing device as closely as possible we directly change the particle position. Because this change has priority over all other forces or constraints we set the mass of the particle to infinity. Clearly several particles can be constrained in this way.

For acceleration computations we use the concept of inverse masses as described in [Bar98]. The m_i^{inv}, which becomes zero when m_i is infinite, are used as

follows

$$\mathbf{a_i} = \frac{\mathbf{f_i}}{\mathbf{m_i}} = \mathbf{m_i^{inv}} \cdot \mathbf{f_i}. \quad (11)$$

This concept avoids the handling of different cases and therefore eases computation considerably. For the evaluation of Eq. 6 and Eq. 7 we simply set $m_1 = 0$ and $m_2 = 1$ if m_j is infinite. The effect is, that particle j is not moved at all, as intended. In the case that both masses are infinite m_1 and m_2 are both set zero.

Integration Method

The standard explicit euler integration yields the following two equations, which describe the motion of the particles over time

$$\mathbf{v_i^{n+1}} = \mathbf{v_i^n} + dt \cdot \frac{\mathbf{f_i^n}}{m_i} \quad (12)$$

$$\mathbf{x_i^{n+1}} = \mathbf{x_i^n} + dt \cdot \mathbf{v_i^{n+1}}. \quad (13)$$

This integration method can be implemented easily and computes efficiently the new particle positions, but unfortunately it is unstable for large time steps, when strong forces are involved. Since our method replaces internal forces by constraints and the external force of gravity is not high, this is no problem for our system.

Another problem are rapid changes in particle positions caused e.g. by collision response or user input lead to instabilities. To cope with this, [Mey01] proposed to correct the velocities after each time step. This is equal to correcting it before a time step by

$$\mathbf{v_i^n} = \frac{\mathbf{x_i^n} - \mathbf{x_i^{n-1}}}{dt}. \quad (14)$$

The effect of this equation is that the velocities are given only by the particle positions and therefore changes in position directly affect the velocity of the next time step. This increases the stability of the numerical integration. This is not a surprise, because this integration scheme is equal to the Verlet integration [Ver97] which updates the position without computing any velocities by

$$\mathbf{x_i^{n+1}} = -\mathbf{x_i^{n-1}} + 2\mathbf{x_i^n} + dt^2 \frac{\mathbf{f_i^n}}{m_i}. \quad (15)$$

This can be seen by substituting Eq. 12 into Eq. 13. This yields

$$\mathbf{x_i^{n+1}} = \mathbf{x_i^n} + dt(\mathbf{v_i^n} + dt \cdot \frac{\mathbf{f_i^n}}{m_i}). \quad (16)$$

By further substituting Eq. 14 into Eq. 16 and rewriting we get

$$\mathbf{x_i^{n+1}} = \mathbf{x_i^n} + \mathbf{x_i^n} - \mathbf{x_i^{n-1}} + dt^2 \cdot \frac{\mathbf{f_i^n}}{m_i} \quad (17)$$

which is equal to Eq. 15.

As described in [Hua02] the Verlet integration scheme is one of the best choices for problems with low or no damping. A damping term can be introduced in the system by multiplying 14 with an appropriate value. In our system we are able to set the damping to 1.00, i.e. no damping occurs. In order to simulate air drag, we choose an value of 0.99 in the examples shown, since it damps the movement of the cloth a bit, which looks more natural.

4. COLLISION DETECTION

Self-Collisions

As can be seen in Fig.1, a proper self-collision handling is mandatory for a cloth animation system. In general, there are two possibilities for handling self-collisions: Resolve them after they happened or never let them occur. Our approach belongs to the latter category.

We have decided to solve the problem approximately by not testing particles against triangles and edges against each other, but we consider only pairs of particles. Clearly, we now have to hold the particles a little bit away from each other to avoid artifacts of not detected intersecting triangles. But this approach saves a lot of computations. Since we are using a fine triangulation and the distance, which we have to preserve, depends on the size of the triangles, our approach is feasible.

In order to keep the particles apart, we simply add constraints between nearby particles. These constraints are like the structural ones used to avoid compression of the cloth. Due to its simplicity a single test can be implemented very efficiently. In particular, the common case, that the particles are not nearby, is carried out by a few arithmetic operations. Otherwise, the correction method described above is applied to the particles.

Nevertheless, a method for reducing the number of tests has to be applied, because the number of potential nearby particles grows with $O(n^2)$. A common speedup test is to build a bounding volume hierarchy of the triangles and to test the hierarchy against itself for collisions. Since we are testing only particles against each other the hierarchy bases on them not on triangles. The structure of the hierarchy does not change during animation and is computed from the flat cloth in advance. After each time step the hierarchy is updated from bottom to top. The bounding boxes are made larger to take into account the distance the particles should keep. Finally the hierarchy is traversed in order to find nearby particles.

We have to admit, that our method for self-collision avoidance can not compete with the more sophisticated ones described in [Pro97, Vol00, Bri02], because after a self-collision accidently occurs, it can not be repaired by our method. But as an approximate method for interactive VR applications, although not perfectly robust, it is extremely efficient. Furthermore, it enhances the quality of the animation tremendously in comparison to prior systems, which did not check self-collisions at all.

Cloth–Environment Collisions

We do not focus in this work on the field of collision detection between the cloth and a complex environment. For the examples shown, we implemented a fast and simple collision detection for spheres and axis aligned boxes. Since it is easy to calculate the distance to these primitives, our algorithm is based on particle-primitive distances and the corresponding normals.

The collision response works as follows. If a particle is detected to be closer to the primitive surface than a given threshold t it is set back in the direction of the normalized normal $\mathbf{n}$. Let d be the distance of the particle to the surface, then

$$\mathbf{r_n} = \mathbf{n} \cdot (t - d) \tag{18}$$

computes the normal component of our collision response. For an approximate modelling of friction we compute a tangential component $\mathbf{r_t}$ by scaling the tangential part $\mathbf{d_t}$ of the particle movement vector

$$\mathbf{d_t} = \mathbf{d} - \mathbf{n}(\mathbf{d} \bullet \mathbf{n}) \tag{19}$$

$$\mathbf{r_t} = -c_f \mathbf{d_t} \tag{20}$$

where $\mathbf{d} = \mathbf{x_i^{n+1}} - \mathbf{x_i^n}$ and c_f is the friction parameter (1.0 for friction cancels all the tangential movement). The final particle position is computed by

$$\mathbf{x_i^{n+1}} += \mathbf{r_n} + \mathbf{r_t}. \tag{21}$$

5. EXPERIMENTAL RESULTS

In order to demonstrate the capabilities of our animation algorithm, we show several examples. In Fig. 2 a comparison between different bending rigidities is made. By changing the value of this parameter we are able to animate different materials. Changing the coefficients of the structural constraints is limited since a value beyond 1.1 does not produce pleasing animations. They are the result of overly and uneven stretched triangles.

In Fig. 3 a sequence of an animated tabletop, which is taken from the accompanying video, is shown. It validates the soundness of our cloth model since the animation looks very natural and the user, moving the tabletop around, gets the impression of dragging real cloth around. The mesh of the tabletop consists of 1000 particles and the time-step dt is 0.0083, i.e. four simulation iterations per frame. The constraints describing the internal forces were applied four times per iteration. The other examples were made with the same time step, but for a higher number of particles more iterations are necessary.

The implementation was carried out in Java and uses Java3D for the visualization. The system runs on a a dual Intel Xeon^{TM} processor at 2Ghz. We did not yet parallelize our animation algorithm. Only the rendering routines of Java3D are using the second processor.

6. CONCLUSION

We have proposed an algorithm for animating cloth in real-time. Our algorithm is stable for large time steps. This is due to describing internal forces by constraints. Only external forces like gravity and the changes of velocities are integrated over time.

Furthermore, our algorithm produces realistic cloth motions even for cloth with low bending rigidity. This becomes possible since we include an algorithm, which avoids self-collisions in a very efficient but approximate way. Artifacts due to this approximation occur seldom. Nevertheless, there is room for improvement, since the particles have to be kept away from each other a quite considerable distance. Also, increasing the number of particles beyond 1600 slows down the algorithm too much due to self-collision treatment. In [Pro97, Vol00] methods basing on the curvature of the cloth are described, which further reduce the number of tests to the order of actual collisions. We did not follow these methods, since they bear a constant overhead, e.g. for computations of normals, which are not needed in our particle based approach. For animations of even larger meshes than the ones used in this paper a closer look at these methods is certainly necessary.

Moreover, we conclude, that our system is able to animate interactively cloth of different kinds of material quite well. Although our major concern was not physical correctness the animations look quite natural. One open problem is the mapping of measured material properties of real cloth to our model. Another problem is the handling of cloth which can be stretched for more than one tenth of its rest length, since in our system no in between forces are acting on the particles.

Finally, we are currently extending our cloth-environment collision detection algorithm to support more complex shapes than the ones shown in this paper.

7. ACKNOWLEDGMENTS

This work was partially supported by the bmb+f (Bundesministerium für Bildung und Forschung) Virtual Try-On grant. We thank our Virtual Try-On project partners at the University of Bonn for providing the textures used for rendering the cloth. We are very grateful to our partners at the University of Tübingen for helpful discussions about cloth simulation.

8. REFERENCES

[Bri02] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Transactions on Graphics (SIGGRAPH 2002)*, volume 21, 2002.

[Bre94] David E. Breen, Donald H. House, and Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. In *SIGGRAPH 94 Conference Proceedings*, Annual Conference Series, pages 365–372, Orlando, FL, USA, July 1994. ACM SIGGRAPH.

[Bar98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 43–54, Orlando, FL, USA, July 1998. ACM SIGGRAPH.

[Cho02] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. In *ACM Transactions on Graphics (SIGGRAPH 2002)*, volume 21, 2002.

[Cor02] Frederic Cordier and Nadia Magnenat-Thalmann. Real-time animation of dressed virtual humans. In *EUROGRAPHICS 2002*, volume 21, 2002.

[Ebe00] Bernhard Eberhardt, Olaf Etzmuss, and Michael Hauth. Implicit-explicit schemes for fast animation with particle systems. In *Eurographics Computer Animation and Simulation Workshop 2000*, 2000.

[Ebe96] B. Eberhardt, A. Weber, and W. Strasser. A fast, flexible particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–59, September 1996.

[Hau01] Michael Hauth and Olaf Etzmuß. A high performance solver for the animation of deformable objects using advanced numerical methods. In *Proc. Eurographics 2001*, 2001.

[Hua02] Michael Hauth, Olaf Etzmuß, Bernd Eberhardt, Reinhard Klein, Ralf Sarlette, Mirko Sattler, Katja Daubert, and Jan Kautz. Cloth animation and rendering. In *Eurographics 2002 Tutorials*, 2002.

[Jak01] Thomas Jakobson. Advanced character physics. In *Proceedings of Game Developers Conference*, USA, March 2001.

[Kan02] Young-Min Kang and Hwan-Gue Cho. Bilayered approximate integration for rapid and plausible animation of virtual cloth with realistic wrinkles. In *Computer Animation 2002*, page 203, Geneva, Switzerland, 2002.

[Kan01] Young-Min Kang, Jeong-Hyeon Choi, Hwan-Gue Cho, and Do-Hoon Lee. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer*, 17:147–157, 2001.

[Kan00] Young-Min Kang, Jeong-Hyeon Choi, Do-Hoon Lee, Chan-Jong Park, and Hwan-Gue Cho. Real-time animation technique for flexible and thin objects. In *WSCG 2000*, pages 322–329, Plzen, Czech., 2000.

[Mey01] Mark Meyer, Gilles Debunne, Mathieu Desbrun, and Alan H. Barr. Interactive animation of cloth-like objects for virtual reality. *The Journal of Visualization and Computer Animation*, 12:1–12, May 2001.

[Pro95] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface 95*, pages 147–154, 1995.

[Pro97] Xavier Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Graphics Interface 97*, pages 177–189, 1997.

[Rud02] Isaac Rudomin and Jose Luis Castillo. Real-time clothing: geometry and physics. In *WSCG 2002 Posters*, pages 45–48, Plzen, Czech., 2002.

[Ver97] Loup Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical Review*, 159(1):98–103, 1997.

[Vol95] P. Volino, M. Courchese, and N. Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 137–144, Los Angeles, CA, USA, August 1995. ACM SIGGRAPH.

[Vol00] Pascal Volino and Nadia Magnenat-Thalmann. *Virtual Clothing, Theory and Practice*. Springer, 2000.

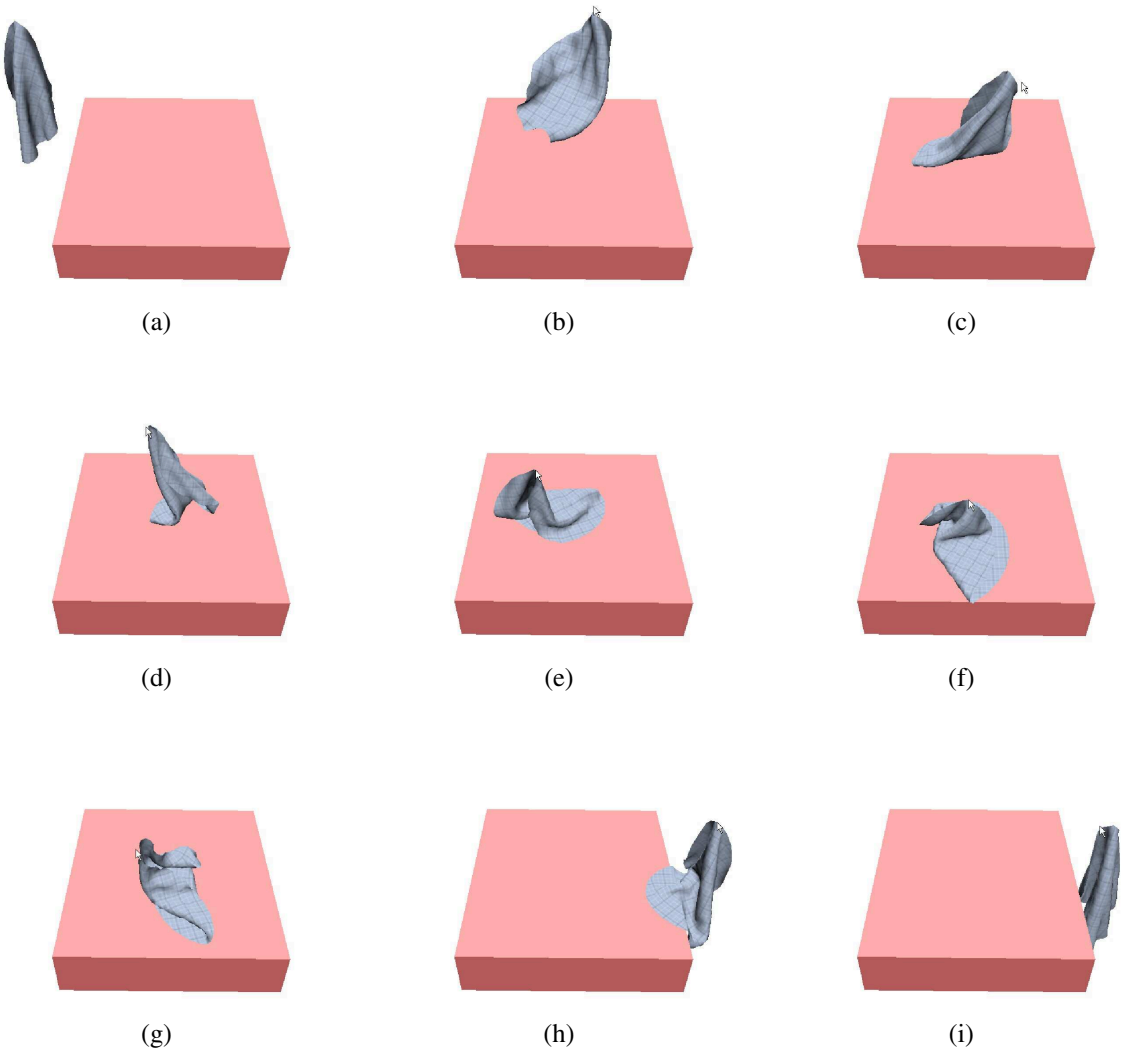


Figure 3: Sequence of an animated tabletop. It is interactively dragged around by the user at frame rate of 30Hz.

[Vol01] P. Volino and N. Magnenat-Thalmann. Comparing efficiency of integration methods for cloth animation. In *Proceedings of Computer Graphics International 2001 (CGI'01)*, 2001.

[Vas01] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. In *Proceedings of Eurographics 2001*, 2001.

[Zha00] Dongliang Zhang and Matthew M.F. Yuen. Collision detection for clothed human animation. In *Proceedings of the 8th Pacific Graphics Conference on Computer Graphics and Application*, pages 328–337, 2000.