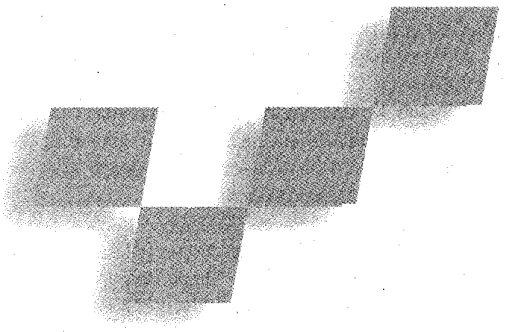
An Evolving System for Simulating Clothes on Virtual Actors



Pascal Volino and Nadia Magnenat Thalmann
MIRAlab, University of Geneva

Shen Jianhua and Daniel Thalmann
Swiss Federal Institute of Technology

This cloth-modeling and animation system aims to support virtual actors who can dress and undress themselves. It now creates autonomous clothes, independent of the synthetic human wearing them.

In most computer-generated films involving virtual humans, clothes are simulated as part of the body and have no autonomous motion. In recent years, however, software has been developed to design 2D garment panels interactively and to optimize the layout of the panels on fabric. For example, Hinds and McCartney[1] treated garment panels as surfaces of complex 3D shapes. They represented a static mannequin's body as bicubic B-spline surfaces, designed the garment panels around the body, then reduced the panels to 2D cutting patterns. By contrast, other computer graphics approaches use geometric methods to model the garment panels and then superimpose harmonic and sinusoidal functions to visualize folds and drapes.

Hinds and McCartney's work addressed garment design and manufacturing in industrial contexts. So did the work of Mangen and Lasudry.[2] They proposed an algorithm for finding the intersection polygon of any two polygons and applied it to the automatic optimization of laying out polygonal garment panels in 2D rectangular fabrics.

Modeling more realistic clothes requires solutions to two separate problems:

- cloth motion without collision detection and
- collision detection of the cloth with the body and with itself.

With regard to cloth motion, researchers have developed physically based models to animate cloth-like objects in many kinds of situations. (For a survey of research in cloth models, see the article by Ng and Grimsdale in this issue, pp. 28-41.) Object deformations can be represented by different geometrical models. Triangular grids are the most common, but polynomial surfaces and particle systems are also used for solving specific mechanical simulation problems. Geometric models yield nice-looking, accurate deformations, and they constrain both the initial shape and the allowed deformation.

With regard to collision detection and response, techniques have been developed to successfully stop cloth from penetrating the body and, less successfully, to prevent self-collisions between different parts of the cloth. This required solving the time-consuming problem of extracting the possible colliding elements from the whole set of elements composing the cloth and the body surfaces. Researchers have developed many techniques adapted for various surface representations. Unfortunately, these techniques are not well suited to efficient detection on deformable surface animations, as they require either expensive z-buffer rendering or construction of the objects' convex hull at each frame.

Evolution of clothing software

At the University of Geneva's MIRAlab and the Swiss Federal Institute of Technology's Computer Graphics Lab, we have been evolving cloth modeling and animation software that would ultimately allow virtual actors to dress and undress themselves. Figure 1 shows a frame from the "Flashback" sequence, which was our first approach to animating cloth on a synthetic actor.[3] We modeled the skirt by attaching a simple conic surface to the actor at the waist. Collision avoidance was implemented through a repulsive force field around the body. We achieved efficient computation by decomposing the body into segments and dressing each of them independently.

We have made several improvements to this model. The mechanical model, derived from Terzopoulos' elastic surface model,[4] was given more accurate damping using Raleigh's dissipative function.[5,6] Collision response was enhanced to simulate perfectly inelastic contacts.[7]

Even more important, we developed a panel-and-seaming system for building cloth. This system supports garment design using 2D polygonal panels, which are

0272-17-16/96/$5.00 © 1996 IEEE

then seamed together using forces that bring the panels together around the synthetic actor. The panels are kept together by geometrically forcing the seamed vertices together. A design interface was developed for the panels,[8] using an improved data structure that allowed complex panel shapes.

The 2D panel system provided very nice, realistic results for clothes worn by synthetic actors (see Figure 2). It basically opened the way for building clothes and dressing an actor by seaming 2D panels together—the same way real-life clothes are assembled.

New animation goals

We have maintained this 2D design interface while evolving the clothing system to meet new animation goals. Our early focus was on realistic cloth deformation and appearance on the animated actor. We next wanted to extend the simulation possibilities and animate cloth in diverse situations, such as folded, piled, or crumpled; worn in layers; and being put on or taken off by an actor.

These new situations required handling clothes more like an independent object than a set of panels linked to an actor's body. The cloth model would have to address new mechanical requirements, such as high deformation and bending. Collision and self-collision detection would have to work efficiently in multilayer or crumpled situations.

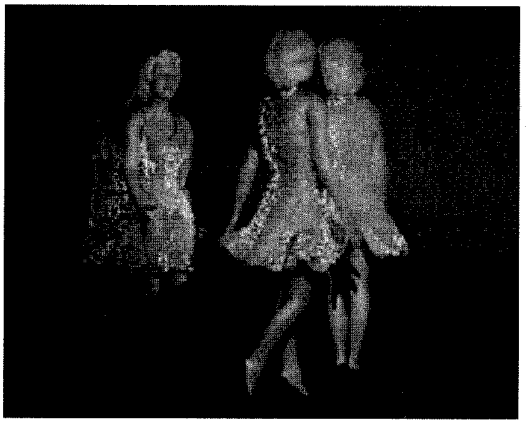
We also wanted to generalize the application range of the software, extending the simulation engine to animate a wide range of deformable objects simply by importing triangular meshes and giving them mechanical properties. The system could then include rigid object simulation as a new class of animated objects.

Another goal was to be able to dress any type of virtual human. This motivated us to improve the creation of new synthetic actors—a tedious and delicate task in the early software. It required the significant vertices defining the surface to be input by either 3D digitizing or interactive transformation of existing polygonal models. It was difficult to evolve a realistic surface across joints and relatively easy to produce surface singularities or anomalies. Moreover, thousands of vertices are needed to specify a reasonably detailed human body, which makes storage and communication inconvenient.

Improvements in the new software

To cope with these new goals, we completely rewrote the mechanical simulation procedures and collision-detection algorithms, and we adopted new human body modeling principles. Our main concern was to be able to handle clothes as objects independent of the underlying body. Thus, we changed the data structure. The simulation engine can now handle any triangular mesh surface, whether regular or not. We can still generate a garment by assembling panels, but when the panels are seamed together, the garment becomes a single and independent object. This clothing model uses collision response and attach points to interact with the body or other clothes, but when the garment is removed from the body, it continues to react on its own. For example, it can drop to the ground and crumple.

1 A frame from the "Flashback" sequence, animating a garment attached to a synthetic actor.



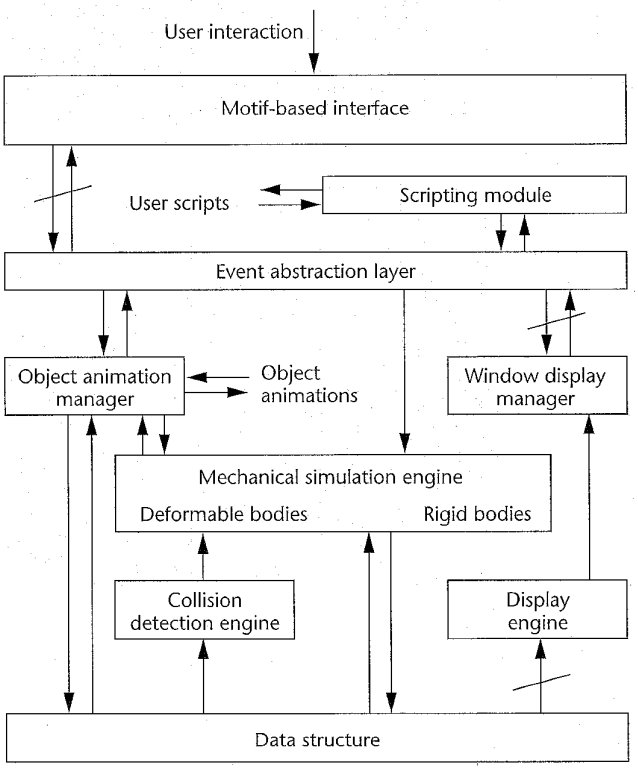
2 A dressed synthetic actor.

The primary difficulties generally encountered in cloth simulation are crumpling, high deformation, and collision interaction. We dealt with these problems as follows:

- A new collision-detection algorithm achieves high efficiency in detecting collisions, particularly self-collisions. This algorithm builds a hierarchy of the surface mesh and takes advantage of curvature regularity within surface regions for determining the possibility of self-collision within these regions.[9] With this algorithm, detecting self collisions is no longer a time-critical process and can therefore be performed as needed to accurately model cases with numerous self-collisions, such as crumpling.
- A modified mechanical model robustly handles cases with highly nonlinear deformations resulting from severe elongation and bending, even when discretization is irregular and rough. Rather than using global integration methods, such as finite elements in which nonlinearities and discontinuous behaviors cannot easily be integrated, we implemented a direct integration of Newton's second law. This approach opens up several possibilities, including the ability to act directly on position and speed (for direct and

interactive manipulation), and integrating highly nonlinear and time-varying behaviors (such as high deformations, collision response, and stability control).[10]

- We paid particular attention to efficient collision response. To handle complex situations with numerous interacting collisions—for example, crumpling—in an accurate and stable way, we chose not to use force feedback in computing response. Instead the system determines response directly, using position and velocity correction according to the mechanical conservation laws. Thus, we avoid the use of strong discontinuous fields requiring very small time steps for accurate handling, and all collisions can be handled independently of other mechanical computation within the same unaltered time step. The system then uses an iterative process locally on elements involved in several collisions. We obtain good stability with this technique, even in complex situations, and the global computation time is not severely affected by extra forces or the reduced time step otherwise created by collisions.
- We unified the data structure between all the different kinds of objects, increasing simulation versatility. The system can animate any object of the scene as a deformable surface, rigid body, precomputed animation, or immobile scene object. All these objects share the same file format.
- We completely rebuilt the interface to take advantage of all the possibilities brought by the new simulation engine.



3 The clothing simulation software modules.

These improvements mainly address versatility. They provide tools for simulating clothes and other deformable objects in many different conditions.

The new software also includes a facility for creating any kind of person to wear our clothes. This means we can dress men, women, and children of various sizes. This is part of a more general goal to make efficient, realistic human modeling and deformation capabilities available to the engineering and entertainment communities without the need for physical prototypes or scanning devices. We have developed a highly effective multilayered approach for design and animation of human bodies. It has three interrelated levels:

- An underlying articulated skeleton hierarchy, composed of only articulated line segments, on which the body movement is defined.
- Grouped volume primitives arranged in an anatomic-based approximation of the real body shape and attached to the proximal joints of the skeleton. By transforming and deforming those primitives, we can mimic the gross behavior of bones, fat, and muscles.
- The skin surface of the body, which is automatically derived from the position and shape of the first and second layer.

Our model can be easily integrated into existing environments, producing a body mesh with fixed topology. Since files for implicit models are typically at least two to three orders of magnitude smaller than those modeled with polygon or parametric patches, this model offers a compact, parameterized body, suitable for communication.

Description of the simulation software

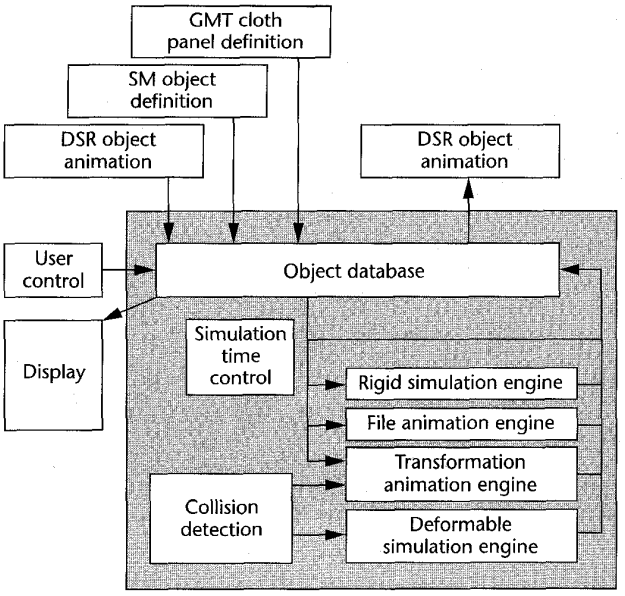
Figure 3 illustrates the overall software architecture. It has several parts:

- The data structure, storing all object information.
- The collision detection engine, which computes the collisions between all objects in the structure.
- The mechanical simulation engine, which animates objects according to different mechanical models for rigid or deformable objects.
- The software for human body modeling and deformation.
- Display modules, handling either window management or object display, which also provide feedback on user graphical interaction (selection, displacements).
- Object and event managers, which handle all object attributes and visualization parameters and process

the interface events and scripting.
- An X-Motif-based interface, giving users high-level interactive control.

The scripting system allows the program to run without any user interaction. The interface component, as well as the display components, may be suppressed, and the resulting program will run in a nongraphical context, allowing, for example, batch processing on remote sites.

Underlying principles of modularity and abstraction have guided development of the system architecture. For example, event abstraction allows us to substitute an interface system by modifying only a small number of high-level modules.



4 The simulation data flow.

Figure 4 illustrates the data flow, which is basically a computation loop where several animation engines animate their respective types of objects. The system detects collisions for all the objects and alerts the mechanical simulation engines to respond. The objects include deformable surfaces, rigid objects, animated sequences, transformation-based animations, and immobile scene objects imported in various formats (DSR polygonal surfaces, SM objects, GMT garment definitions) from various sources, including animations. The output is a collection of frames of the computed animation.

Mechanical simulation procedures

We designed the mechanical simulation system specifically for handling deformable surfaces in very severe mechanical situations, such as those encountered in crumpling surfaces. The new model improves on the version published in 1995.[10] We wanted the discretization to delineate severe bending and wrinkling yet remain coarse enough to simulate several garments concurrently with reasonable computation time. Furthermore, the system had to efficiently and robustly handle the numerous interacting collisions resulting from crumpling or multilayer cloth.

To comply with all these constraints, we based the model on direct integration of Newton's second law applied to a particle system model. This model allows us to treat each surface element independently, including independent manipulation for simulating position and speed constraints, and to handle linearities such as collision response as well. Furthermore, the model allows explicit and simple formulation of any kind of complex and nonlinear behavior, allowing precise modeling for big deformations.

We use various parameters to model the mechanical behavior of a fabric—mainly elasticity and thickness. We have added viscosity and relaxation times for both stretching and bending deformations to more accurately model damping and residual deformations. The new software also includes normal and tangent air viscosity.

The main mechanical parameters for collision response are geometrical surface thickness, static and dynamic friction, and bouncing elasticity. These parameters allow a quite satisfactory modeling of various materials (silk, fabric, leather, and so on) with different weights, elasticities, and thicknesses. However, the determination of these parameters is quite empirical and necessitates subsequent corrections using the "real" parameters measured from actual materials, since the rough discretization artificially "rigidifies" the structure.

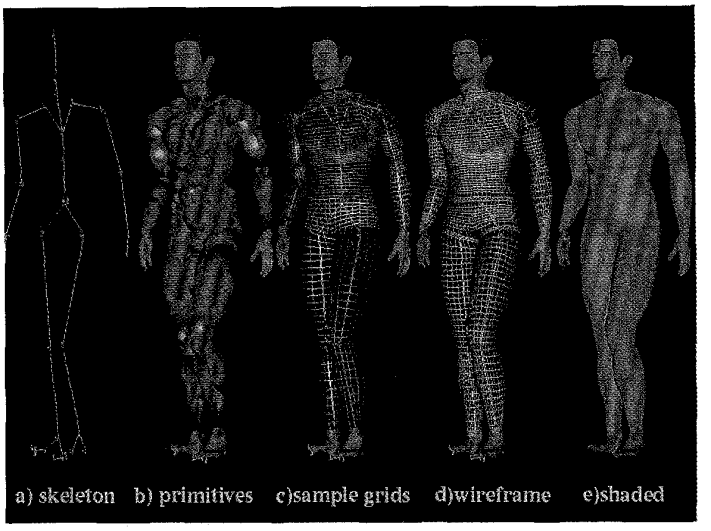
This model provides particularly efficient collision-response processing. After performing deformation calculations without accounting for constraints, a second step directly corrects the position and speed of colliding elements according to mechanical conservation laws. All collisions are thus processed independently from the deformation computation, leaving the time step unaltered. This technique also lets us avoid the use of strong and discontinuous force fields that alter the simulation efficiency.

Taking advantage of the model's flexibility, we implemented several other facilities, such as support for direct and interactive manipulation while the computation is running. This lets us interactively add "elastics"—attachment forces that bring together couples of vertices. Elastics are an efficient technique for adding attachments and seaming lines.

Collision detection

Collision detection is often the bottleneck of deformable surface simulation programs that handle highly discretized objects. This is particularly the case when complete self-collision detection is required, as in wrinkle and crumple situations.

Our program employs an efficient collision-detection algorithm, which we first presented in 1994.[9] It builds a geometrical surface region hierarchy and takes advantage of surface curvature within and between adjacent surface regions to optimize the detection of self-colli-



5 Layered human body model.

sions. The hierarchy is built once at the beginning of the simulation. After this, the algorithm recomputes only bounding boxes and normals for the hierarchy elements, and it performs the detection within an element or between two adjacent elements only if their curvature is compatible with the existence of collisions.

This algorithm is particularly efficient for huge and regular surfaces, independent of the refinement of the discretization. In normal cloth simulations, full self-collision detection accounts for less than 10 percent of the total collision-detection time. The algorithm lets the program use self-collision detection extensively without hypothesizing about which surface parts may collide. This offers great flexibility in handling the difficult situations involving complex wrinkling and crumpling.

Because we deal with both dressed bodies and wrinkling, there is no way to define an "inside-outside orientation" for all the simulated surfaces. Thus, we cannot say what side should be repelled from the surface in a collision.

Additionally, some situations can occur where interacting collisions and high deformation cause the collision response to fail temporarily in preventing surface crossover. However, in such cases, we wanted the system to be robust enough to correct the situation during the next simulation steps. To do this, we implemented extra algorithms for correcting collision orientations in case of crossover. They work by statistically analyzing the orientations of neighboring collisions. The simulation can thus recover from crossover situations, despite the lack of preset orientation information.

In combination with the new mechanical simulation engine, this collision-detection algorithm preserves its efficiency and robustness in most situations encountered in cloth simulation. (For more information about this algorithm, see Volino, Courchesne, and Magnenat Thalmann.[10])

Creating the bodies

Several researchers have devoted significant efforts to representing and deforming the human body shape.[11-15] As already mentioned, our body representation is based on three levels: skeleton, volume (muscles, fat, and bones), and skin, as shown in Figure 5. The volume primitives are based on *metaballs*, a particular subset of implicit surfaces.[16,17] An implicit surface is the set of points $\mathbf{x} \equiv (x, y, z) \in \mathbf{R}^3$ such that $f(\mathbf{x}) = 0$. Implicit surfaces are typically defined by starting with simple building-block functions and then creating new implicit functions using the sum, minimum, or maximum of the simpler functions. These surfaces can model and animate complex organic shapes at a fraction of the cost in data points of more common patching techniques. The final object is constructed by blending the primitives. As the primitives are moved and deformed, the resulting blended surface changes shape.

We categorize the volume primitives in our system as either *blendable*, which blend with other blendable volumes in the same group, and *unblendable*, which do not blend with other primitives. There is a trade-off between using complicated volume primitives or more simple ones. For simplicity and efficiency, we currently use only ellipsoids for unblendable volume primitives and isosurfaces with ellipsoidal density distribution for blendable volume primitives.

We associate each *deformable primitive* with a *reference joint*, whose value dynamically determines the center and shape of that primitive. When the underlying skeleton moves, all primitives attached to their relevant joints undergo the joint hierarchy transformations as rigid-body motions. Then the deformable primitives change their size, center, and orientations according to the current value of their reference joints.

In some sense, the human structure is a sum of different parts. Every part is composed of mass blocks, such as bones, muscles, and fat tissue. We build our human models according to this idea, considering a human body in several parts. In each part, we use volume primitives to approximate the shape of internal structures that observably affect surface form. We attach each primitive to its proximal joint, defined in the joint local coordinate system of the underlying skeleton, and assign each primitive to a group in accordance with the body part to which it contributes. Some primitives, located near a joint, can fall into several groups if they contribute to multiple parts.

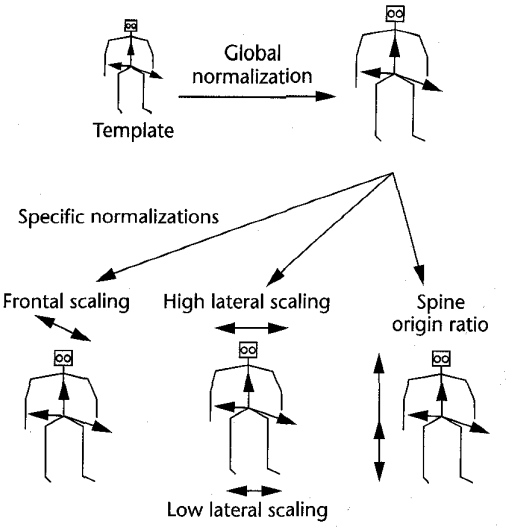
To obtain the skin, we sample an implicitly defined surface with ray-casting on semiregular cylindrical grids.[18] These sample points are used directly as cubic B-spline control points to smooth out the skin surface. We triangulate individual B-spline patches and stitch these triangular meshes together to connect different parts of the human body for final rendering and output.

To automatically generate human models with different sizes and proportions, we scale the standard

skeleton template to accommodate variations in age, sex, and race by using five normalized parameters (see Figure 6). Body scaling with these parameters is straightforward in our model. The x, y, and z axes of the global frame represent respectively the *lateral*, *frontal*, and *height* directions of the skeleton in a default posture. We associate a *tag* indicating the correspondence of each principal direction of a primitive to the frontal, lateral, or vertical direction of the skeleton, and then selectively scale the corresponding axis length and center according to the current skeleton ratio while keeping the same orientation. The tag of an ellipsoidal primitive is established by finding the dominant component of its principal directions in the global frame. That is, if $\mathbf{R}$ represents one principal direction in the global frame, then

$$\mathbf{R} \Leftrightarrow \text{lateral}, \quad \text{if } |\mathbf{R}_x| = \max(|\mathbf{R}_x|, |\mathbf{R}_y|, |\mathbf{R}_z|)$$
$$\mathbf{R} \Leftrightarrow \text{frontal}, \quad \text{if } |\mathbf{R}_x| = \max(|\mathbf{R}_x|, |\mathbf{R}_y|, |\mathbf{R}_z|)$$
$$\mathbf{R} \Leftrightarrow \text{height}, \quad \text{if } |\mathbf{R}_x| = \max(|\mathbf{R}_x|, |\mathbf{R}_y|, |\mathbf{R}_z|)$$

We can apply the five scaling operations successively to automatically generate a variety of human shapes as shown in Figure 7. Figure 8 shows a scene with a man, woman, and child generated this way.



6 Five scaling parameters for a skeleton: global, frontal, high lateral, spine origin ratio, and low lateral.

Clothing design and animation interface

The new clothing software is more than just a cloth program designed for dressing an actor. It is a complete and very general tool for performing rigid or deformable mechanical simulation on any kind of object, whether cloth or not.

Interface principle

The software is basically an object animator that simultaneously moves and deforms several objects concurrently with interactions such as collisions and attachments.

The software handles four types of objects:

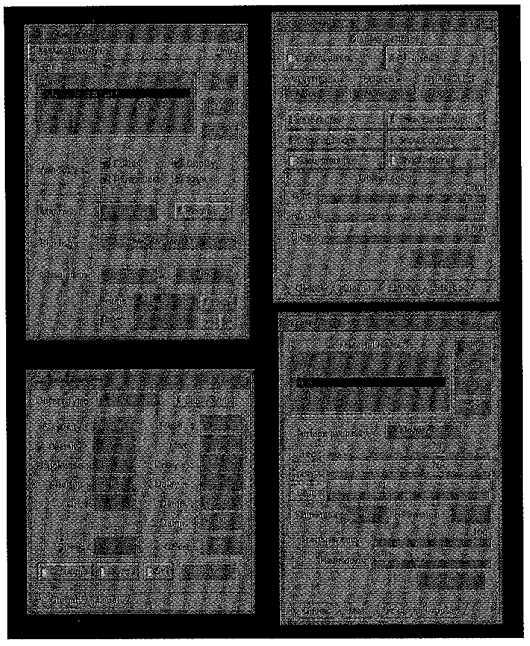
- Static objects.
- Animated objects, moved by either an animation sequence or a geometrical transformation.
- Rigid mechanical objects animated by mechanical computation.
- Deformable mechanical objects animated and deformed by mechanical computation.

Each object can be loaded from a different source in various file formats. All are concurrently animated with appropriate collision detection.

Animation is then run and mechanical computation is per-

7 Applying the scaling parameters successively generates a variety of human shapes.

8 Man, woman, and child in a museum. Their bodies were generated using the scaling operations.



9 The MainPanel window and some other control panel windows.

formed for the rigid and deformable objects. At each frame instance, the animated objects are saved with their respective frame numbers. User interaction, such as modifying parameters or manipulating objects, is possible at any time.

Garment design and the simulation process

Clothes are animated as any kind of deformable objects. They differ only by how they are constructed, that is, by assembling 2D panels. The clothing design and simulation process is divided into two parts:

- The 2D panel design process, which uses 2D drawing software to design garment models as flat fabric panels. Seams are defined around the borders of the panels.
- The 3D simulation process, which uses mechanical simulation to assemble the garment panels in the context where the animation will take place and then simulates the animated scene and characters.

The scene may contain several objects, static or animated, that will interact with the garments through collision. In particular, the actor to be dressed is typically an animated object.

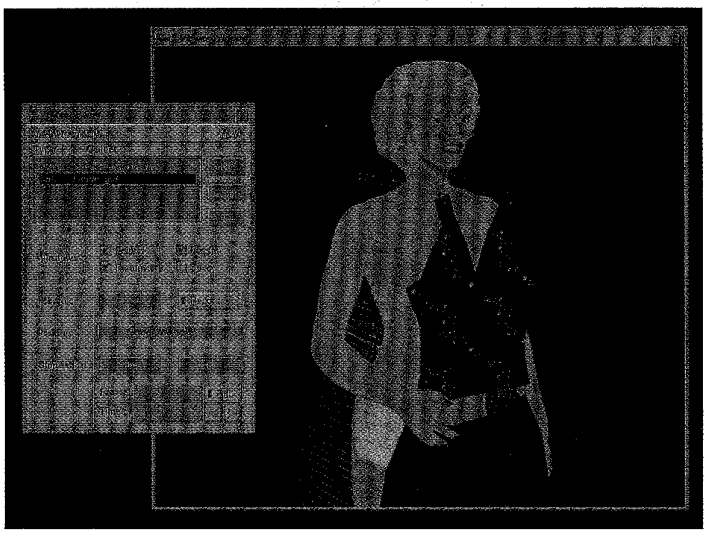
Garments are loaded from a file containing the 2D panel descriptions. The panels are discretized into triangle meshes and then interactively placed in an initial position suitable for seaming. When dressing an actor, the initial position is around the actor's body. Mechanical simulation is used to pull the panels together along the seaming lines using the "elastics" described earlier. These attachment forces pull together the corresponding vertices of both panels along the seaming line. Once the seaming lines are close enough, they are topologically merged and the panel set becomes one unique object. The object can then be handled independently and relies neither on the former panel definition nor on the body that supported it when it was built.

Once a garment is defined this way, the mechanical computation can proceed with the final animation. At any time, users can add extra elastics as attach points within the cloth or between the cloth and other objects, adding new design possibilities.

The program currently animates complex dressings containing several garments concurrently. Collision detection provides full interaction, and optimization for multilayer animated objects gives stability to the overall system. We also use incremental collision detection when relative movements are slow enough.

The system can record all operations performed during simulation in scripts, which can be executed for automatically setting up subsequent simulations. Scripts can also be organized into specialized libraries, providing tools for setting up materials, fabric types, simulation conditions, and so forth.

Finally, we record the animated garments frame by frame as animations that can be used as input data for subsequent computations. This allows incremental garment design for complex cloth.



10 Assembling cloth panels in the Viewer window.

We designed the interface to provide maximum interaction for controlling the simulation. The user can, at any time, block some points of the animated objects, add or remove "elastics," interactively move vertices of the deformable objects, and move whole objects.

To use the program requires at least two windows:

- The MainPanel window, shown in Figure 9, contains all the information representing the scene.
- The Viewer window in Figure 10 displays the objects in a 3D world.

In the Viewer window, the mouse is used to move the objects or the camera and to select objects or separate triangles, edges, or vertices in specific objects.

The MainPanel window includes a list of all objects in the scene that can be added or removed interactively with simple buttons. The MainPanel also gives access to a series of subpanels showing several parameters that can be changed interactively. We can specify parameters affecting:

- The scene (position of the camera, position and intensity of the lights, and so forth).
- The individual objects: (a) physical parameters such as elasticity, thickness, density, and rigidity, (b) geometric parameters such as position, orientation, and color, and (c) display parameters depending on the machine's capabilities, for example, show/hide the texture.
- Special "tools," for example, an elastic between two specific points belonging to distinct objects to force the points to approach each other (used for clothing animation).

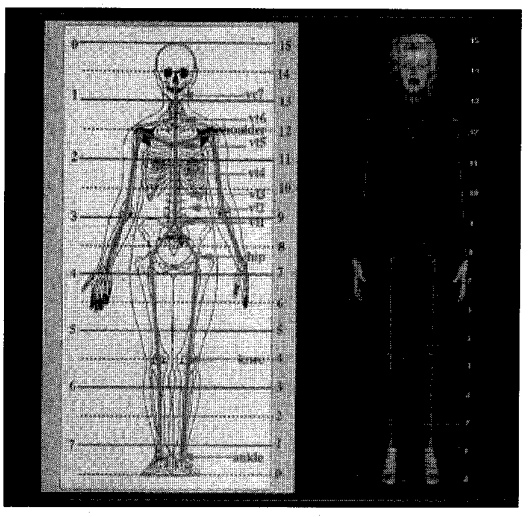
From the MainPanel, we can also record script files and launch a physical simulation of the scene. Once all objects are correctly placed according to their individual parameters, we can ask the computer to calculate the next frames according to the physical model of the scene.

The case of clothing animation requires two steps. First, we place the clothing patterns around the body, attaching them by elastic forces. Then we start the first simulation, which forces the patterns to fit the body and creates seams in the garment (Figure 10). Then we apply the desired deformations to the dressed bodies (for example, walking) and begin the second simulation, forcing the clothes to follow these deformations. Performance is highly dependent on the cloth complexity and the mechanical parameters, but a simulation involving one actor dressed in a typical garment usually takes from 30 seconds to 5 minutes per frame on a 150-MHz SGI Indigo2.

The body-creation interface

The new interface for creating bodies is composed of two parts, allowing the user to model the shape of the human body using metaball primitives and to scale the whole body in order to reproduce different body sizes.

The interactive primitive editor. The implicit surface technique is inherent to interactive design. You start with a rough shape consisting of just a few primitives, then add details by simply editing these primitives. For example, you can add, delete, transform, or adjust the parameters of primitives. We wrote an interactive primitive editor for shape design. The editor can display shaded or wireframe colored primitives and skin surfaces in near real time. It displays blendable primitives independently as ellipsoids with either the effective or the threshold radius. The "threshold" mode shows the visible size of a primitive, while the "effective" mode shows the influence range.



11 Proportion box mapping on the skeleton.

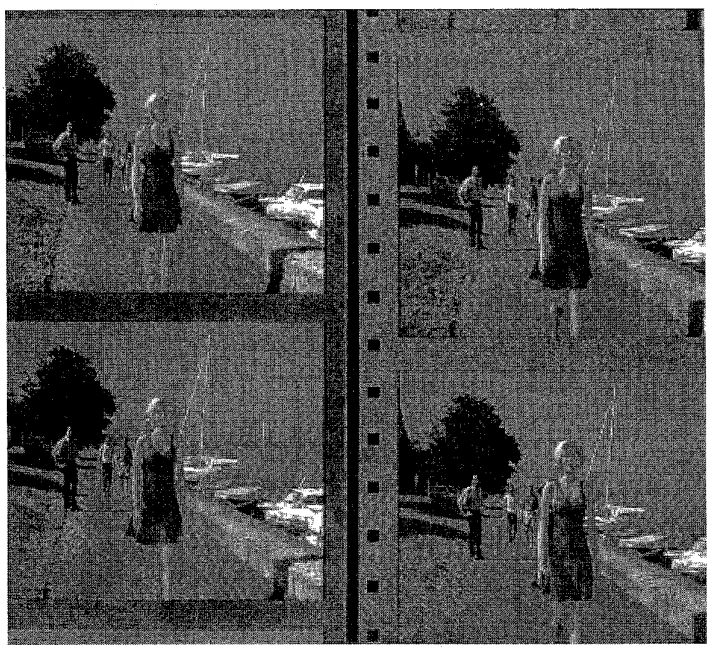
The interface uses widget panels to interactively adjust the size, weight, center, and orientation of the primitives. A primitive can be switched between the states of blendable or nonblendable and deformable or nondeformable. Different types of primitives are displayed with different colors. You can interactively create, delete, pick, join, attach or detach, and group primitives. By turning various display entities of different layers on or off, you can selectively check the skeleton, primitives, contours, and skin envelope simultaneously. A spaceball or trackball lets you rotate the model or camera in space for different viewing. You can get quick feedback of the resulting skin form during editing. Models can be saved into a file and loaded later for further sculpting.

Body design with the "proportion box." Users can create new models either from scratch or by modifying existing ones. They begin by specifying the skeleton height (in mm), followed by some scaling if necessary. A proportion system is very helpful in 3D design of a full figure. The interface includes a "proportion box" to guide both the construction of bony sketches and the development of fleshy masses and surface details, and to keep correct proportions between different parts of the body.

Proportions are defined relative to one unit of comparison, which must be defined. We use the head-length (from the crown to the tip of the chin) scheme proposed by Paul Richor,[19] which is widely used by artists. In Figure 11, the height of the figure equals 7.5 heads. Richor also describes different proportions for different genders and ages. We incorporated this scheme into our system. It is particularly valuable for designing aesthetic figures, and it lets us efficiently scale these "ideal" models to get individual variations.

Conclusion

The interactive editor, proportion box, and body-scaling feature provide a set of intuitive tools for animators to design a rich variety of human shapes, and we have shown how to dress them with autonomous clothes.



12 Dressed "Marilyn" walking in a real environment.

Figure 12 shows frames from an animation of our "Marilyn" model walking in a real environment. She was dressed using the system and techniques we have described here. The new software considerably improves system performance, versatility, and ease of use, and our goal of simulating actors that can dress and undress themselves is significantly closer. ■

Acknowledgments

We are grateful to Stéphane Carion, Mehdi Davary, and Jean-Claude Moussaly for their help in the production of some images. We also thank Marcia Riley for improving the English text and the referees for their constructive comments. This work is supported by the Swiss National Research Foundation.

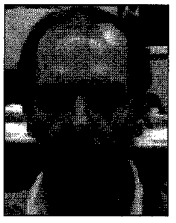
References

1. B.K. Hinds and J. McCartney, "Interactive Garment Design," *Visual Computer*, Vol. 6, 1990, pp. 53-61.
2. A. Mangen and N. Lasudry, "Search for Intersection Polygon of Any Two Polygons: Application to the Garment Industry," *Computer Graphics Forum*, Vol. 10, 1991, pp. 19-28.
3. B. Lafleur, N. Magnenat Thalmann, and D. Thalmann, "Cloth Animation with Self-Collision Detection," *Proc. IFIP Conf. Modeling in Computer Graphics*, Springer, New York, 1991, pp. 179-187.
4. D. Terzopoulos, J.C. Platt, and H. Bar, "Elastically Deformable Models," *Computer Graphics* (Proc. Siggraph), Vol. 21, No. 4, July 1987, pp. 205-214.
5. A.C. Eringen and E.S. Suhubi, *Elastodynamics*, Vol. 1, Academic Press, New York, 1974.
6. J.C. Platt and A.H. Barr, "Constraints Methods for Flexible Models," *Computer Graphics* (Proc. Siggraph), Vol. 23, No. 3, 1988, pp. 21-30.
7. M. Carignan et al., "Dressing Animated Synthetic Actors with Complex Deformable Clothes," *Computer Graphics* (Proc. Siggraph), Vol. 26, No. 2, July 1992, pp. 99-104.
8. H.M. Werner, N. Magnenat Thalmann, and D. Thalmann, "User Interface for Fashion Design," *IFIP Trans. Graphics Design and Visualization*, North-Holland, Amsterdam, 1993, pp. 197-204.
9. P. Volino and N. Magnenat Thalmann, "Efficient Self-Collision Detection on Smoothly Discretised Surface Animations Using Geometrical Shape Regularity," *Computer Graphics Forum* (Proc. Eurographics), Vol. 13, No. 3, 1994, pp. 155-166.
10. P. Volino, M. Courchesne, and N. Magnenat Thalmann, "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects," *Computer Graphics*, Vol. 29, 1995, pp. 137-144.
11. N. Magnenat Thalmann and D. Thalmann, "The Direction of Synthetic Actors in the Film Rendez-vous à Montréal," *IEEE CG&A*, Vol. 7, No. 12, Dec. 1987, pp. 9-19.
12. K. Komatsu, "Human Skin Model Capable of Natural Shape Variation," *Visual Computer*, Vol. 3, 1988, pp. 265-271.
13. D.R. Forsey, "A Surface Model for Skeleton-Based Character Animation," *Proc. Second Eurographics Workshop on Animation and Simulation*, Springer, New York, 1991, pp. 55-71.
14. J.E. Chadwick et al., "Layered Construction for Deformable Animated Character," *Computer Graphics* (Proc. Siggraph), Vol. 23, No. 3, July 1989, pp. 243-252.
15. M.P. Gascuel et al., "A Modeling System for Complex Deformable Bodies Suited to Animation and Collision Processing," *J. Visualization and Computer Animation*, Vol. 2, 1991.
16. J.F. Blinn, "A Generalization of Algebraic Surface Drawing," *ACM Trans. Graphics*, Vol. 1, No. 3, 1982, pp. 235-256.
17. J. Bloomenthall and K. Shoemaker, "Convolution Surfaces," *Computer Graphics* (Proc. Siggraph), Vol. 25, No. 4, 1991, pp. 251-256.
18. J. Shen and D. Thalmann, "Interactive Shape Design Using Metaballs and Splines," *Proc. Eurographics on Implicit Surfaces*, Springer, New York, 1995, pp. 187-196.
19. S.R. Peck, *Atlas of Human Anatomy for the Artist*, Oxford Univ. Press, Oxford, UK, 1982.

Pascal Volino *is a PhD student in computer science working at MIRAlab, University of Geneva, Switzerland. His research interests are in cloth animation models and focus on data structure, efficient collision detection, robust simulation, and interactive cloth manipulation. His work is part of the Espri Humanoid II European project to create and simulate virtual actors. Volino graduated in 1992 from Ecole Centrale de Lyon engineering school in Lyon, France.*

Nadia Magnenat Thalmann *is full professor of computer science at the University of Geneva, Switzerland, and adjunct professor at HEC Montreal in Canada. She leads a research group working on 3D computer animation, image synthesis, and multimedia systems at MIRAlab, University of Geneva. Magnenat Thalmann received a BS in psychology, an MS in biochemistry, and a PhD in quantum physics and computer graphics from the University of Geneva.*

Shen Jianhua *is a researcher in the Computer Graphics Laboratory at the Swiss Federal Institute of Technology in Lausanne. His research focus is on 3D character animation. His PhD on human body modeling is planned for October 1996. He has an MS in geometric modeling from Zhejiang University.*



Daniel Thalmann *is full professor and director of the Computer Graphics Laboratory at the Swiss Federal Institute of Technology in Lausanne. He is also adjunct professor at the University of Montreal, Canada. He received his diploma in nuclear physics and a PhD in computer science from the University of Geneva. Thalmann is coeditor-in-chief of the* Journal of Visualization and Computer Animation *and an editorial board member of several publications including* The Visual Computer *and the* CADDM Journal *of the China Engineering Society.*

Readers may contact Volino at 24, MIRAlab, University of Geneva, 24 Rue du General Dufour, CH-1211 Geneva, Switzerland, e-mail pascal@cui.unige.ch.