Volume 17 (1998), number 2 pp. 135–152 COMPUTER GRAPHICS forum

# Interactive Construction and Animation of Layered Elastically Deformable Characters

Russell Turner[1] and Enrico Gobbetti[2]

[1]UMBC, CSEE Department, Baltimore MD 21250, USA
[2]CRS4, Via Sauro 10, 09123 Cagliari, Italy
E-mail: turner@cs.umbc.edu, gobbetti@crs4.it

**Abstract**

*An interactive system is described for creating and animating deformable 3D characters. By using a hybrid layered model of kinematic and physics-based components together with an immersive 3D direct manipulation interface, it is possible to quickly construct characters that deform naturally when animated and whose behavior can be controlled interactively using intuitive parameters. In this layered construction technique, called the elastic surface layer model, a simulated elastically deformable skin surface is wrapped around a kinematic articulated figure. Unlike previous layered models, the skin is free to slide along the underlying surface layers constrained by geometric constraints which push the surface out and spring forces which pull the surface in to the underlying layers. By tuning the parameters of the physics-based model, a variety of surface shapes and behaviors can be obtained such as more realistic-looking skin deformation at the joints, skin sliding over muscles, and dynamic effects such as squash-and-stretch and follow-through. Since the elastic model derives all of its input forces from the underlying articulated figure, the animator may specify all of the physical properties of the character once, during the initial character design process, after which a complete animation sequence can be created using a traditional skeleton animation technique. Character construction and animation are done using a 3D user interface based on two-handed manipulation registered with head-tracked stereo viewing. In our configuration, a six degree-of-freedom head-tracker and CrystalEyes shutter glasses are used to display stereo images on a workstation monitor that dynamically follow the user head motion. 3D virtual objects can be made to appear at a fixed location in physical space which the user may view from different angles by moving his head. To construct 3D animated characters, the user interacts with the simulated environment using both hands simultaneously: the left hand, controlling a Spaceball, is used for 3D navigation and object movement, while the right hand, holding a 3D mouse, is used to manipulate through a virtual tool metaphor the objects appearing in front of the screen. Hand-eye coordination is made possible by registering virtual space to physical space, allowing a variety of complex 3D tasks necessary for constructing 3D animated characters to be performed more easily and more rapidly than is possible using traditional interactive techniques.*

**Keywords:** Character Animation, Physics-Based Animation, Deformation, Elasticity, 3D Interaction.

## 1. Introduction

Computer generated character animation remains an open research subject. While recent films such as *Toy Story, Jurassic Park*, and *The Lost World* have demonstrated that certain types of characters can be animated extremely well on the computer, traditional hand-drawn animation remains the most expressive and powerful way to bring characters to life, provides the greatest artistic flexibility, and can call upon thousands of years of artistic tradition. Nonetheless, traditional character animation has its limitations: it is highly labor intensive, it is difficult to make changes and see the results rapidly in an iterative creative process, and it requires considerable skill on the part of the animator to control perspective, rendering and animation simultaneously.

All of these limitations can be potentially overcome by using the computer as a creative tool for the character animator. The most advanced approach is to model the character directly in three dimensions as an articulated figure. Animation can then be accomplished by evolving the model's joint angles over time using a variety of techniques such as forward and inverse kinematics[1, 2], forward and inverse dynamics[3, 4, 5, 6], procedural modeling[7], motion capture[8, 9], and spacetime constraints[10]. Many commercial software systems (e.g. Alias/Wavefront, Softimage) and several in-house software systems (notably, Pixar's *Marionette*) allow the interactive creation and animation of 3D articulated characters. However, at least two major problems still prevent 3D character animation from being fully accepted by the animation community as a genuinely superior alternative to traditional animation. First is the difficulty of building good 3D character models that allow natural looking deformation of the skin surface as well as high-level animator control. Second is the lack of intuitive and powerful user interfaces that take full advantage of the animator's artistic real-world artistic technique.

The ideal 3D character model should provide a good compromise between high-level animator control and physically realistic behavior. One approach to this is to allow the animator to directly control the joint motion of the articulated figure through animating its joint angles, while the skin deformation is calculated automatically in a natural-looking way according to some small, intuitive set of skin behavior parameters. A variety of geometric modeling methods have been proposed for representing deformable animated characters, such as standard polygonal surface meshes, free-form deformations[11, 12], implicit surfaces such as soft objects[13, 14] and parametric surfaces such as tensor product splines or hierarchical B-splines[15, 16]. These purely geometric techniques are straightforward, provide ease of control and rapid computation but they have little relation to the physical reality of a flesh and blood creature, and therefore tend to lack realism. In particular, they tend to represent characters either as geometric surfaces, or as uniform solids, both of which ignore the complex internal structure of human or animal anatomy and are unable to provide dynamic deformation effects that are the hallmark of high-quality traditional animation. These kinds of effects are referred to by animators as squash-and-stretch, in which volume is conserved as a character undergoes deformation, and follow-through which describes the deformation caused by inertia when a character decelerates[2].

Physics-based deformable models[17], are derived from the elastic and viscous properties of continuous media and therefore can produce very realistic looking simulations of deformable materials. Like all physics-based models, however, they are usually difficult to control, so to be useful to animators they must be constrained properly[18, 19]. Because they represent continuous media as large numbers of discrete nodal elements, elastic models can also be very CPU-intensive, especially when simulating solids using three-dimensional lattices. However, elastic surfaces, simulated as two-dimensional lattices, require fewer numbers of discrete nodes to produce visually interesting results and therefore are not as demanding of CPU time. It is now possible, using high-end workstations, to simulate a reasonably complex surface of several hundred mass points at interactive rates.

The exact details of such a character model are no more important, however, than the types of interactive techniques used to construct and animate it. Ideally, the user interface for a computer-based artistic medium should be as natural and intuitive as a real-world medium. We believe the most promising approach to this for character animation is to provide a virtual environment in which the user can interact directly with a simulated character using 3D input devices. Since this requires the ability to simulate and render the animated character as faithfully as possible at interactive rates, a practical character model must provide a compromise between interactive speed and realism. We therefore believe that the most promising approach to modeling a 3D animated character is a hybrid one, in which layered models are constructed using a combination of geometric, kinematic and physics-based techniques. This model can then be simulated and rendered in real-time, using either a traditional or some type of immersive display, and the animator can interact directly with the character using various multi-dimensional input devices and interaction metaphors. In this kind of system, the sophistication of the character model allows the animator to focus on defining the evolution of a reduced number of key degrees of freedom, while complex and natural secondary effects are obtained through physical simulation.

The goal of this paper is to describe a working system that addresses both the modeling and simulation aspects of creating and animating characters on the computer. The system is based on earlier versions[20, 21, 22] to which we have added an improved character model, a completely new equation solver and performance evaluation. The system, called LEMAN (Layered Elastic Model ANimation system), uses a highly interactive direct manipulation user interface and a hybrid character model that combines an elastic simulation of skin deformation together with standard techniques for animating the articulated skeleton. For the remainder of this paper we will present the different aspects of LEMAN and demonstrate how it can be used to construct and animate simple characters as well as some limited forms of human animation. First, we give an overview of the layered approach to character modeling. Second, we discuss

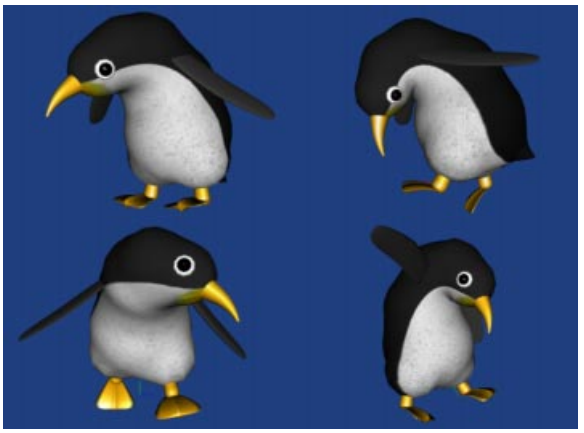

**Figure 1:** *Layered Elastic Character Deformation Under Various Skeleton Postures*

our elastic surface layer model and the techniques we use for physical simulation and for the numerical solution of the equations. Then, we present our 3D user interface and illustrate the construction of a simple character. Finally, we discuss the results obtained and provide a view of future work.

## 2. The Layered Approach

For designing animated characters, a common approach taken by artists and traditional animators is to work in layers. First a stick figure is drawn, representing the skeleton, followed by rounded forms to represent the flesh, followed by the finished outline, representing the skin[23]. This same sort of approach is taken in clay animation, where plasticene is wrapped around a metal armature.

### 2.1. Layered Models

It is not surprising that the first computer animated characters should also be constructed in layers. The film *Tony De Peltrie* used combinations of digitized facial expressions to deform a polygonal surface[24]. Implicit surfaces, called soft objects or blobbies, surrounding a stick figure skeleton have been used to create deformable characters[13]. Forsey used hierarchical B-splines with control points attached to a skeleton for modeling animals and human joints[25]. Shen combines these two approaches, wrapping NURBS surfaces around an internal layer of implicit surfaces[16].

One of the major advantages of layered computer models is that they allow the animation process to be divided into two stages: character construction, in which the behavior of the layers and attachment to the skeleton is defined, and character animation, in which only the skeleton motion is specified. The outer layers then derive all their input from the skeleton motion alone, greatly simplifying the animation process. One basic limitation with all of these character models is the absence of any physical basis for the model. Both the skeleton and the surface envelope are purely geometric models. Furthermore, outer layers are usually tightly bound to the underlying skeleton, preventing the skin from sliding along the underlying layers.

### 2.2. Layered Elastic Models

Layered elastic models add physics-based elastic components to some or all of the layers to improve realism. We use the term *elastic* to distinguish them from other deformable models that are not based on simulation of elastically-deformable materials. A simple example of this type of approach is Pacific Data Images' *Goop* system, in which a mass and spring with damping are attached to each vertex of a polygonal model[26]. Moving the model causes the vertex points to oscillate, causing a jello-like effect. However, the surface points are not attached to each other, so the skin has no surface-like physical behavior.

A more sophisticated examples of layered elastic construction for animated characters is found in the Critter system[27] in which a network of connected springs and masses is used to create a control point lattice for free-form deformations of the geometric surface. Some of the control points are bound to links of the underlying skeleton so that, when the skeleton is animated, the unattached mass points are influenced to move through the spring lattice. In this way, a physical simulation controls a solid deformation. Although the mass-spring lattice allows for shape control over the muscle deformation and a technique for bending at the joints, the skin is still fundamentally a geometric surface model, not a model of a physical skin. Similarly, a model for articulated figures is proposed by Gascuel and colleagues in which the control points for an interpolating spline surface are bound to a rigid bone layer by springs[28].

The finite element method is used by Gourret et al[29], who describe a human hand modeled as a volume element mesh surrounding bones, and Chen et al.[30], who have developed a biomechanically-based model of muscles on bone. These approaches focus more on obtaining realistic simulations, while being computationally too expensive to be effectively used for interactive character animation.

A sophisticated example of a layered elastic model is used by Terzopoulos and colleagues to implement facial animation[31, 32]. In this model, an elastic solid simulation, consisting of a mass-spring lattice of depth three, is attached to a human skull model and deformed by

muscles which take the form of force constraints between the skin surface and the underlying bone. The springs are biphasic to emulate the non-linear behavior of real skin, and volume preserving constraints simulate the effects of incompressible fatty tissue.

Many of these techniques are now becoming available in commercial character animation systems such as the Dynamation system of Alias/Wavefront's Maya product, which implements soft-body dynamics for characters using simulated damped springs attached to skin surface control points[33].

## 3. The Elastic Surface Layer Model

We have developed the elastic surface layer model in an attempt to improve realism of the skin surface and to find a practical compromise between purely kinematic and purely dynamic layered modeling approaches. By modeling the skin as an independent elastic surface and using geometric constraints to push it outside the underlying layers, we can achieve more realistic skin behavior in a practical interactive system for constructing and animating 3D characters. To explain how we do this, first let us step back and re-examine the fundamental layered character construction problem: modeling anatomy.

### 3.1. Modeling Anatomy

Human and animal character animation requires a careful study of anatomy and even the most stylized, animated characters still have an underlying structure which contributes to their outer shape and dynamics. How does one begin to try to construct a computer model of human or animal anatomy? Obviously, the anatomical figure is immensely complex, but it is possible to break its important components down into several well-defined layers that contribute to the overall visual appearance and behavior. Going from the inside out, these layers can be defined as: *skeleton, bone, muscle, fat*, and *skin*. The term *skeleton* refers to the purely stick-figure representation of articulated links and joints, while *bone* defines the geometric shapes of the bones. On top of the skin layer can be added *hair* (or *fur*) and *clothing*, if desired, although this will not be addressed in this paper. From the point of view of creating a computer model, each of these layers has distinct geometrical and dynamic properties which make it suitable for particular modeling techniques.

Bones, for example, can be regarded for animation purposes as rigid bodies, and their arrangement in a skeleton can be modeled very well as an articulated hierarchy of rigid bodies. Muscles are highly deformable and furthermore the only structure under active control, so physics-based models of muscle shape are probably too complex and computationally intensive for practical use. Geometrical models of deformable surfaces, with a few input parameters such as joint angle and tension, are probably the most appropriate in this case. The fat and skin layers, by contrast, are completely passive structures and therefore more amenable to physics-based simulation. The skin layer is characterized by being relatively thin and is therefore a good candidate for simulation using an elastic surface. The fat layer separates the muscle layer from the skin and can be defined by its thickness at each point on the skin.

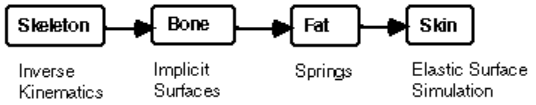


**Figure 2:** *Layered character animation pipeline*

### 3.2. The Character Animation Pipeline

An attractive aspect of this kind of layered breakdown is that, to a good approximation, each layer is dependent only on what is inside it, not the other way around. Therefore, it is possible to construct a character animation pipeline in which each stage of the pipeline adds another layer and can be modeled by a separate algorithm. For example, there exist a number of techniques for generating animated skeleton sequences. Output from an algorithm using any of these techniques can be used to derive the skeleton motion which can then drive the outer layers. This is not a perfect model, of course. In reality it is the muscles that drive the skeleton and not the other way around, skin, fat and muscle contribute mass which effects the dynamic motion of the joint, and collisions with other objects are transmitted back through the skin to the skeleton. Nonetheless, these situations can often be handled as special cases of feedback so that the pipeline model can make a good approximation under a variety of circumstances.

Another interesting aspect of the layered breakdown is that, with the exception of the underlying skeletal motion, each successive layer is more and more visible and therefore requires more realism. This suggests that it may be more profitable to concentrate computational resources on sophisticated physics-based models for the outer layers while implementing the inner layers using more efficient kinematic or geometric models. We have chosen to make the skin layer our starting point for a physics-based simulation not only because it is the most visible layer, but also because it can be represented fairly accurately as a two-dimensional surface which requires less computational resources for a given level of visual complexity than a three-dimensional solid.

This model of soft anatomy, in which all the mass

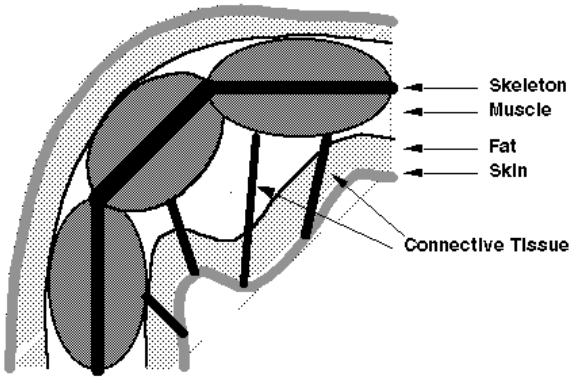


**Figure 3:** *Components of the Elastic Surface Layer Model*

is essentially concentrated at the surface, is able to simulate many types of skin-deformation effects such as wrinkling, and loose, floppy skin. While it may seem overly simplistic for modeling the overall three-dimensional shapes of soft-body masses, it is amenable to refinement by adding varying levels of physical realism to the inner layers, such as rigid body dynamics for the skeleton layer or volume-conservation constraints to the fat and muscle layers.

### 3.3. A Hybrid Model

The different characteristics of the different anatomical layers suggest a hybrid model in which different modeling techniques are used at each stage of the pipeline to form the final layered model. Each layer uses the type of modeling technique most suited to its characteristics. Figure 3 shows a diagram of the different components of the elastic surface layer model. We start with the outer skin layer which we model as a physics-based simulation of an elastic surface. We then work inward, considering each successive layer as a constraint acting on the layer outside it.

### 3.4. Skin Layer

The skin layer is at the starting point for the elastic surface layer model and is the only layer that is purely physics-based, using a simplified physical model of a continuous elastic surface[17].

The various physical parameters of the surface such as elasticity, rest metric, mass and damping, can all be specified at each point on the surface, allowing fine control of the intrinsic aspects of the surface behavior. Increasing the mass, for instance, increases the dynamic follow-through and squash-and-stretch effects of the skin, while increasing the damping density retards them. Globally adjusting the elasticity tensor affects the relative looseness or tightness of the skin, while selectively setting the elasticity tensor values at certain regions of the surface can simulate such effects such as wrinkles and tendons under the skin.

As the skin surface evolves over time, external forces can be applied to each point. These forces constitute the physical constraints which bind it to the underlying surface layers.

### 3.5. Fat and Connective Tissue Layers

The fat and connective tissue layers both separate and attach the skin to the underlying muscle and bone layers. The repulsive component is the fat layer which is specified simply as a thickness between the skin and muscle layers and is implemented using constraints to push the skin the required distance out from the underlying layers. This thickness can be specified at each point on the skin surface so that a considerable amount of sculpting of the final character's appearance can be performed simply by adjusting this parameter. The fat layer also provides a form of shape blending over the sharp corners of the underlying muscle layer. Increasing the fat thickness increases the amount of blending and helps to obscure any lack of realism in the underlying muscle model

The connective tissue can be thought of as rubber bands strung between points on the skin surface and on the muscle layer surface. They are implemented as Hookian spring force constraints acting between the two points. The spring constants of the rubber bands can be varied individually as well as their damping coefficients, allowing the degree of looseness or tightness of the skin attachment to be controlled. Also, various amounts of squash-and-stretch and follow-through effects of the skin can be controlled this way. An important parameter of the connective tissue layer is the exact binding points between the skin surface and the muscle layer surface. In section 6 we describe some interactive techniques we have developed for this purpose.

### 3.6. Muscle and Bone Layer

The muscle layer of the elastic surface layer model actually represents all of the solid components of the anatomy underneath the fat layer. Normally, this is primarily composed of muscle tissue, but where it comes close to the skin surface, it can also represent bone or cartilage. Bone and cartilage can be easily modeled as rigid, non-deformable solids. Muscles, however, are highly deformable solid shapes, whose elastic properties are dynamically changing under the active control of the nervous system.

Physics-based models of passive elastic materials are

therefore not necessarily the most appropriate technique to represent them. Also, since the shape of the muscle layer is somewhat obscured by the overlying layers, the need for a physically accurate model is not as important as for the skin.

We have therefore chosen to represent the muscle layer, as well as the underlying bone layer, by deformable geometrical solid surfaces which the skin may not penetrate. Geometric constraints are used to force the skin outside of the muscle layer surface, but leave it free to slide along the surface until it finds an energy minimum. Since, in the worst case, every point on the surface must be tested for penetration, it is important that the muscle geometric models allow for quick inside/outside tests. For this we use spheres and implicit surfaces such as super quadrics which have a simple inside/outside function, together with global deformation functions[34, 35]. The muscle shapes are attached as links to the skeleton joints so that they move as rigid components of an articulated figure. The flexing and bending of muscles is simulated by animating the parameters of the global deformations, either directly using key-frame interpolation, or by tying them to the joint angle values of the joints. While this is a rather simplistic model of muscle, more sophisticated physics-based or procedural models can easily be substituted, provided that they have an efficient inside/outside test algorithm, while leaving the bone layer as simple rigid bodies.

### 3.7. Skeleton Layer

We use the term skeleton in the computer animation sense of the word: a stick figure representing the positions and orientations of the joints which make up the articulated figure. The skeleton can be animated using a variety of techniques. Fortunately, the problem of skeleton animation is largely orthogonal to the problem of deformation, and therefore we can treat the problems separately. In particular, given our pipeline model, the skeletal motion completely controls the outer physics-based layers while the reverse is not true, so there are no complicating issues of feedback from skin deformation model to the skeleton motion.

## 4. Physical Model

The computation of the motion of the skin around the skeleton can be cast into a constrained dynamic simulation problem. The continuous skin surface is represented as a mesh of $P$ discrete 3D mass points, and the muscles on the skeleton are represented as a set of $Q$ solid shapes. The behavior of the sampled points that make up the skin mesh is obtained by evolving through time the constrained differential equation:

$$\mathbf{M}\frac{\partial^2 \mathbf{r}}{\partial t^2} = \mathbf{f}(\mathbf{r}(t), \mathbf{v}(t)) \tag{1a}$$

$$\forall i \in [1, P], j \in [1, Q] : g_j(\mathbf{r}_i(t)) \geq 0 \tag{1b}$$

where $t$ is the simulation time, $\mathbf{r}(t) = (\mathbf{r}_1(t)^\mathsf{T} \ldots \mathbf{r}_P(t)^\mathsf{T})^\mathsf{T}$ is the $P$-dimensional vector of mass point positions at time $t$, $\mathbf{v}(t)$ is the vector of mass point velocities at time $t$, $\mathbf{M}$ is the $PxP$-dimensional constant diagonal inertia matrix, $\mathbf{f}(\mathbf{r}(t), \mathbf{v}(t))$ is the vector of forces applied to mass points at time $t$, $g_k(\mathbf{r})$ is the inside-outside function the implicit surface representing muscle $j$. Note that $\mathbf{r}$, $\mathbf{v}$ and $\mathbf{f}$ are $P$-dimensional vectors containing 3-dimensional vector elements.

### 4.1. Force and Constraint Computation

The motion of the skin is determined by the forces and constraints applied to each of the discretized points, which describe both the elastic behavior of the skin material and the influence of the surrounding environment.

#### 4.1.1. Elastic Forces

To determine the elastic forces, the deformable surface is modeled as a three-dimensional function of a two-dimensional material coordinate system, referred to as the $u$-coordinate system. The total elastic energy of the surface can then be represented by a scalar-valued functional $\varepsilon(\mathbf{r})$. The elastic force at each point on the surface is represented by the variational[36] derivative $\frac{\delta\varepsilon(\mathbf{r})}{\delta\mathbf{r}}$. The fact that this local quantity depends on a functional of the entire surface is the reason this kind of elastic model is non-linear. Note that this is an equation in force per $u$-coordinate area, or force density.

The choice of functional, $\varepsilon(\mathbf{r})$, is determined by the elastic properties of the surface and choosing a good model of these properties can influence ease of solution as well as the quality of the simulated behavior. We use the elastic model proposed by Terzopoulos (see reference 17 for full details of the elastic force model).

Following Terzopoulos[17] we discretize equation (1$a$) using finite differences, into an $MxN$ rectangular mesh such that the $M$ and $N$ dimensions correspond to the $u_1$ and $u_2$ local coordinates respectively. We then obtain the approximation

$$\mathbf{f}^{elastic} = \mathbf{K}(\mathbf{r})\mathbf{r} \tag{2}$$

where $\mathbf{K}(\mathbf{r})$ is an $MxN$-dimensional sparse matrix computed from the elastic parameters, the mass point position vector $\mathbf{r}$, and the connectivity information used to estimate its partial derivatives with respect to the local surface coordinates using finite differences. We can affect the skin behavior by adjusting various intrinsic material properties associated with each point on the surface such as the rest metric tensor, which determines its desired size and the stretching elasticity, which determines its resistance to stretching.

#### 4.1.2. Connective Tissue

Each point on the elastic surface may be bound to a point on the underlying layers by a "rubber-band" which exerts a spring force on the surface point. These rubber-bands simulate the connective tissue which binds skin to muscle. The force equation for each rubber-band can then be written as:

$$\|\mathbf{l}_i\| > l_i^0 : \mathbf{f}_i^{tissue} = -\left[k_s\left(\|\mathbf{l}_i\| - l_i^0\right) + k_d\left(\dot{\mathbf{l}}_i \cdot \frac{\mathbf{l}_i}{\|\mathbf{l}_i\|}\right)\right]\frac{\mathbf{l}_i}{\|\mathbf{l}_i\|} \tag{3a}$$

$$\|\mathbf{l}_i\| \leq l_i^0 : \mathbf{f}_i^{tissue} = 0 \tag{3b}$$

where $\mathbf{l}_i$ is the difference vector from the fixed point to the skin surface point, $\dot{\mathbf{l}}_i$ is the difference between the fixed point velocity and the skin surface point velocity, $l_{ij}^0$ is the initial length of the spring, $k_s$ is the spring constant and $k_d$ is the damping coefficient. By tuning the parameters of the springs ($k_s, k_d$ and $l_{ij}^{(0)}$), the animator can fine-tune the behavior of the skin. High values of $k_s$, for example, result in skin that clings tightly to the skeleton, while low values result in loose, floppy skin that hangs down below the skeleton under the influence of gravity. Adjusting the values of $k_d$ controls how fast the skin follow-through dies down.

#### 4.1.3. Gravity and Global Damping

Gravity and global damping forces are the remaining two forces that act on the surface. These two forces are simply

$$\mathbf{f}_i^{gravity} = \mathbf{M}\mathbf{g} = \mu_i \mathbf{g} \tag{4}$$

$$\mathbf{f}_i^{damping} = -\gamma \mathbf{v} \tag{5}$$

where $\mathbf{g}$ is the acceleration of gravity in the appropriate direction and $\gamma$ is a global damping factor.

#### 4.1.4. Non Penetration Constraints

During simulation, the skin must be maintained outside the bone and muscle layers at all times. Instead of modeling non-penetration through reaction or constraint forces[18, 19], we model this behavior using a two-step process, first evolving the unconstrained ODE (1*a*) for a small timestep before directly correcting the state of the system to enforce possibly violated constraints (1*b*). The constraint resolution technique is detailed in the section on numerical integration.

Constraints are defined by the implicit inside-outside functions $g_k(\mathbf{r})$ that determine the shape of the muscles (currently spheres or superquadrics with local and global deformations[34, 35]). To speed-up the inside-outside tests, the mesh surface is traversed hierarchically using a quadtree scheme in which a lower-resolution subset of the surface points is selected initially for the inside-outside tests, from which a higher-resolution subset is determined according to a distance threshold. To model fat thickness at a point $\mathbf{r}_i$ we actually apply the constraints to the corresponding point on the inside of the fat layer

$$\mathbf{r}_i^{fat} = \mathbf{r}_i - \tau_i \hat{\mathbf{n}}_i \tag{6}$$

where $\tau_i$ is the local fat thickness and $\hat{\mathbf{n}}_i$ is the skin surface normal at $\mathbf{r}_i$.

### 4.2. Numerical Integration

Animation is obtained by kinematically controlling the skeleton while concurrently evolving equation (1) though time.

#### 4.2.1. Integration Technique

The choice of the integration technique is guided by the need to find a compromise between two conflicting criteria:

- **High Simulation Rate Criterion:** since force evaluations are time-consuming, the integration technique should try to minimize them, avoiding the use of exceedingly small simulation timesteps and requiring a small number of force evaluations (ideally one) per step;
- **High Interaction Rate Criterion:** since the system is used in an interactive context, with a user applying forces at unpredictable times and requiring low latency and high bandwidth feedback, the maximum simulation timestep that can be taken and the amount of CPU-time that can be spent to advance the system of one single step are severely limited.

We have implemented a differential equation library that makes it possible to switch at runtime between a variety of implicit and explicit solution techniques, including Euler (first-order, implicit and explicit), Verlet leapfrog (second-order, explicit), Rosenbrock (fourth-order, implicit), and Runge-Kutta (second-order and fourth-order, explicit).

While implicit techniques are generally more stable and permit bigger stepsizes, the time spent in the construction and solution of linear systems of equations is too big, at least in our implementation, to meet the high simulation rate criterion. Among the explicit techniques, we have obtained the best results with the second-order Verlet method[37, 38], modified to take into account velocity-dependent forces[39]. The method is explicit and self-starting, evolving the state of the system from a time $t$ to a time $t+h$ starting from initial conditions $(\mathbf{r}_t, \mathbf{v}_t)$ according to the following procedure:

$$\mathbf{r}_{t+h/2} = \mathbf{r}_t + (h/2)\mathbf{v}_t \tag{7a}$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t \tag{7b}$$

***loop***

$$\mathbf{v}_{t+h/2} = (\mathbf{v}_{t+h} + \mathbf{v}_t)/2 \tag{7c}$$

$$\mathbf{a}_{t+h/2} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{r}_{t+h/2}, \mathbf{v}_{t+h/2}) \tag{7d}$$

$$\mathbf{v}^{(old)}{}_{t+h} = \mathbf{v}_{t+h} \tag{7e}$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t + (h/2)\mathbf{a}_{t+h/2} \tag{7f}$$

***until*** $\|(\mathbf{v}^{(old)}{}_{t+h} - \mathbf{v}_{t+h}) * \mathbf{v}^{(scale)}\|_\infty \leq \varepsilon^{(v)}$ (7g)

$$\mathbf{r}_{t+h} = \mathbf{r}_{t+h/2} + (h/2)\mathbf{v}_{t+h} \tag{7h}$$

where $\mathbf{v}^{(scale)}$ is a scaling vector used to normalize errors, $\varepsilon^{(v)}$ is a user defined tolerance for the estimation of velocities, and "$*$" performs componentwise multiplication. To provide the user with control on both absolute and relative error, $\mathbf{v}^{(scale)}$ is defined as:

$$\mathbf{v}^{(scale)} = \begin{bmatrix} 1/\max(v_{11xt}, v^{threshold}) \\ \mathtt{M} \\ 1/\max(v_{MNzt}, v^{threshold}) \end{bmatrix} \tag{8}$$

where $v^{(threshold)}$ defines the maximum magnitude of a velocity component after which error control switches from absolute to relative[40]. Since small adaptive timesteps are used, the fixed point iteration (7c)–(7g) for the velocity estimation converges very rapidly. Because $\mathbf{r}_{t+h/2}$ remains constant during the iteration, only velocity-dependent forces have to be recomputed at each step.

#### 4.2.2. Stepsize Control

Regardless of the type of integration technique used, variable stepsizes are needed in our type of applications because of the presence of very large external and internal forces applied at unpredictable times during interaction. Using fixed stepsizes would force the system to make worst case assumptions, selecting stepsizes which are excessively small for most of the simulation, thus degrading the overall performance of the system.

To control the stepsize, we estimate the accuracy of the solution $\mathbf{r}_{t+h}$ by comparing it with the alternate solution given by embedded lower-order formulas[40]. In case of the Verlet method, the embedded formula is simply the explicit Euler method:

$$\mathbf{r}^{(Euler)}{}_{t+h} = \mathbf{r}_t + h\mathbf{v}_t \tag{9}$$

Since

$$\mathbf{r}^{(Euler)}{}_{t+h} = \mathbf{r}^{(*)}{}_{t+h} + O(h^2) \tag{10a}$$

$$\mathbf{r}_{t+h} = \mathbf{r}^{(*)}{}_{t+h} + O(h^3) \tag{10b}$$

we can estimate the truncation error by

$$\Delta = \|(\mathbf{r}^{(Euler)}{}_{t+h} - \mathbf{r}_{t+h}) * \mathbf{r}^{(scale)}\|_\infty \tag{11}$$

where $\mathbf{r}^{(*)}{}_{t+h}$ is the unknown exact position vector at $t+h$, and the scaling vector $\mathbf{r}^{(scale)}$ is determined from $\mathbf{r}_t$ and the user-defined parameter $r^{(threshold)}$ by the same technique used for $\mathbf{v}^{(scale)}$. The next stepsize $h_1$ can then be determined by

$$h_1 = h(\Delta/\varepsilon^{(r)})^{-1/2} \tag{12}$$

where $\varepsilon^{(r)}$ denotes the user-specified desired accuracy. If $\Delta$ is smaller than $\varepsilon^{(r)}$ the step is accepted and $h_1$ is used for the next step, otherwise the step is rejected and the smaller stepsize $h_1$ is used to repeat it[40].

#### 4.2.3. Enforcing Constraints

After a step has been accepted, the system has to enforce all constraints in (1b) that have been violated. This is done by directly moving any points inside the muscle shapes to the surface, and by adjusting the velocities to simulate an inelastic collision. For each point $(i, j)$ violating a constraint $g_k$, we thus compute the new state of the system by

$$\mathbf{r}_{ij}{}^{constrained}{}_{t+h} = \mathbf{r}_{ij}{}^{unconstrained}{}_{t+h} + \lambda_g \nabla \mathbf{g}_k \tag{13a}$$

$$\mathbf{v}_{ij}{}^{//}{}_{t+h} = (\mathbf{v}_{ij}{}^{unconstrained}{}_{t+h} \nabla \mathbf{g}_k) \nabla \mathbf{g}_k \tag{13b}$$

$$\mathbf{v}_{ij}{}^{\perp}{}_{t+h} = \mathbf{v}_{ij}{}^{unconstrained}{}_{t+h} - \mathbf{v}_{ij}{}^{//}{}_{t+h} \tag{13c}$$

$$\mathbf{v}_{ij}{}^{constrained}{}_{t+h} = -k^{//}\mathbf{v}_{ij}{}^{//}{}_{t+h} + k^{\perp}\mathbf{v}_{ij}{}^{\perp}{}_{t+h} \tag{13d}$$

where $\mathbf{r}_{ij}{}^{unconstrained}{}_{t+h}$ and $\mathbf{v}_{ij}{}^{unconstrained}{}_{t+h}$ are the position and velocity computed during the integration step, $\lambda_g$ and $\nabla \mathbf{g}_k$ are the distance and the unit gradient to the constraint surface evaluated at $\mathbf{r}_{ij}{}^{unconstrained}{}_{t+h}, \mathbf{v}_{ij}{}^{//}{}_{t+h}$ and $\mathbf{v}_{ij}{}^{\perp}{}_{t+h}$ are the parallel and perpendicular component of the velocity with respect to the constraint gradient, and $\mathbf{r}_{ij}{}^{constrained}{}_{t+h}$ and $\mathbf{v}_{ij}{}^{constrained}{}_{t+h}$ are the corrected position and velocity components.

To control the error introduced during constraint resolution, we limit the magnitude of the correction of each position component to be smaller than a user specified threshold $\varepsilon^{(c)}$. If the threshold is exceeded, the step is rejected and a new step with stepsize $h_c < h$ is attempted. $h_c$ is determined by a standard root bracketing procedure so that $g_k(\mathbf{r}_{ij}(t + h_c)) \geq 0$ for all constraints violated at time $t + h$. The process is similar to the collision resolution techniques used in rigid-body simulation systems[41]. Alternative techniques to enforce non-penentration are reaction[18] or Lagrangian constraints[42]. However, reaction constraints, while having computational costs similar to the technique outlined here, have the important drawback that violated constraints require multiple timesteps to be resolved. Lagrangian constraints solve this problem, but are more expensive to compute, requiring the computation of constraint Jacobians and the solution of a linear system.

## 5. A User Interface for Character Construction

Character construction is an important creative aspect of physics-based layered character animation since so much

of the character's animated behavior is determined by its structure. We therefore focus our efforts on improving the user-interface for this task, which, given the structure of the layered model, is mainly concerned with providing effective ways to perform the following operations[21]:

- **Skeleton building:** as a first step, a hierarchical skeleton must be created. This requires specification of all the joints in the articulated figure, their positions and orientations with respect to each other, and their hierarchical relationships. In addition, the joints' range of motion must be specified, which often requires manipulating the figure into various postures to observe the effects of different values. The result is a stick figure skeleton of pure joints, and is analogous to an armature used in traditional stop motion animation;
- **Adding muscles:** the next step is to "flesh out" the skeleton by adding solid geometrical shapes to the joints, which move with the skeleton and form the basic shape of the articulated figure. These shapes, which correspond to body shape masses in traditional figure drawing, constitute the muscle layer of the layered character model and are based on deformed implicit surfaces such as spheres and superellipses. Their parameters and dimensions, as well as orientations with respect to their associated joints must be specified;
- **Connecting the skin:** the physics-based skin surface is attached to the figure at the skin boundaries and also through simulated connective tissue, which takes the form of "rubber-band" force constraints between points on the skin surface and points on the underlying muscle layer. This is a complex spatial task since it requires associating every point on a global rectangular surface mesh (the skin) with a local point on the surface of a hierarchical shape (the articulated figure). In addition, the stiffness of each rubber-band constraint may be specified globally or individually;
- **Sculpting the fat layer:** the fat layer is represented as a geometric distance separating the skin and muscle layers. It is specified as a simple thickness parameter at each point on the skin, and therefore corresponds to sub-cutaneous fat which moves with the skin. This operation therefore involves associating a single scalar parameter with each point on the skin surface.

All these operations require the specification of 3D data and the understanding of complex 3D information. The user interface should therefore be designed so as to facilitate these operations.

### 5.1. Device Configuration and Interaction Metaphor

High-performance 3D graphics workstations and a variety of multi-dimensional input devices have begun to make highly interactive, direct manipulation environments practical. Finding the right kinds of interaction metaphors by which these devices can control a 3D character model, however, requires experimentation with many of the various possibilities.

It is only recently that techniques for interaction with synthetic worlds have tried to go beyond straightforward interpretation of physical device data[43]. By contrast, recent research in the 3D interaction field has focused on exploring responsive 3D interfaces with better affordances, functional fidelity and mental appeal[44, 45, 46]. This research, while dramatically improving the expressive power of desktop computers to accomplish 3D tasks, has not taken advantage of the latest developments of virtual reality technology to increase the bandwidth and fidelity of man-machine communication. In most cases, the interaction tools reside in 3D space, but are operated with the 2D mouse and presented to users using a conventional perspective view on a workstation monitor.

One of our goals for the animation system was to augment the 3D workstation desktop user-interface to create a restricted, but high-quality form of virtual reality, one that would give a compelling sense of immersion within the confines of the desktop world, but would at the same time be expressive and ergonomic enough for people to use for extended periods of time to do practical work. Recent research on "fishtank VR"[47] and on projection-based systems[48] has started to focus on these techniques.

In the LEMAN system, a six degree-of-freedom head-tracker and CrystalEyes shutter glasses are used to produce stereo images that dynamically follow the user head motion. We have used both Polhemus Fastrak magnetic and Logitech ultrasonic sensors to track the head and 3D mouse motion. As in[49], the left and right viewing frusta are recomputed at each frame from the physical position of the viewer's eyes and the physical position of the display surface. Each of the frusta is a truncated pyramid with apex at the eye position and edges passing through the corners of the rendering viewport. The position of the left and right eyes are computed from offsets from the tracked head-position, while the position of the display surface in tracker coordinates is determined by a calibration step that has to be re-executed only when the screen monitor or the tracker reference frame are moved. Our current calibration procedure simply consists of measuring the position of the four corners of the workstation monitor with the 3D mouse. Thanks to the registration between virtual and physical coordinates, 3D virtual objects can be made to appear at a fixed location in physical space which the user may view from different angles by moving his head. The combination of physically accurate perspective, stereo viewing, and motion parallax provide a compelling illusion of the existence of the simulated 3D objects.

To construct 3D animated characters, the user inter-

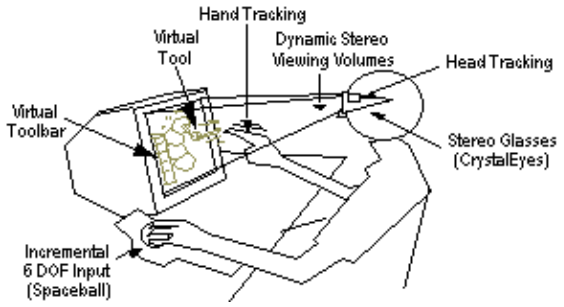


**Figure 4:** *Device configuration and interaction metaphor*

acts with the simulated environment using both hands simultaneously: the left hand, controlling a Spaceball, is used for 3D navigation and object movement, while the right hand, holding a 3D mouse, is used to manipulate the objects appearing in front of the screen through a virtual tool metaphor. In this way, both incremental and absolute interactive input techniques are provided by the system. This combination of input techniques provides several benefits.

Thanks to head-tracking, camera motion can take advantage of simultaneous position and velocity control, and a single control mode has characteristics which are at the same time appropriate for close inspection, exploration, and navigation[50]. In our system, the Spaceball incrementally controls a virtual vehicle, and tracked head and right hand positions are interpreted local to that vehicle. Relying on an incremental device such as the Spaceball for vehicle control reduces the user's fatigue, as opposed to solutions based on absolute devices such as those presented in reference 50, since the left hand can rest on the Spaceball support and only very small finger movements are necessary for motion control.

The different components of an animated character are created, connected and manipulated using virtual tools which are encapsulations of a physical appearance and a behavior[43, 51]. Since tools are manipulated with the right hand using absolute input, the user can have the visual impression of touching the virtual objects that are close to him. These virtual tools are in some ways 3D analogs of the types of interactive tools found in typical 2D drawing programs (e.g. select, resize, create-circle, spray-paint). However, since they are inherently three-dimensional, they are capable of performing more sophisticated 3D spatial tasks, and are often more intuitive for 3D operations than their two-dimensional counterparts since they correspond more closely to a real-world tool. Like a 2D drawing program, the various tools are arranged in a toolbar from which the current tool may be selected using the 3D mouse. Once selected, a copy of the tool is displayed at the current position of the 3D mouse, representing a 3D cursor. A visible "wand" extends a few centimeters out from the cursor and is used for picking objects and specifying positions in space, and a button on the 3D mouse allows picking and dragging operations. The large number of degrees of freedom and direct-manipulation capabilities of these virtual tools allow complex interactive operations, which might otherwise require several 2D widgets and 2D mouse, to be performed naturally with a single virtual tool.

We have experimented with both Polhemus Fastrak magnetic trackers and Logitech ultrasonic trackers to input the head and 3D mouse motion. While the Polhemus is less obtrusive and does not suffer from occlusion problems, its performance degrades considerably when operated close to the workstation monitor due to emitted magnetic fields, which cause jitter, and the presence of metal in the environment, which distorts position values and lowers correspondence between the physical and virtual coordinate systems. The Logitech trackers do not suffer these kinds of problems and work quite well close to the screen, as long as care is taken not to occlude the sensor microphones.

### 5.2. Spatio-temporal Realism

To give users the impression of manipulating real objects, it is important that the lag between their movements and the effects in the synthetic world be as small as possible. This is obtained in LEMAN by decoupling the simulation and tracking activities[22]. At each frame, the following actions are taken:

- the latest values of the 3D mouse and head tracker sensors are read, and virtual camera position and virtual tool position are updated;
- events from the various devices are handled, and the behavior of the active tool is executed;
- if the skin is deforming, simulation steps are executed until the time that remains in the current frame is less or equal to the time spent rendering the previous frame;
- the latest values of the motion tracker sensors are read again, and a new frame is rendered with the latest positions of the virtual camera and tool.

Since the motion tracker is sampled at a much higher rate than the frame rate of the application (60 Hz vs. 10 Hz), and since on our machine (an SGI Onyx RE2) computing the simulation is more expensive in time than rendering the character, reading tracker values twice per frame (once before computing the application behavior and once just before rendering the frame) allows the reduction of the most important lags, i.e. those of the objects directly associated with the physical position of

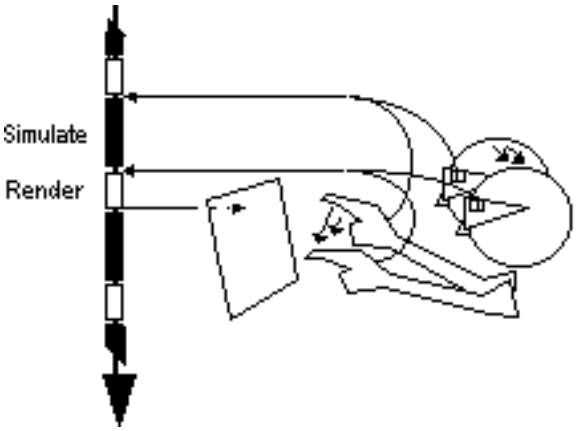


**Figure 5:** *LEMAN Time-line*

**Figure 6:** *Constructing a character*

the user. The perceived hand and head motion lag is determined by the rendering time and does not depend on the simulation time. This lag reduction technique is similar to just-in-time-synchronization[52] with a priority given to the user's position as in DIVER[53].

## 6. Constructing a Character

With LEMAN, the animator can build up a three-dimensional character model from scratch, adjusting parameters and moving it interactively throughout the process to test its appearance and behavior. Once the character is constructed, it can be saved to a file for future use. The main operations involved in character creation have been implemented so as to take advantage of head-tracked stereo-viewing and a two-handed direct-manipulation interface. The hand-eye coordination made possible by the registration between virtual and physical space which allows a variety of complex 3D tasks necessary for constructing 3D animated characters to be performed more easily and more rapidly than is possible using traditional interactive techniques. The following sections describe the main steps involved in the creation of a 3D characters, contrasting our user-interface with traditional desktop approaches such as those used in our previous work[20, 21]. The pictures presented in the following sections were taken by tracking the position of the camera using the Logitech ultrasonic head tracker. MPEG video files presenting live video clips can also be viewed at the following web address: *http://www.crs4.it/~gobbetti/group_homepage/contents/projects/ongoing/leman/index.html.* Figure 6 shows a user in the process of constructing a character.

### 6.1. Skeleton Building and Muscle Creation

In a direct manipulation user environment, it is usually easier to build both muscle and skeleton layers simultaneously, since the geometric link shapes provide something tangible to manipulate. Articulated figures are constructed by selecting one of the shape creation tools (e.g. sphere, superellipse) and selecting a point in space causing a new joint to be created at that location. Dragging the tool, by holding the 3D mouse button down while changing its position and orientation, changes the size and shape of the muscle associated with the joint. For example, drag-translating the tip of the sphere creation tool wand will vary the radius of the sphere, while drag-rotating the superellipse creation tool will vary its curvature parameters. The direct correspondence between the physical location of the hand tracker and the effect on the manipulated objects gives invaluable help when creating the character, since the size of the character's muscles and the position of the joints are controlled directly in the physical space (Figure 7). Traditional user-interfaces such as the one used in reference 21 offer only indirect control over these parameters.

The system maintains a current joint, which is displayed in a highlighted form and can be specified using the selection tool. Newly created joints are connected to the current joint, allowing the skeleton hierarchy to be easily specified. Selecting and dragging an existing joint with a shape creation tool will modify the parameters of the existing joint rather than creating a new one. Model editing and model creation can thus be done with a single tool. By combining head-tracking, vehicle movements controlled by the Spaceball, and model creation using 3D tools using a 3D mouse, an entire skeleton can be created and viewed from all angles in a few minutes without any interaction mode changes in the application.

In order to check that the articulated figure has been properly constructed, the user must be able to move the

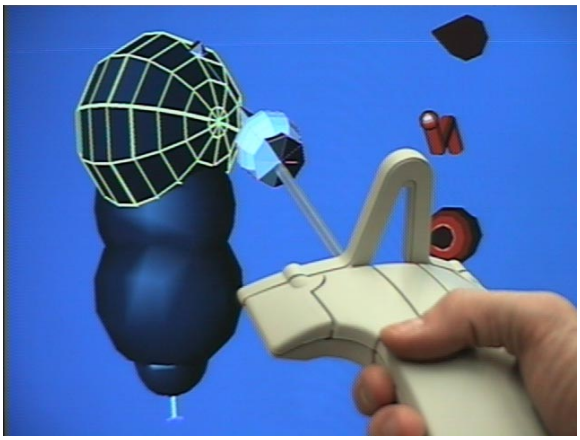

**Figure 7:** *Skeleton creation*

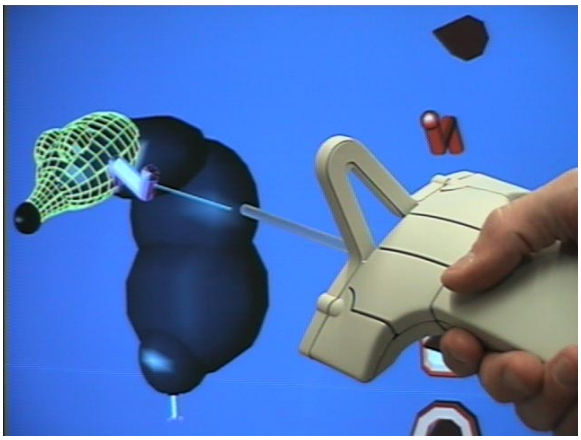

**Figure 8:** *Inverse kinematics manipulation*

skeleton into various postures easily. This is done in the LEMAN system using the inverse kinematics tool, which implements a standard technique used in computer-based character animation that allows the posture of a kinematic chain (e.g. arm, leg, spine) to be specified by moving the end joint of the chain, known as the end-effector. To change the posture of the character's skeleton, the user simply selects the end-effector joint with the inverse kinematics tool and drags it to the desired new position. The entire kinematic chain up to the nearest branching point in the skeleton hierarchy will then follow the motion of the end-effector[1]. This movement can be either a translation, a rotation or both. This differential vector is then multiplied by the inverse Jacobian of the kinematic chain to determine the corresponding differential joint angle values, which are added to the current joint angles[54]. This process is repeated for each event at interactive speeds so that the user has the impression of directly manipulating the skeleton by moving a particular joint. This can easily be replaced with a more sophisticated method when desired. The direct correspondence between physical and virtual locations allows for a direct control in space of the end-effector movement, while the control offered by a non head-tracked application can only be approximate. The behavior of the character can thus be tested more effectively. Figure 8 shows the inverse kinematics positioning of a character's body.

### 6.2. Attaching the Skin

The skin is created and attached to the articulated figure using the skin attachment tool. The skin surface is created and its top and bottom boundaries are fixed to their respective joints by selecting the joints with the tool in sequence from bottom to top. Once both boundaries are specified, the differential equation solver starts

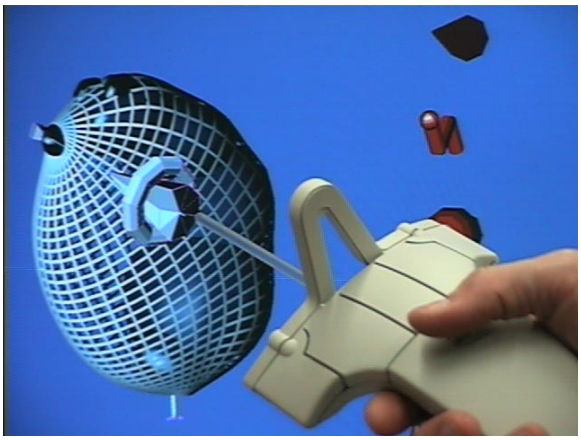

**Figure 9:** *Skin attachment*

up and the skin, initially in a spherical shape, shrinks down around the muscle layer of the character (Figure 9). The connective tissue attachments between the skin and the muscle layers may be created by issuing a global "attach" command which causes all points on the skin surface to be attached to the nearest point on the muscle perpendicular to the skin surface. These attach points may then be adjusted with the skin attachment tool by selecting individual points on the skin surface, dragging the wand to the desired attach point on the muscle surface, and releasing. Selection of regions of the skin surface can be done naturally by touching the virtual skin with the tool. The user can view the effects of the changes from different angles while using the skin attachment tool, simply by moving his head or, if larger motions are needed, by using the Spaceball. Errors caused by editing operations using a single per-

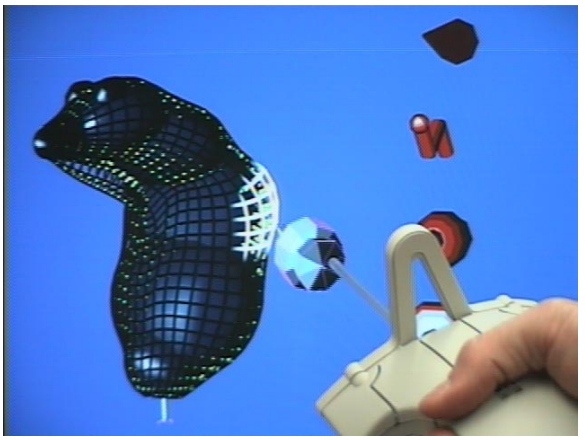

**Figure 10:** *Sculpting the fat layer*

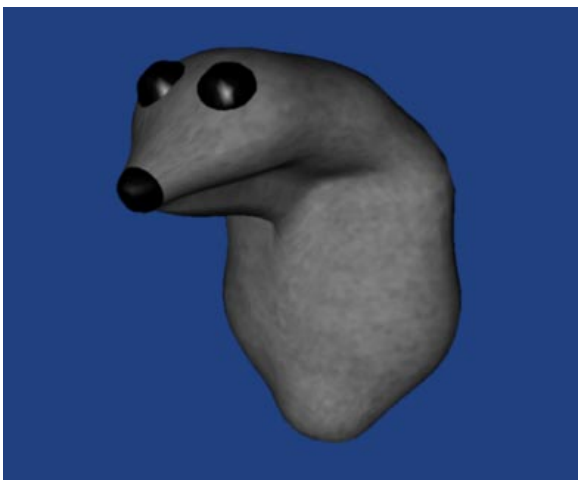

**Figure 11:** *Interactively created character*

spective view are thus reduced, since motion parallax effectively conveys the needed 3D information.

### 6.3. Sculpting the Fat Layer

The final character shape may be sculpted to a fair degree by locally adjusting the thickness of the fat layer. Since the fat thickness is a single parameter associated with each point on the skin surface, it can be controlled very naturally using the "liposuction" tool, which increases or decreases the fat thickness of the skin in a Gaussian distribution around the selected point on the skin surface (Figure 10). The implementation of fat thickness is described in detail in[20]. This tool is analogous in some ways to a spray-can tool in a 2D paint program in that the longer the user holds down the mouse button, the more fat is injected into the skin. The orientation of the tool along its main axis differentiates between injection or removal of fat: a clockwise rotation with respect to the initial orientation injects fat, while a counter-clockwise rotation removes it. Figure 11 shows the final character, which was created in only a few minutes. Texture-maps have been added using a conventional user-interface.

### 6.4. Animation

Once the character has been constructed and all of its physical parameters defined, it may be animated simply by animating the skeleton. The motion of the skeleton provides the input forces which drive the skin surface. There exists a variety of dynamic and kinematic techniques for animating articulated figures. We have chosen a key-frame animation technique in which a series of key postures is specified and a smooth skeletal motion is interpolated between them[1]. The key postures are specified interactively using the inverse kinematics manipulator described in the previous section. Although the interpolated skeletal motion is a purely kinematic one, the resulting dynamic motion of the skin is physically simulated, resulting in a richer form of automatic inbetweening. For example, a perfectly cyclical skeletal motion such as a walk sequence will not necessarily result in a perfectly cyclical skin motion, but rather will vary somewhat from cycle to cycle, depending on the time constants of the elastic surface.

Once a desired posture has been found, the user can store its joint angles as a key posture. A series of key postures can then be used to interpolate a smooth motion, using interpolating splines on the joint angles[55].

To animate the figure, the user positions the skeleton into a sequence of key postures, either without the elastic surface, or with the simulation running at a low surface resolution for interactive speed. A smooth motion can then be created by interpolating the joint angles using an interpolating spline[55]. This motion sequence can be played back in real time, to check the animation, or in (usually non-real) simulated time to calculate an animation sequence at high surface resolution. To give an idea of what the final animation sequence will look like, the simulation can be turned off and then the skeleton motion sequence can be played back at full-speed. To get an accurate impression of the skin dynamics, the simulation can be turned back on while the skeleton motion is played back in simulation time. Key postures can be edited by selecting the particular key and repositioning the skeleton interactively.

### 6.5. Increasing the Surface Resolution

Since our current implementation uses a rectangular mesh to represent the surface, the mesh resolution can

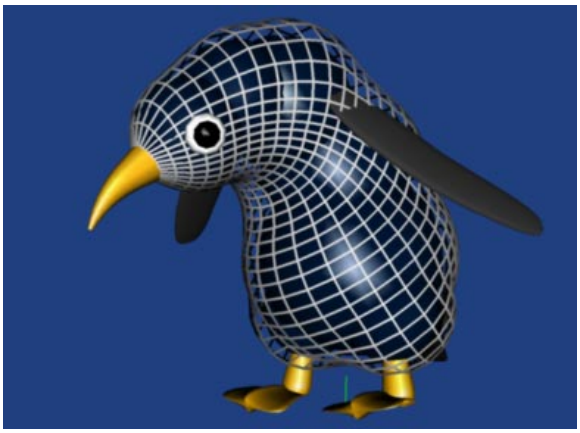

**Figure 12:** *Penguin*

**Figure 13:** *Wombat*

be changed quite easily. All the current values of the mesh (e.g. position, velocity, elasticity, spring constants) are bilinearly interpolated to determine the values of the higher resolution points. Although the character is usually designed and animated at a low surface resolution, once a motion sequence has been specified, the resolution can be increased and the same motion played back in simulation time to calculate a final motion sequence. This motion sequence is stored as a large array of successive elastic surface meshes and can be played back at interactive rates and viewed from different angles to check the final animation. Then the entire sequence can be rendered off-line using a standard rendering package.

## 7. Implementation and Results

LEMAN has been implemented in C on Silicon Graphics workstation. Figures 13, 14, and 15 show some of the characters that have been created so far with the system. Our current implementation makes it possible to work at interactive rates with characters of sufficient complexity. Table 1 summarizes the performance obtained with our system on a Silicon graphics Onyx RE2. Three different explicit solvers were used to integrate the differential equations of motion: a fixed timestep first-order Euler solver, the adaptive second-order Verlet solver described in section 4.2, and an adaptive second-order Runge-Kutta solver[40]. For each of the solvers, performance has been measured with the high-resolution versions of the three characters presented in figures 13 to 15. The Verlet solver performs consistently better, enabling high-resolution simulations to run at 1/8 to 1/28 of real-time. Simpler versions of the same characters with skin meshes of 256 mass points are interactively animated at 1/2 to 1/7 of real-time, and may thus be effectively used for behavior testing and during animation design.

**Figure 14:** *Torso*

For moderately complex characters such as the torso and the wombat, the performance bottleneck in our current implementation is the computation of the geometric constraints, which involves performing inside-outside tests between each mass point on the skin and each implicit surface defining the skeleton, a relatively costly operation in the case of arbitrarily oriented superquadrics. At the moment, only per-primitive bounding box tests are performed to reduce the number of muscles examined for each mass point. Localizing the search for constraint forces to only neighboring surfaces for a given skin point would be an effective and straightforward optimization.

## 8. Conclusions and Future Work

We believe that the elastic surface layer model is a promising approach to constructing animated three-dimensional characters. By modeling the skin as a sep-

| | PENGUIN | WOMBAT | TORSO |
|---|---|---|---|
| Character structure | | | |
| Skin mesh | 1024 mass points | 1024 mass points | 1024 mass points |
| Muscle Layer | 7 spheres | 2 spheres<br>5 superellipsoids<br>1 cylinder | 26 spheres<br>12 superellipsoids<br>2 cylinders |
| Euler 1 solver | | | |
| Timestep interval (ms) | [2.5–2.5] | [1.6–1.6] | [2.2–2.2] |
| Simulation slowdown | 20.7× | 80× | 58.1× |
| % Force evaluation | 58.8 | 16.3 | 16.1 |
| % Constraints | 16.3 | 75.5 | 76.8 |
| % Other | 24.9 | 8.2 | 7.1 |
| Leapfrog Verlet 2(1) solver | | | |
| Timestep interval (ms) | [3.5–14] | [3.5–14] | [4–16] |
| Simulation slowdown | 8.1× | 28.5× | 24.5× |
| % Force evaluation | 58.9 | 15.5 | 16.0 |
| % Constraints | 17.4 | 77.5 | 77.2 |
| % Other | 23.7 | 7.0 | 6.8 |
| Runge-Kutta 2(1) solver | | | |
| Timestep interval (ms) | [6.5–26] | [5.5–22] | [6–24] |
| Simulation slowdown | 15.1× | 29.9× | 28.0× |
| % Force evaluation | 81.0 | 47.1 | 47.8 |
| % Constraints | 4.7 | 45.4 | 45.8 |
| % Other | 14.3 | 7.5 | 6.4 |

**Table 1:** *Simulation results*

arate elastic surface, which is free to slide along its underlying muscle layers while being held to it by attracting connective tissue, we have been able to simulate a rich variety of realistic-looking animation effects, with a conceptually simple model. Since the physical simulation is of an elastic surface only, while the underlying layers are geometrical models, we have been able to execute the model at interactive rates, allowing a three-dimensional, direct manipulation environment for layered character construction and animation. One of the main limitations of our current implementation resides in the topological restrictions of the skin mesh, which make it difficult to create surfaces with thin appendages (like arms and legs). Addition of self-collision detection to the skin would allow greater deformations at the joints and more pronounced wrinkling, but at considerable CPU time cost. Adding dynamic properties to other layers such as the fat and muscle layers would also enhance realism.

The techniques presented in this paper make it possible to efficiently simulate the behavior of a moderately complex layered elastic character on current graphics workstations. However, in order for them to be used effectively in an interactive context, one major problem remains to be solved: establishing the relation between real-time and simulation time. During interactive manipulation of the skeleton, the secondary animation of the skin has to follow the skeleton's motion, at a constant simulation speed that should be determined by the user. Our current solution is to try to maintain a constant simulation time over real time ratio by permitting only small stepsize adaptations, so that the number of integration steps per frame does not vary too much. This solution forces users to select conservative stepsize intervals in order to avoid stability problems. We are currently exploring the possibility of letting the user specify the desired relation between real time and simulation time, and to adaptively resample the simulated model at each frame, selecting a number of simulated mass-points as a function of the load of the system in order to meet the timing constraints. The animator thus has the option of deciding whether to give preference to high spatial resolution dynamic effects to be viewed in slow motion, or to geometrically simplified characters animated at real-time speeds. Only secondary skin animation is affected by this choice, since the skeleton is always manipulated in real-time and the constraints that ensure that the skin is wrapped around it are always met.

We also believe that highly interactive user interfaces based on 3D direct manipulation can be successfully applied to the task of constructing animated characters. By viewing a stereo display that dynamically follows the user's head motion and by interacting with the simulated character model using both hands simultaneously, the animator can manipulate the objects appearing in front of the screen through a virtual tool metaphor. This makes it possible for construction and animation of 3D characters to be performed more easily and more rapidly than is possible using traditional interactive techniques.

Our future work on the system's user interface will concentrate on developing more tools for character construction and character animation, the goal being the creation of a system where all interaction is done in three dimensions. We would also like to improve registration between virtual and physical space by developing visual calibration procedures. Such a system, we believe, would provide a prototype user-interface metaphor useful in a variety of highly-interactive desktop VR applications in areas such as surgical simulation, surface modeling and scientific visualization as well as animation.

**Acknowledgements**

This work was partially supported by the United States National Science Foundation Interactive Systems grant number 1R1-95-03-093 and by the financial contribution of Sardinian Regional Authorities.

**References**

1. M. Girard. Interactive Design of 3D Computer-Animated Legged Animal Motion, *IEEE Computer Graphics and Applications*, **7**(6), pp. 39–51 (1987).
2. J. Lasseter. Principles of Traditional Animation Applied to 3D Computer Animation, *Proc. SIGGRAPH '87, Computer Graphics*, **21**(4), pp. 35–44 (1987).
3. W. W. Armstrong, and M. Green. The Dynamics of Articulated Rigid Bodies for Purposes of Animation, *Proc. Graphics Interface '85*, pp. 407–416 (1985).
4. J. Wilhelms, and B. A. Barsky. Using Dynamics Analysis for the Animation of Articulated Bodies Such as Human and Robots, *Proc. Graphics Interface '85*, pp. 97–104 (1985).
5. P. M. Isaacs and M. F. Cohen. Controlling Dynamic Simulation with Kinematic Constraints, Behavior Functions and Inverse Dynamics, *Proc. SIGGRAPH' 87, Computer Graphics*, **21**(4), pp. 215–224 (1987).
6. J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Birein. Animating Human Athletics. *Proc. SIGGRAPH*: pp. 71–78 (1995).
7. B. M. Blumberg and T. A. Galyean. Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments. *Proc. SIGGRAPH*: pp. 47–54 (1995).
8. A. Bruderlin and L. Williams. Motion Signal Processing. *Proc. SIGGRAPH*: pp. 97–104 (1995).
9. A. Witkin and Z. Popovi'c. Motion Warping. *Proc. SIGGRAPH*: pp. 105–108 (1995).
10. A. Witkin and M. Kass. Spacetime Constraints, *Proc. SIGGRAPH '88, Computer Graphics*, **22**(4), pp. 159–168 (1988).
11. T. W. Sederberg and S. R. Parry. Free-Form Deformations of Solid Geometric Models, *Proc. SIGGRAPH'86 Computer Graphics*, **20**(4), pp. 151–160 (1986).
12. S. Coquillart and P. Jancene. Animated Free-Form Deformations: An Interactive Animation Technique. *Proc. SIGGRAPH*: pp. 23–26 (1991).
13. J. Bloomenthal and B. Wyvill. Interactive Techniques for Implicit Modeling, *Proc. SIGGRAPH Symposium on Interactive 3D Graphics, Computer Graphics*, **24**(2), pp. 109–116 (1990).
14. M. Desbrun, N. Tsingos, and M. P. Gascuel. Adaptive Sampling of Implicit Surfaces for Interactive Modeling and Animation. *Proc. Implicit Surfaces* (1995).
15. D. R. Forsey and R. H. Bartels. Hierarchical B-Spline Refinement, *Proc. SIGGRAPH '88, Computer Graphics*, **22**(4), pp. 205–212 (1988).
16. J. Shen and D. Thalmann. Interactive Shape Design Using Metaballs and Splines. *Proc. Implicit Surfaces '95* (1995).
17. D. Terzopoulos, J. C. Platt, A. H. Barr, and K. Fleischer. Elastically Deformable Models, *Proc. SIGGRAPH '87, Computer Graphics*, **21**(4), pp. 205–214 (1987).
18. J. C. Platt and A. H. Barr. Constraint Method for Flexible Models, *Proc. SIGGRAPH '88, Computer Graphics*, **22**(4), pp. 279–288 (1988).
19. A. Witkin and W. Welch. Fast Animation and Control of Non-Rigid Structures. *Proc. SIGGRAPH '90, Computer Graphics* **24**(4), pp. 243–252 (1990).
20. R. Turner and D. Thalmann. The Elastic Surface Layer Model for Animated Character Construction. *Proc. Computer Graphics International* (1993).

21. R. Turner. LEMAN: A System for Constructing and Animating Layered Elastic Characters. *Proc. Computer Graphics International* (1995).

22. R. Turner, E. Gobbetti, and I. Soboroff. Head Tracked Stereo Viewing with Two-Handed Interaction for Animated Character Construction. *Proc. EUROGRAPHICS* (1996).

23. S. Culhane. Animation From Script To Screen, St. Martin's Press, New York (1988).

24. P. Bergeron and P. Lachapelle. Controlling facial expressions and Body Movements in the Computer-Generated Animated Short "Tony De Peltrie". *SIGGRAPH '85 Advanced Computer Animation Seminar Notes* (1985).

25. D. R. Forsey. A Surface Model for Skeleton-Based Character Animation, *Proc. Second Eurographics Workshop on Animation and Simulation*, INRIA/IRISA, Rennes: pp. 55–74 (1991).

26. G. Walters. The Story of Waldo C. Graphic, 3D Character Animation By Computer, *SIGGRAPH '89 Tutorial Notes* (1989).

27. J. Chadwick, D. R. Haumann, and R. E. Parent. Layered Construction for Deformable Animated Characters, Proc. *SIGGRAPH '89, Computer Graphics*, **23**(3), pp. 234–243 (1989).

28. M. P. Gascuel, A. Verroust, and C. Puech. A Modelling System for Complex Deformable Bodies Suited to Animation and Collision Processing, *The Journal of Visualization and Computer Animation*, **2**(3), pp. 82–91 (1991).

29. J. P. Gourret, N. Magnenat-Thalmann, and D. Thalmann. (1989) Simulation of Object and Human Skin Deformations in a Grasping Task, *Proc. SIGGRAPH '89, Computer Graphics*, **23**(3), pp. 21–30.

30. D. T. Chen and D. Zeltzer. Pump It Up: Computer Animation of a Biomechanically Based Model of Muscle Using the Finite Element Method, *Proc. SIGGRAPH '92, Computer Graphics*, **26**(2), pp. 89–98 (1992).

31. D. Terzopoulos and K. Waters. Techniques for Realistic Facial Modeling and Animation, in: Magnenat Thalmann N, Thalmann D, *Computer Animation '91*, Springer-Verlag, Tokyo: pp. 59–74 (1991).

32. Y. Lee, D. Terzopoulos, and K. Waters. Realistic Modeling for Facial Animation. *Proc. SIGGRAPH '95*: pp. 55–62 (1995).

33. Alias/Wavefront Maya Backgrounder: The First Step Towards A Vision. (White Paper) http://www.aw.sgi.com / pagees / home / pages / about_us / pages / white_papers / pages / 95_08_06 Maya_Backgrounder/ (1995).

34. A. H. Barr. Superquadrics and Angle Preserving Transformations, *IEEE Computer Graphics and Applications*, **1**(1), pp. 11–23 (1981).

35. A. H. Barr. Global and Local Deformations of Solid Primitives, *Proc. SIGGRAPH '84, Computer Graphics*, **18**(3), (1984).

36. I. M. Gelfand and S. V. Fomin. Calculus of Variations, Prentice-Hall, Englewood Cliffs, New Jersey (1963).

37. L. Verlet. Computer Experiments on Classical Fluids. I. Termodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.* **159**, 98–103 (1967).

38. M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford Science Publications (1994).

39. A. M. Mathiowetz. *Dynamic and Stochastic Protein Simulations: From Peptides to Viruses*. PhD Dissertation. Caltech (1993).

40. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numeical Recipes in C*. Cambridge University Press (1992).

41. D. Baraff. Analytical Methods for the Dynamic Simulation of Non-Penetrating Rigid Bodies. *Proc. SIGGRAPH*: 223–232 (1989).

42. A. Witkin, K. Fleischer, and A. H. Barr. Energy Constraints on Parameterized Models, *Proc. SIGGRAPH '87, Computer Graphics*, **21**(4), pp. 225–232 (1987).

43. D. B. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, and A. Van Dam, Three-Dimensional Widgets. *ACM SIGGRAPH Symposium on Interactive 3D Graphics*: pp. 183–188 (1992).

44. K. P. Herndon, A. van Dam, and M. Gleicher. Workshop Report: The Challenges of 3D Interaction. *SIGCHI Bulletin*, October (1994).

45. E. Gobbetti and J. F. Balaguer. An Integrated Environment to Visually Construct 3D Animation. *Proceedings ACM SIGGRAPH* (1995).

46. K. P. Herndon, R. C. Zeleznik, D. C. Robbins, D. B. Conner, S. S. Snibbe, and A. Van Dam. Interactive Shadows. *Proceedings UIST*: 1-6 (1992).

47. C. Ware, K. Arthur, and K. S. Booth. Fishtank Virtual Reality. *Proc. INTERCHI*: pp. 37–42 (1993).

48. W. Krueger and B. Froehlich. The Responsive Workbench. *IEEE Computer Graphics and Applications* **14**(3), pp. 12–15 (1994).

49. M. Deering. High Resolution Virtual Reality. *Proceedings ACM SIGGRAPH* pp. 195–202 (1992).

50. C. Ware and S. Osborne. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. *Proc. ACM Workshop on Interactive 3D Graphics*, pp. 175–183 (1990).

51. E. Gobbetti and J. F. Balaguer. VB2: A Framework for Interaction in Synthetic Worlds. *Proceedings ACM UIST* (1993).

52. M. Wloka. Lag in Multiprocessor Virtual Reality. PRESENCE: Teleoperators and Virtual Environments **4**(1), pp. 50–63 (1995).

53. R. Pausch, M. Conway, R. DeLine, R. Gossweiler, and S. Miale. Alice and DIVER: A Software Architecture for Building Virtual Environments. NSF Invitational Workshop, University of North Carolina at Chapel Hill, March 23-24: pp. 154–177 (1993).

54. H. Klein. Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators, *IEEE Transaction on Systems, Man and Cybernetics*, **132**(3) (1983).

55. D. H. Kochanek and R. H. Bartels. Interpolating Splines with Local Tension, Continuity, and Bias Control, *Proc. SIGGRAPH '84, Computer Graphics*, **18**, pp. 33–41 (1984).