Animation of deformable models

Uğur Güdükbay and Bülent Özgüç

Although kinematic modelling methods are adequate for describing the shapes of static objects, they are insufficient when it comes to producing realistic animation. Physically based modelling remedies this problem by including forces, masses, strain energies and other physical quantities. The paper describes a system for the animation of deformable models. The system uses physically based modelling methods and approaches from elasticity theory for animating the models. Two different formulations, namely the *primal formulation* and the *hybrid formulation*, are implemented so that the user can select the one most suitable for an animation depending on the rigidity of the models. Collision of the models with impenetrable obstacles and constraining of the model points to fixed positions in space are implemented for use in the animations.

Keywords: modelling, animation, simulation

An animation system which is implemented for the animation of nonrigid (deformable) models is discussed in this paper. The system is built on top of a modelling system for representing 3D freeform objects[1]. The static models obtained can be animated using the techniques discussed in this paper. The system is implemented using C language on a Unix* workstation environment (it runs on Sun 3 and Sparc workstations).

Currently, most of the methods used for modelling are *kinematic*. This becomes a major drawback when we want to create realistic animation since these methods are passive; they do not interact with each other or with external forces. The behaviour and forms of many objects are determined by the objects' gross physical properties. To build and animate active models, physically based techniques should be used. These techniques facilitate the creation of models capable of automatically synthetizing complex shapes and realistic motions that are attainable only by skilled animators. *Physically based modelling* achieves this by adding physical properties to the models. Such properties may be forces, torques, velocities, accelerations, kinetic and potential energies, heat etc. Physical simulation is then used to produce animation on the basis of these properties. Researchers continue to present faster and simpler formulations to build and control the motions of models using physically based approaches†.

Another aspect in realistic animation is that of modelling the behaviour of *deformable objects*. To simulate the behaviour of deformable objects, we should approximate a continuous model by using discretization methods, such as finite difference methods and finite element methods. For finite difference discretization, a deformable object could be approximated by using a grid of control points where the points are allowed to move in relation to one another. The manner in which the points are allowed to move determines the properties of the deformable object. Various researchers[3–5] have presented discrete models to define and enforce constraints for modelling and animating deformable objects.

The plan of this paper is to first present a short description of the formulations which we have used to animate elastically deformable models. We will then explain the implementation details of these formulations in the context of our system, and the algorithmic solution of the problems, such as the collision of flexible models with impenetrable obstacles. Then, some simulation results representing the features of the system are given and a quantitative comparison of the formulations is presented. Finally, conclusions are given.

NONRIGID MODELS

To animate nonrigid objects in a simulated physical environment, we should use the methods of elasticity theory. Elasticity theory provides methods to construct the differential equations that model the behaviour of nonrigid curves, surfaces, and solids as a function of time. Real materials exhibit both elastic and inelastic behaviour. Some materials undergo perfectly elastic deformations so that, when the forces acting on the materials are removed, objects restore themselves to their natural shapes completely. However, there are other materials, such as cloth, paper etc., which restore themselves to their initial shapes slowly (or partially) upon removal of the forces that cause deformations.

To model elastic materials, physical properties such as tension and rigidity should be simulated. In this way,

*Unix is a registered trademark of AT&T Laboratories, USA.

Department of Computer Engineering and Information Science, Bilkent University, Bilkent, 06533 Ankara, Turkey
Paper received: 7 October 1993. Revised: 28 May 1994

†A complete bibliography of computer animation can be found in Reference 2.

0010-4485/94/12/0868–08

static shapes of a wide range of deformable objects, including string, rubber, cloth, paper, and flexible metals, can be modelled. Dynamics of these materials can be simulated by including physical properties, such as mass and damping. The simulation entails the numerical solution of the partial differential equations that govern the evolving shape of the deformable object and its motion through space.

Formulation of deformable models

To simulate the dynamics of elastically deformable models, we use two different formulations, namely the *primal formulation*[6] and the *hybrid formulation*[7].

Primal formulation

In the primal formulation, a deformable model is formulated by using the material coordinates of points in the body (denoted by Ω). For a solid body, $\mathbf{u}=(u_1, u_2, u_3)$, for a surface, $\mathbf{u}=(u_1, u_2)$, and, for a curve, $\mathbf{u}=(u_1)$ denotes the material coordinates. The Euclidean 3-space positions of points in the body are given by the time-varying vector-valued function $\mathbf{x}(\mathbf{u}, t)=[x_1(\mathbf{u}, t), x_2(\mathbf{u}, t), x_3(\mathbf{u}, t)]$. The body in its natural resting state is given by $\mathbf{x}^0(\mathbf{u})=[x_1^0(\mathbf{u}), x_2^0(\mathbf{u}), x_3^0(\mathbf{u})]$ (see *Figure 1*). The equations of motion for a deformable model can be written in Lagrangian form (which should hold for all $\mathbf{u}$ in the material domain Ω) as

$$\frac{\partial}{\partial t}\left(\mu\frac{\partial \mathbf{x}}{\partial t}\right)+\gamma\frac{\partial \mathbf{x}}{\partial t}+\frac{\delta\varepsilon(\mathbf{x})}{\delta \mathbf{x}}=\mathbf{f}(\mathbf{x}, t) \tag{1}$$

where $\mu(\mathbf{u})$ is the mass density of the body at $\mathbf{u}$, $\gamma(\mathbf{u})$ is the damping density of the body at $\mathbf{u}$, $\mathbf{f}(\mathbf{x}, t)$ is the net externally applied force, and $\varepsilon(\mathbf{x})$ is the energy functional which measures the net instantaneous potential energy of the elastic deformation of the body.

The shape of a body is determined by the Euclidean distances between nearby points. As the body deforms, these distances change. Let $\mathbf{u}$ and $\mathbf{u}+d\mathbf{u}$ denote the material coordinates of two nearby points in the body.

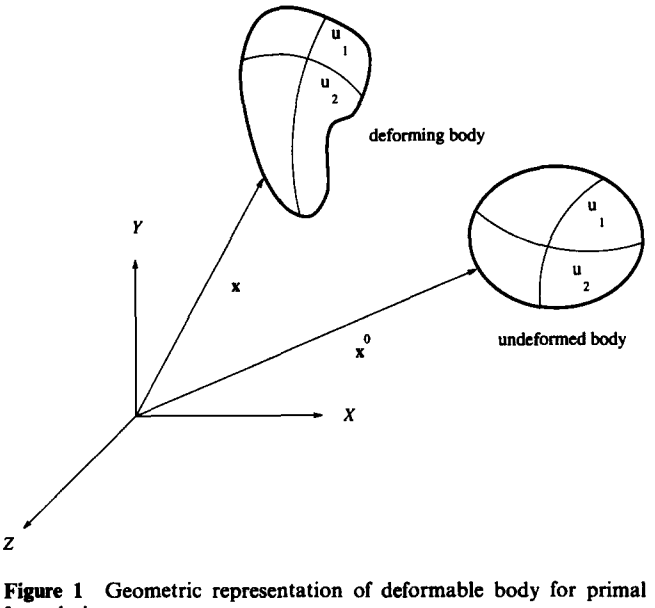

Figure 1 Geometric representation of deformable body for primal formulation

The distance between these points in the deformed body in Euclidean 3-space is given by

$$dl=\sum_{i,j} G_{ij}\, du_i\, du_j \tag{2}$$

where the symmetric matrix

$$G_{ij}(\mathbf{x}(\mathbf{u}))=\frac{\partial \mathbf{x}}{\partial u_i}\cdot\frac{\partial \mathbf{x}}{\partial u_j} \tag{3}$$

is the metric tensor, which is a measure of deformations (the dot indicates the scalar product of two vectors).

Two 3D solids have the same shape (differ only by a rigid body motion) if their 3×3 metric tensors are identical forms of $\mathbf{u}=[u_1, u_2, u_3]$ for all $\mathbf{u}$ in the material domain Ω. Two surfaces have the same shape if their metric tensors $\mathbf{G}$ as well as their curvature tensors $\mathbf{B}$ are identical forms of $\mathbf{u}=[u_1, u_2]$ for all $\mathbf{u}$ in the material domain Ω. The components of the curvature tensor are

$$B_{ij}(\mathbf{x}(\mathbf{u}))=\mathbf{n}\cdot\frac{\partial^2 \mathbf{x}}{\partial u_i\, \partial u_j} \tag{4}$$

where $\mathbf{n}=[n_1, n_2, n_3]$ is the unit surface normal. Two space curves have the same shape if their arc length $s(\mathbf{x}(u))$, curvature $\kappa(\mathbf{x}(u))$, and torsion $\tau(\mathbf{x}(u))$ are identical forms of $\mathbf{u}=[u_1]$. See Reference 8 for a detailed discussion of these formulations.

Using the above differential quantities, potential energies of deformation for use in Lagrange equations can be defined as the norm of the difference between the fundamental forms of the deformed body and those of the undeformed body. This norm measures the amount of deformation away from the natural shape so that the potential energy is zero when the body is in its natural shape and it increases as the model is increasingly deformed away from its natural shape.

If the fundamental forms associated with the natural shape are denoted by the superscript 0, then the strain energy for a curve can be defined as

$$\varepsilon(\mathbf{x})=\int_\Omega w^1(s-s^0)^2+w^2(\kappa-\kappa^0)^2+w^3(\tau-\tau^0)^2\, du \tag{5}$$

where w^1, w^2 and w^3 are the coefficients for the curve that show the amounts of resistance to stretching, bending, and twisting, respectively. The strain energy for a surface can be defined in a similar way:

$$\varepsilon(\mathbf{x})=\int_\Omega \|\mathbf{G}-\mathbf{G}^0\|_{w^1}^2+\|\mathbf{B}-\mathbf{B}^0\|_{w^2}^2\, du_1\, du_2 \tag{6}$$

where the weighted matrix norms $\|\cdot\|_{w^1}$ and $\|\cdot\|_{w^2}$ involve the weighting functions $w_{ij}^1(u_1, u_2)$ and $w_{ij}^2(u_1, u_2)$. Analogously, a strain energy for a deformable solid is

$$\varepsilon(\mathbf{x})=\int_\Omega \|\mathbf{G}-\mathbf{G}^0\|_{w^1}^2\, du_1 du_1 du_3 \tag{7}$$

where the weighted matrix norm $\|\cdot\|_{w^1}$ involves the weighting functions $w_{ij}^1(u_1, u_2, u_3)$.

These energies denote the amount of energy required to restore the deformed objects to their natural shapes.

The net external force in Lagrange's equations is the sum of various types of external force, such as gravitational force and constraint forces.

The weighting functions in the above energies ($w^1_{ij}(u_1, u_2)$ and $w^2_{ij}(u_1, u_2)$ for a deformable surface) determine the properties of the simulated deformable material. The weighting function $w^1_{ij}(u_1, u_2)$ determines surface tensions and shear strengths which minimize the deviation of the surface's actual metric coefficients G_{ij} from its natural coefficients G^0_{ij}. As w^1_{ij} is increased, the material becomes more resistant to length deformation, with w^1_{11} and w^1_{22} determining this resistance along u_1 and u_2, and $w^1_{12} = w^1_{21}$ determining the resistance to shear deformation. The functions $w^2_{ij}(u_1, u_2)$ control surface rigidities which act to minimize the deviation of the surface's actual curvature coefficients B_{ij} from its natural coefficients B^0_{ij}. As w^2_{ij} is increased, the material becomes more resistant to bending deformation, with w^2_{11} and w^2_{22} determining this resistance along u_1 and u_2, and $w^2_{12} = w^2_{21}$ determining the resistance to twist deformation. To simulate a stretch rubber sheet, for example, we make w^1_{ij} relatively small and set $w^2_{ij} = 0$. To simulate relatively stretch resistant cloth, we increase the value of w^1_{ij}. To simulate paper, we make w^1_{ij} relatively large and we introduce a modest value for w^2_{ij}. Springy metal can be simulated[6] by increasing the value of w^2_{ij}.

To create animation with deformable models, the differential equations of motion should be discretized by applying finite difference approximation methods and solving the system of linked ordinary differential equations of motion obtained in this way.

Hybrid formulation

In this formulation, a deformable body is represented as the sum of a reference component $\mathbf{r}(\mathbf{u}, t)$ and a deformation component $\mathbf{e}(\mathbf{u}, t)$ (see *Figure 2*). The positions of mass elements in the body relative to a body frame ϕ (whose origin coincides with the body's centre of mass and which should be evolved over time according to rigid body dynamics to have a rigid body motion besides its elastic motion) is given by

$$\mathbf{q}(\mathbf{u}, t) = \mathbf{r}(\mathbf{u}, t) + \mathbf{e}(\mathbf{u}, t) \tag{8}$$

Deformations are measured with respect to the reference shape $\mathbf{r}$. Elastic deformations are represented by an energy $\varepsilon(\mathbf{e})$ which depends on the position of the body frame ϕ.

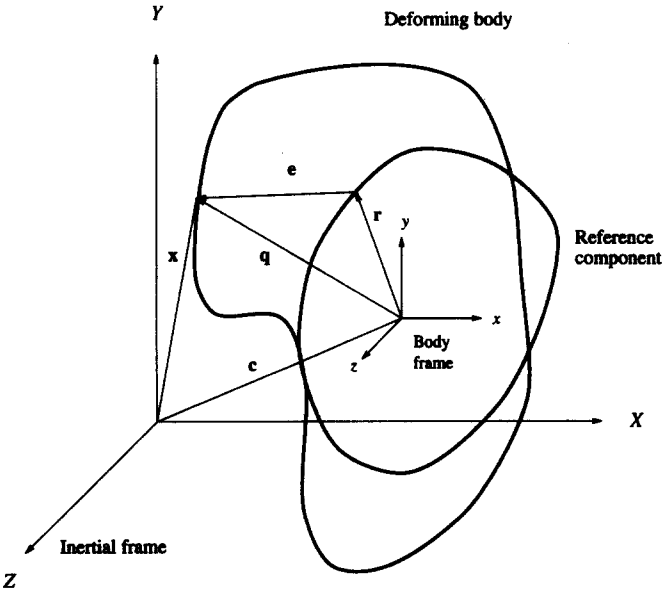

Figure 2 Geometric representation of deformable body for hybrid formulation

IMPLEMENTATION OF PRIMAL FORMULATION

To simulate the dynamics of a deformable surface, we must discretize the following expression for the elastic force, which is an approximation of the derivative of the expression for potential energy:

$$\varepsilon(x) = \sum_{i,j=1}^{2} -\frac{\partial}{\partial u_i}\left(\alpha_{ij}\frac{\partial \mathbf{x}}{\partial u_j}\right) + \frac{\partial^2}{\partial u_i u_j}\left(\beta_{ij}\frac{\partial^2 \mathbf{x}}{\partial u_i u_j}\right) \tag{9}$$

where the functions $\alpha_{ij}(u, \mathbf{x})$ and $\beta_{ij}(u, \mathbf{x})$ determine the elastic properties of the material. The expressions for $\alpha_{ij}(u, \mathbf{x})$ and $\beta_{ij}(u, \mathbf{x})$ are as follows:

$$\alpha_{ij}(u, \mathbf{x}) = w^1_{ij}(u)(G_{ij} - G^0_{ij}) \tag{10}$$

$$\beta_{ij}(u, \mathbf{x}) = w^2_{ij}(u)(B_{ij} - B^0_{ij}) \tag{11}$$

The discretization of the expression for elastic force is achieved by applying the finite difference approximation method.

Since the body coordinates of the models are in the unit square domain, $\Omega = 0 \leqslant u_1, u_2 \leqslant 1$, we discretize this domain as a regular $M \times N$ discrete grid of nodes. Here, the internode spacings are $h_1 = 1/M$ and $h_2 = 1/N$ in the u_1 and u_2 directions, respectively. The nodes on the discrete model are indexed by integers $[m, n]$, where $1 \leqslant m \leqslant M$ and $1 \leqslant n \leqslant N$. Therefore, if $\mathbf{x}$ (which is a continuous vector function $\mathbf{x}(u, t)$ represents the 3D coordinates of the positions of points, we discretize it by arrays of continuous time vector-valued nodal variables $\mathbf{x}_t[m, n] = \mathbf{x}(mh_1, nh_2, t)$.

Since the elastic force requires the approximations to the first and second derivatives of the nodal variables, we should first define them for the vector-valued position function $\mathbf{x}$.

The forward difference operators are

$$D_1^+\mathbf{x}[m, n] = (\mathbf{x}[m+1, n] - \mathbf{x}[m, n])/h_1 \tag{12}$$

$$D_2^+\mathbf{x}[m, n] = (\mathbf{x}[m, n+1] - \mathbf{x}[m, n])/h_2 \tag{13}$$

and the backward difference operators are

$$D_1^-\mathbf{x}[m, n] = (\mathbf{x}[m, n] - \mathbf{x}[m-1, n])/h_1 \tag{14}$$

$$D_2^-\mathbf{x}[m, n] = (\mathbf{x}[m, n] - \mathbf{x}[m, n-1])/h_2 \tag{15}$$

Using Equations 13 and 15, we can define the forward and backward cross difference operators as

$$\begin{aligned} D_{12}^+\mathbf{x}[m, n] &= D_{21}^+\mathbf{x}[m, n] = D_1^+ D_2^+\mathbf{x}[m, n] \\ &= (\mathbf{x}[m+1, n+1] - \mathbf{x}[m+1, n] \\ &\quad - \mathbf{x}[m, n+1] + \mathbf{x}[m, n])/h_1 h_2 \end{aligned} \tag{16}$$

$$D_{12}^{-}\mathbf{x}[m,n]=D_{21}^{-}\mathbf{x}[m,n]=D_1^{-}D_2^{-}\mathbf{x}[m,n]$$
$$=(\mathbf{x}[m,n]-\mathbf{x}[m,n-1]-\mathbf{x}[m-1,n]$$
$$+\mathbf{x}[m-1,n-1])/h_1h_2 \quad (17)$$

and the central difference operators as

$$D_{11}\mathbf{x}[m,n]=D_1^{-}D_1^{+}\mathbf{x}[m,n]$$
$$=\mathbf{x}[m+1,n]-2\mathbf{x}[m,n]+\mathbf{x}[m-1,n])/h_1^2 \quad (18)$$

$$D_{22}\mathbf{x}[m,n]=D_2^{-}D_2^{+}\mathbf{x}[m,n]$$
$$=(\mathbf{x}[m,n+1]-2\mathbf{x}[m,n]+\mathbf{x}[m,n-1])/h_2^2 \quad (19)$$

Now, using the grid functions $\mathbf{x}[m,n]$, $w_{ij}^1[m,n]$, $w_{ij}^2[m,n]$ to represent their continuous counterparts, we can discretize the expressions in Equations 10 and 11 as follows:

$$a_{ij}[m,n]=w_{ij}^1[m,n](D_i^{+}\mathbf{x}[m,n]\cdot D_j^{+}\mathbf{x}[m,n]-G_{ij}^0[m,n]) \quad (20)$$

$$b_{ij}[m,n]=w_{ij}^2[m,n](\mathbf{n}[m,n]\cdot D_{ij}^{+}\mathbf{x}[m,n]-B_{ij}^0[m,n]) \quad (21)$$

where the + superscript indicates that the forward cross difference operator is used when $i\neq j$, and

$$\mathbf{n}[m,n]=\frac{D_1^{+}\mathbf{x}[m,n]\times D_2^{+}\mathbf{x}[m,n]}{|D_1^{+}\mathbf{x}[m,n]\times D_2^{+}\mathbf{x}[m,n]|} \quad (22)$$

is the surface normal grid function. The elastic force in Equation 9 can be approximated as

$$\varepsilon[m,n]=\sum_{i,j=1}^{2}-D_i^{-}(a_{ij}D_j^{+}\mathbf{x}[m,n])+D_{ij}^{-}(b_{ij}D_{ij}^{+}\mathbf{x}[m,n]) \quad (23)$$

To introduce free boundary conditions on the free edges of a surface where the inner difference operators in Equation 23 attempt to access nodal variables outside the discrete domain, we set the value of the inner difference operators to zero.

Expressing the grid functions $\mathbf{x}[m,n]$ and $\varepsilon[m,n]$ as $\underline{\mathbf{x}}$ and $\underline{\varepsilon}$ in grid vector notation, where these denote the 3D positions of model points and elastic force for each model point stored in an $M\times N$ vector for an $M\times N$ discrete grid of a deformable model, the elastic force can be written in vector form as

$$\underline{\varepsilon}=\mathbf{K}(\underline{\mathbf{x}})\cdot\underline{\mathbf{x}} \quad (24)$$

where $\mathbf{K}$ is an $MN\times MN$ matrix. $\mathbf{K}$ is a sparse and banded matrix. This becomes a major advantage when we solve the simultaneous system of 2nd-order ordinary differential equations. The band structure of the matrix $\mathbf{K}$ is shown in *Figure 3*.

Then, we need to calculate the total external force for each point of the model which is discretized using body coordinates. In order to achieve this, we must add the forces affecting a point: gravitational, viscous, collision and constraint forces. The constraint forces are taken into account in the following way. When a constrained point tends to move, an opposite force that will bring it back to its original position is calculated and added to the total external force for that point. This method for calculating constraint forces gives good results for small time steps. For larger time steps, the model points are subject to small oscillations since this approach corresponds to a corrective action.

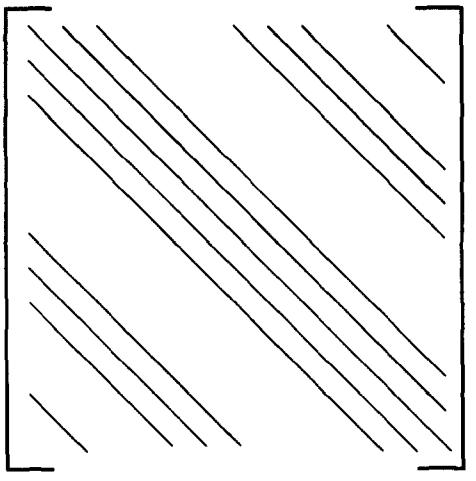

Figure 3 Band structure of stiffness matrix $\mathbf{K}$

The constrained points are specified by the user interactively. The system displays a grid specifying the body coordinates of each point existing in the model to be animated, and the user selects the points to be constrained during the animation (the points that will not move during the animation) using mouse buttons (see *Figure 4*). In other words, any point on a model can be constrained to a fixed location in space so that, when the model is animated, the constrained points remain in their initial positions. The constraint force that connects a material point u_0 on a deformable model to a point $\mathbf{p}_0$ in space by a spring is

$$\mathbf{f}_s(u,t)=k(\mathbf{p}_0-\mathbf{x}(u_0,t))\delta(u-u_0) \quad (25)$$

where k is the spring constant and δ is the unit delta function.

The forces due to the collision of deformable models with impenetrable obstacles are calculated using the obstacle's implicit (inside–outside) function. The obstacle exerts a repulsive force on the deformable model which can be calculated as a function of the obstacle's implicit function such that the force increases quickly if the model attempts to penetrate the obstacle. This is achieved by creating a potential energy function $c\exp(f(\mathbf{x})/\xi)$ around each obstacle, where f is the obstacle's implicit function, and c and ξ are constants determining the properties of the obstacle. In our system, the user can select different obstacles to exist in an animation sequence by the help of a menu. Ellipsoids, toroids and hyperboloids are possible choices for an obstacle. The repulsive force due to an impenetrable obstacle (expressed using the gradient ∇ of the potential energy function) is

$$\mathbf{f}_c(u,t)=-c((\nabla f(\mathbf{x})/\xi)\exp(-f(\mathbf{x})/\xi)\cdot\mathbf{n})\mathbf{n} \quad (26)$$

where $\mathbf{n}(u,t)$ is the unit surface normal vector of the deformable body's surface.

The mass density $\mu(u_1,u_2)$ and the damping density $\gamma(u_1,u_2)$ are discretized as grid functions $\mu[m,n]$ and $\gamma[m,n]$. Let $\mathbf{M}$ be the *mass matrix*, and $MN\times MN$

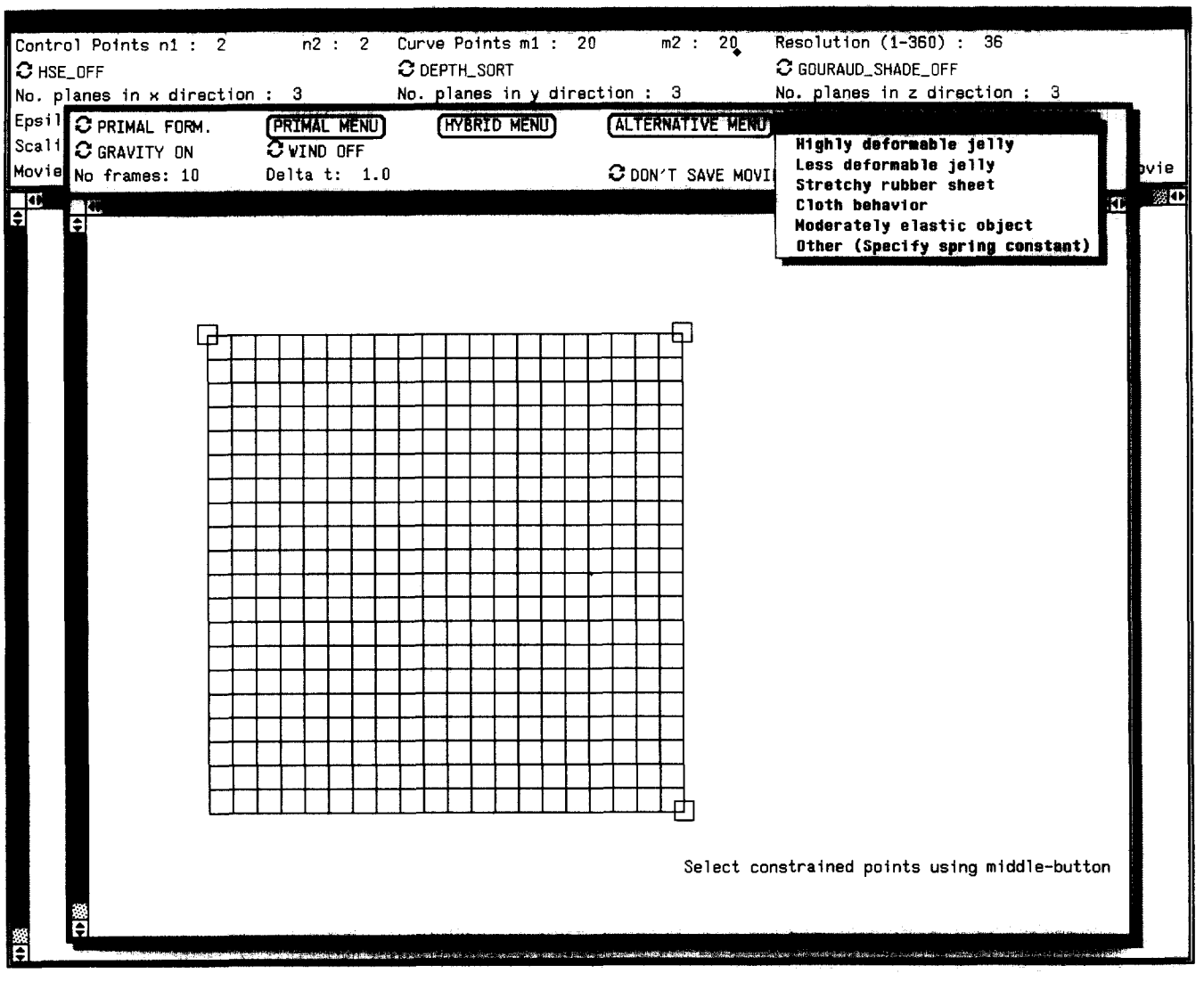

Figure 4 Screen dump during specification of parameters for an animation

diagonal matrix with the $\mu[m, n]$ variables as diagonal elements, and **C** be the *damping matrix*, constructed similarly from $\gamma[m, n]$. Then the Lagrange equations can be expressed in grid vector form by the simultaneous system of 2nd-order ordinary differential equations

$$\mathbf{M}\frac{\mathrm{d}^2\underline{\mathbf{x}}}{\mathrm{d}t^2}+\mathbf{C}\frac{\mathrm{d}\underline{\mathbf{x}}}{\mathrm{d}t}+\mathbf{K}(\underline{\mathbf{x}})\underline{\mathbf{x}}=\underline{\mathbf{f}}(\underline{\mathbf{x}}) \tag{27}$$

where the net external force on the surface $\mathbf{f}(u_1, u_2)$ has been discretized into the grid vector $\underline{\mathbf{f}}$, which represents the grid function $\underline{\mathbf{f}}[m, n]$.

We integrate this system through time using a step-by-step procedure. Evaluating $\mathbf{K}(\underline{\mathbf{x}})$ at time $t+\Delta t$ and $\underline{\mathbf{f}}$ at t, and substituting the discrete time approximations

$$\frac{\mathrm{d}^2\underline{\mathbf{x}}}{\mathrm{d}t^2}\approx(\underline{\mathbf{x}}_{t+\Delta t}-2\underline{\mathbf{x}}_t+\underline{\mathbf{x}}_{t-\Delta t})/\Delta t^2 \tag{28}$$

$$\frac{\mathrm{d}\underline{\mathbf{x}}}{\mathrm{d}t}\approx(\underline{\mathbf{x}}_{t+\Delta t}-\underline{\mathbf{x}}_{t-\Delta t})/2\Delta t \tag{29}$$

into Equation 27, we obtain the semiimplicit integration procedure

$$\mathbf{A}_t\underline{\mathbf{x}}_{t+\Delta t}=\mathbf{g}_t \tag{30}$$

where the $MN \times MN$ matrix

$$\mathbf{A}_t(\underline{\mathbf{x}}_t)=\mathbf{K}(\underline{\mathbf{x}}_t)+\left(\frac{1}{\Delta t^2}\mathbf{M}+\frac{1}{2\Delta t}\mathbf{C}\right) \tag{31}$$

and the effective force vector

$$\mathbf{g}_t=\underline{\mathbf{f}}_t+\left(\frac{1}{\Delta t^2}\mathbf{M}+\frac{1}{2\Delta t}\mathbf{C}\right)\underline{\mathbf{x}}_t+\left(\frac{1}{\Delta t}\mathbf{M}-\frac{1}{2}\mathbf{C}\right)\dot{\underline{\mathbf{x}}}_t \tag{32}$$

with

$$\dot{\underline{\mathbf{x}}}_t=(\underline{\mathbf{x}}_t-\underline{\mathbf{x}}_{t-\Delta t})/\Delta t \tag{33}$$

Applying the above semiimplicit procedure, we can evolve the dynamic solution from given initial conditions $\underline{\mathbf{x}}_0$ and $\dot{\underline{\mathbf{x}}}_0$ at $t=0$. During each time step, we solve a sparse linear algebraic system (Equation 30) for the instantaneous configuration $\underline{\mathbf{x}}_{t+\Delta t}$ using the preceding solution $\underline{\mathbf{x}}_t$ and $\dot{\underline{\mathbf{x}}}_t$.

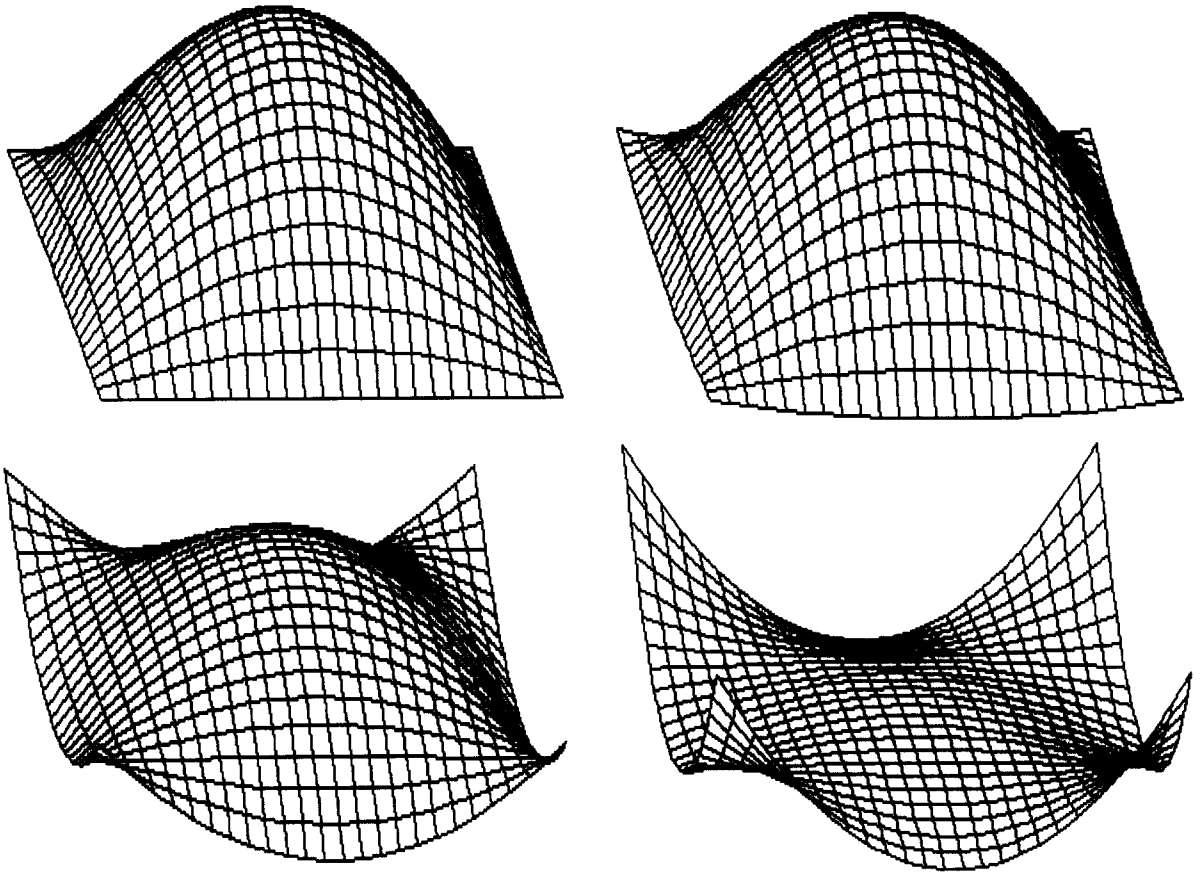

Figure 5 Highly nonrigid surface constrained from its four corners falls

The implementation of the hybrid formulation follows the same steps as those described for the primal formulation. The only difference is that the sparse, banded stiffness matrix **K** is constant. The equations of motion can be expressed in semidiscrete form by a system of coupled ordinary differential equations. The system contains two ordinary differential equations for the translational and rotational motion of the model as if all of its mass were concentrated at its centre of mass, and a system of ordinary differential equations whose size is proportional to the size of the discrete model. These equations are solved in tandem for each time step with respect to the initial conditions given.

SIMULATION EXAMPLES

We have implemented both primal and hybrid formulation in our system so that the user can interactively select between them. In this way, the primal formulation can be selected for highly nonrigid models, and the hybrid formulation can be selected for highly rigid models.

The system can be used to simulate the behaviour of elastically deformable materials, such as cloth, paper, and metals. We can also utilize the system in CAD and CAM applications. For example, automobile bodies could be designed by applying external forces to metal surfaces, and imposing constraints on model points.

In *Figure 5*, we have used the primal formulation, and the material properties are adjusted to simulate a membrane that is not resistant to elongation or contraction, and not resistant to bending. The model is made up of a jelly-like material. In this example, a discrete model of size 20×20 is constrained from its four corners and falls by the effect of gravitational force.

In *Figure 6*, we have used the hybrid formulation and set the material properties to simulate a moderately rigid object (such as a thin metal plate). The model is constrained from different points on its edges and a downward force is exerted on it. We can use such models in CAD and CAM applications to design mechanical parts.

In *Figure 7*, a flat surface that is not resistant to elongation or contraction and not resistant to bending falls on an impenetrable obstacle which is a toroid. The surface takes the shape of the toroid when it collides with

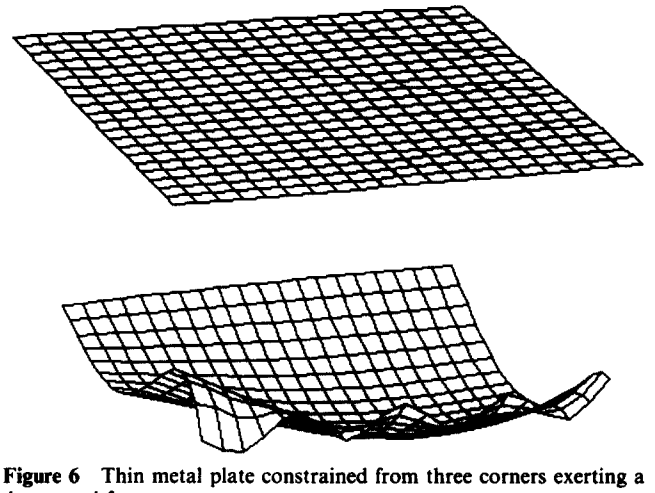

Figure 6 Thin metal plate constrained from three corners exerting a downward force

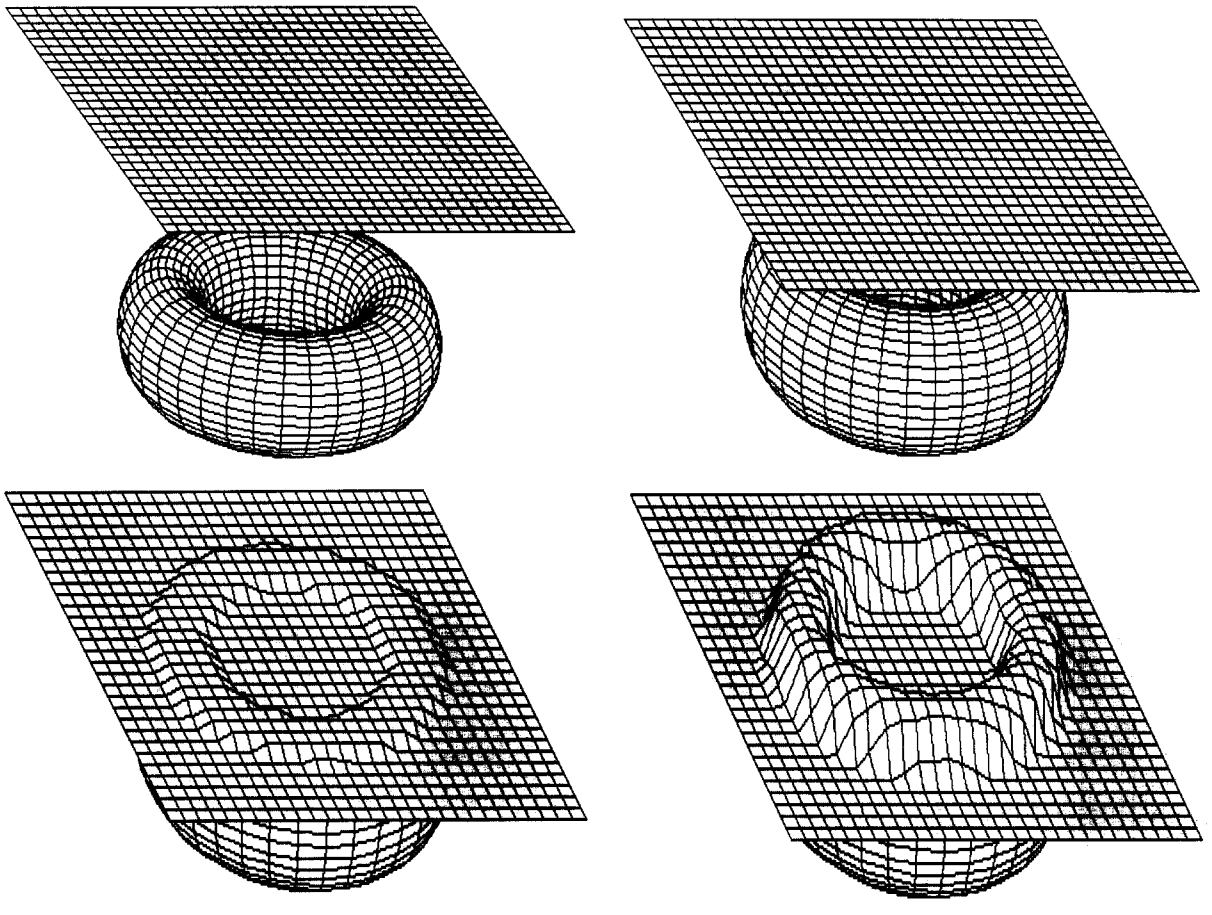

Figure 7 Highly nonrigid surface collides with toroid

it. To obtain better results in collision simulations, we must either take a very small time step, or use adaptive time stepping. Otherwise, we may detect collisions very late, after the model points penetrate the obstacle too deeply.

QUANTITATIVE COMPARISON OF FORMULATIONS

We have compared the processing times for generating an animation frame using the two formulations. To compute processing times, simple Bézier surfaces are animated using the two formulations. *Figures 8* and *9* give the preprocessing and processing times of the animations of the Bézier surfaces of different sizes. The processing times for each frame given in the tables include

- the time taken to calculate the external forces for each model point,
- the time taken to calculate the entries of the stiffness matrix*,
- the time taken to calculate the 3D positions of model points,
- the wireframe rendering time for the calculated frame.

Although it seems from the graphs that the hybrid formulation is superior to the primal formulation, they complement each other for different elasticity properties.

*This is for the primal formulation; for the hybrid formulation, it is included in the preprocessing time.

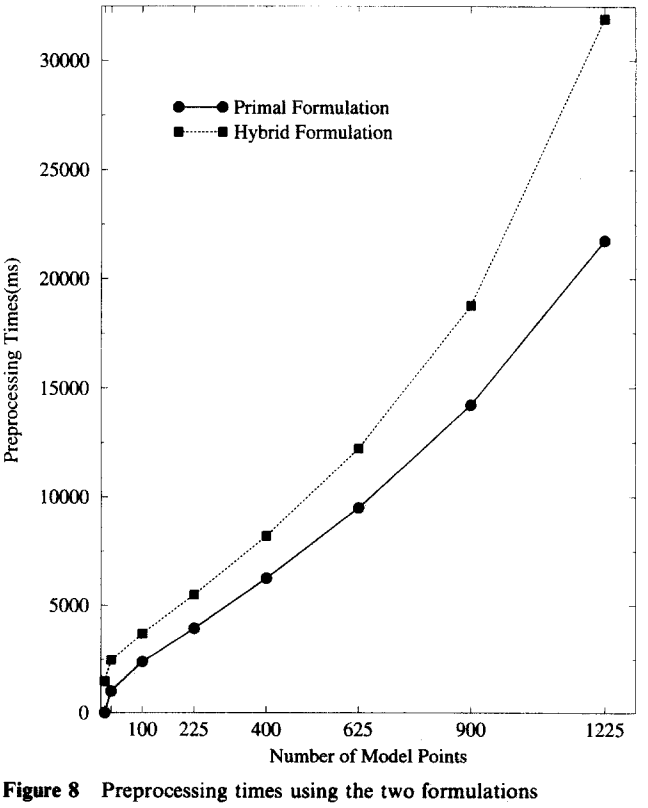

Figure 8 Preprocessing times using the two formulations

The nonquadric energy functional in primal formulation causes a nonlinear elastic force associated with the deformable body to appear in the partial differential equations of motion. Nonlinearity results because the

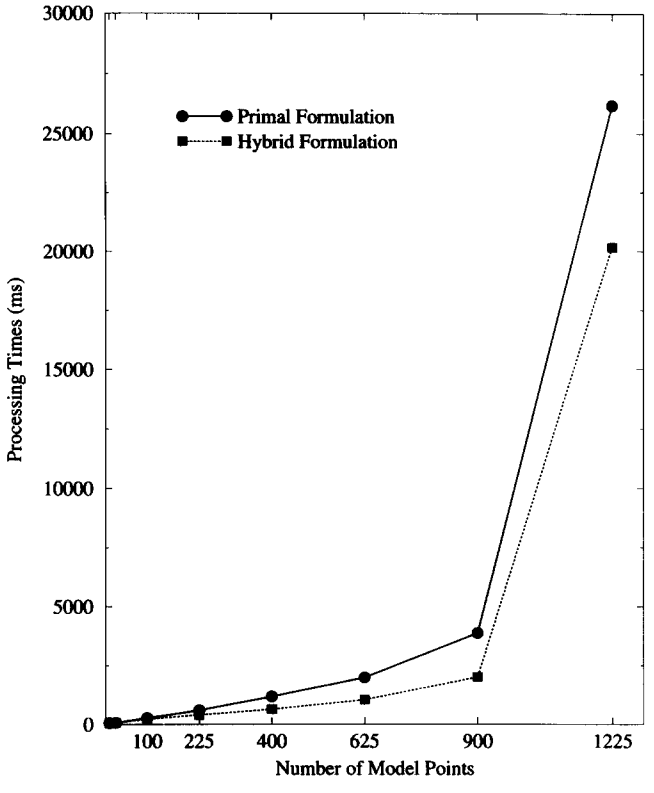

Figure 9 Processing times of one frame using the two formulations

elastic force attempts to restore the shape of the deformed body to a rest shape. The advantage of nonlinear elasticity is that it is in principle the most accurate way to characterize the behaviours of certain elastic phenomena. However, it can lead to serious practical difficulties in the numerical implementation of deformable models for animation. The hybrid formulation offers a practical advantage for fairly rigid models, whereas primal formulation becomes unpractical because of the nonquadric energy functional with increasing rigidity and complexity of the models[7].

An important advantage of primal formulation over hybrid formulation is that it is easier to establish an intuitive link between the weighting functions of the deformable models and the resulting elastic behaviour. This is because of the nature of the weighting functions.

CONCLUSIONS

Physically based modelling has emerged as a means of creating realistic animation. It proposes methods of creating active models which react to applied forces, to constraints, to ambient media, or to impenetrable obstacles, as one would expect from real physical objects. In this way, computer animators are unconcerned with the kinematic details of animations, knowing that physics will dictate the low-level motions.

Physically based modelling adds new levels of representation to object descriptions in addition to geometry. Forces, torques, velocities, kinetic and potential energies, heat, and other physical quantities are used to control the creation and evolution of models. To construct the differential equations of the motion of the models, different techniques, such as Lagrange equations and constraint-based methods, could be used. After constructing the equations of motion for the models, the equations should be solved using fast numerical methods.

In this paper, we explained a system for animating deformable models using a physically based approach. The system uses both the primal formulation and the hybrid formulation for animating these models so that the user can decide which one to use in an animation, taking into consideration the advantages and disadvantages of each formulation.

REFERENCES

1 Güdükbay, U and Özgüç, B 'Free-form solid modeling using deformations' *Comput. & Graph.* Vol 14 No 3 (1990) pp 491–500
2 Thalmann, N M and Thalmann, D 'Six-hundred indexed references on computer animation' *J. Visualization & Comput. Animat.* Vol 3 No 3 (1992) pp 147–174
3 Platt, J and Barr, A H 'Constraint methods for flexible models' *Comput. Graph.* Vol 22 No 4 (1988) pp 279–288 (*Proc. Siggraph '88*)
4 Pentland, A and Williams, J 'Good vibrations: modal dynamics for graphics and animation' *Comput. Graph.* Vol 23 No 3 (1989) pp 215–222 (*Proc. Siggraph '89*)
5 Witkin, A., Fleischer, K and Barr, A H 'Energy constraints on parameterized models' *Comput. Graph.* Vol 21 No 4 (1987) pp 225–232 (*Proc. Siggraph '87*)
6 Terzopoulos, D, Platt, J, Barr, A H and Fleischer, K 'Elastically deformable models' *Comput. Graph.* Vol 21 No 4 (1987) pp 205–214 (*Proc. Siggraph '87*)
7 Terzopoulos, D and Witkin, A 'Physically based models with rigid and deformable components' *Comput. Graph. & Applic.* Vol 8 No 6 (1988) pp 41–51
8 Do Carmo, M P *Differential Geometry of Curves and Surfaces* Prentice-Hall, USA (1974)

Uğur Güdükbay was born in Niğde, Turkey, in 1965. He received a BSc in computer engineering from the Middle East Technical University, Turkey, in 1987. He received an MSc and a PhD from Bilkent University, Turkey, in 1989, and 1994, respectively, in the Computer Engineering and Information Science Department. His research interests include physically based modelling and animation, and deformable models.

Bülent Özgüç joined the Bilkent University Faculty of Engineering, Turkey, in 1986. He is a professor of computer science and the dean of the Faculty of Art, Design and Architecture. He has taught at the University of Pennsylvania, USA, the Philadelphia College of Arts, USA, and the Middle East Technical University, Turkey, and he worked as a member of the research staff at the Schlumberger Palo Alto Research Center, USA. For the last 15 years, he has been active in the field of computer graphics and animation. He received a BArch and an MArch in architecture, both from Middle East Technical University, Turkey, in 1972 and 1973. He received an MS in architectural technology from Columbia University, USA, and a PhD in a joint program of architecture and computer graphics at the University of Pennsylvania, in 1974 and 1978.