Comput. & Graphics Vol. 17, No. 1, pp. 71–77, 1993
Printed in Great Britain.

0097-8493/93 $6.00 + .00
© 1993 Pergamon Press Ltd.

Technical Section

AN ANIMATION SYSTEM FOR RIGID AND DEFORMABLE MODELS

UĞUR GÜDÜKBAY and BÜLENT ÖZGÜÇ
Bilkent University, Department of Computer Engineering and Information Science, 06533 Bilkent, Ankara, Turkey

and

YILMAZ TOKAD
Bilkent University, Department of Mathematics, 06533 Bilkent, Ankara, Turkey

Abstract—We describe a system for the animation of rigid and deformable models. The system uses the approaches from elasticity theory for animating the models. Two different formulations, namely the *primal* and the *hybrid* formulations, are implemented so that the user could select the suitable one for an animation depending on the rigidity of the models. Collision of the models with impenetrable obstacles and constraining model points to fixed positions in space are implemented for use in the animations.

1. INTRODUCTION

This paper discusses an animation system that is implemented for the animation of rigid and nonrigid (deformable) models. The system is built on top of a modeling system for representing 3D free-form objects, that uses Superquadrics[1] and Bézier surfaces[2] as modeling techniques, and regular deformations[3] and Free-Form Deformations[4] for deforming these models to obtain irregular, free-form objects[5]. The static models obtained by these methods can be animated using the techniques discussed in this paper. The system is implemented using C language[6] on a Unix* workstation environment (it runs on Sun_3 and Sparc workstations). The implementation uses the facilities provided by Sun View† system such as windows, panels, and menus[7].

The use of computer graphics and numerical methods for 3D design and modeling provides an interactive environment in which designers can formulate and represent shapes of objects. Modeling the shapes as a composition of geometrically and algebraically defined primitives, simulating scenes with shading and texture, and producing usable design images are the most important requirements for application areas such as Computer-Aided Design and Computer-Aided Manufacturing.

Currently, most of the methods used for modeling are *kinematic.* This becomes a major drawback when we want to create realistic animation because these methods are passive; they do not interact with each other or with external forces. To achieve realism in animation a model should be able to follow predefined paths while still moving in an interesting manner and interacting with other models as real physical objects would do.

To build and animate active models, physically-based techniques should be used. These techniques facilitate the creation of models capable of automatically synthesizing complex shapes and realistic motions that are attainable only by skilled animators. *Physically-based modeling* 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 based on these properties. To this end, solution of the *initial value problems* is required so that the course of a simulation is determined by objects' initial positions and velocities, by the forces and torques applied to the object along the way, *etc.*

Another aspect in realistic animation is modeling the behavior of *deformable objects.* To simulate the behavior of deformable objects, we should approximate a continuous model by using discretization methods, such as finite difference, and finite element method. 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. Simulating the physical properties (such as tension and rigidity) static shapes exhibited by a wide range of deformable objects (including string, rubber, cloth, paper, and flexible metals) can be modeled. For example, to obtain the effect of an elastic surface, the grid points are connected by springs. The physical quantities cited earlier should be used to simulate dynamics of these objects. Various researchers[8–13] presented discrete models that are based on *elasticity* and *plasticity theory* and use *energy fields* to define and enforce constraints for animating deformable objects.

The plan of this paper is to first present a short description of the methods proposed by Terzopoulos et al.[11, 12, 14] for elastically deformable models, and inelastic models. We will then explain the implementation details of these methods in the context of our system, and algorithmic solution of the problems, such

* Unix is a trademark of AT&T Laboratories.
† Sun View is a registered trademark of Sun Microsystems.

as collision of flexible models with impenetrable obstacles, etc. Then, some simulation results representing the features of the system are given.

2. 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 behavior of nonrigid curves, surfaces, and solids as a function of time. Real materials exhibit both elastic and inelastic behavior. 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, static shapes of a wide range of deformable objects, including string, rubber, cloth, paper, and flexible metals, can be modeled. Dynamics of these materials can be simulated by including physical properties, such as mass and damping. The simulation involves numerical solution of the partial differential equations that govern the evolving shape of the deformable object and its motion through space[12].

Viscous and plastic processes within the models evolve a reference component, which describes the natural shape, according to yield and creep relationships that depend on applied force and/or instantaneous deformation. Simple fracture mechanics results from internal processes that introduce local discontinuities as a function of the instantaneous deformations measured through the model[15].

2.1. *Deformable models*

Deformable models can be formulated by using the intrinsic or material coordinates of points in a body 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 time-varying vector-valued function $\mathbf{r}(\mathbf{u}, t) = [r_1(\mathbf{u}, t), r_2(\mathbf{u}, t), r_3(\mathbf{u}, t)]$. The body in its natural rest state is given by $\mathbf{r}^0(\mathbf{u}) = [r_1^0(\mathbf{u}), r_2^0(\mathbf{u}), r_3^0(\mathbf{u})]$ (Fig. 1). The equations of motion for a deformable model can be written in Lagrange's form as

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

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 $\mathbf{u}$, $\mathbf{f}(\mathbf{r}, t)$ is the net externally applied force, and $\varepsilon(\mathbf{r})$ is the energy functional that 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

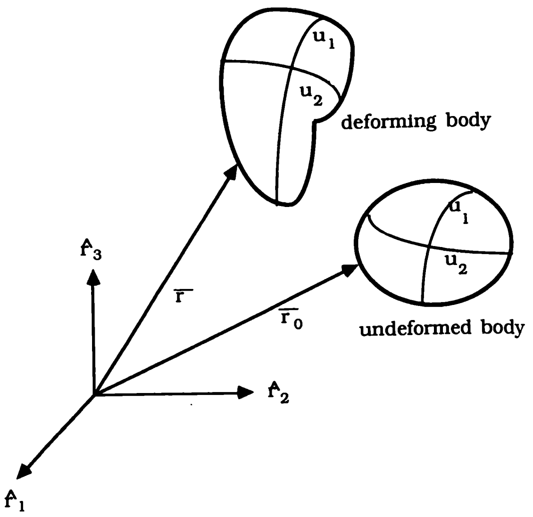

Fig. 1. Geometric representation of a deformable body for primal formulation. Reprinted with permission from "Elastically Deformable Models" by D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. *ACM Computer Graphics, 21* (*Proc. SIGGRAPH* 1987) 205–214. Copyright 1987, Association for Computing Machinery, Inc.

material coordinates of two nearby points in the body. The distance between these points in the deformed body in Euclidean 3-space is given by:

$$dl = \sum_{i,j} G_{ij} du_i du_j, \qquad (2)$$

where the symmetric matrix:

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

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 $\mathbf{u} = [u_1, u_2, u_3]$. Two surfaces have the same shape if their metric tensors $\mathbf{G}$ as well as their curvature tensors $\mathbf{B}$ are identical forms of $\mathbf{u} = [u_1, u_2]$. The components of the curvature tensor are:

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

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 $s(\mathbf{r}(u))$, curvature $\kappa(\mathbf{r}(u))$, and torsion $\tau(\mathbf{r}(u))$ are identical forms of $\mathbf{u} = [u_1]$. See[16] for a detailed discussion of these formulations.

Using the above differential quantities, potential energies of deformation for use in Lagrangian 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 increases as the model gets 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

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

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

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

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

$$\varepsilon(\mathbf{r}) = \int_{\Omega} \|\mathbf{G} - \mathbf{G}^0\|^2_{\mathbf{w}^1} du_1 du_2 du_3 \quad (7)$$

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

These energies denote the amount of energy 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 forces, such as gravitational force, spring forces, viscous forces, *etc.*

The weighting functions in the above energies ($w^1_{ij}(u_1, u_2)$ and $w^2_{ij}(u_1, u_2)$ for an elastic surface) determine the properties of the simulated deformable material. The weighting function $w^1_{ij}(u_1, u_2)$ determines surface tensions and sheer strengths that minimize the deviations 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 that 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 $w^1_{12} = w^1_{21}$ determining the resistance to twist deformation. To simulate a stretchy 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 by increasing the value of w^2_{ij}. Since $w^1_{ij}(u)$ and $w^2_{ij}(u)$ are functions of material coordinates u, we may vary the material properties over the surface, and we may introduce local singularities such as fractures and creases[11, 12].

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.

The above formulation for elastically deformable models is called *primal formulation.* Another formulation, called *hybrid formulation* represents a deformable body as the sum of a reference component $\mathbf{r}(\mathbf{u}, t)$ and a deformation component $\mathbf{e}(\mathbf{u}, t)$ (Fig. 2). The positions of mass elements in the body relative to ϕ is given by

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

In this formulation, 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 a reference frame ϕ whose origin coincides with the body's center of mass and it should be evolved over time according to the rigid body dynamics to have a rigid body motion besides its elastic motion.

The nonquadric energy functional in primal formulation causes a nonlinear elastic force asssociated with the deformable body to appear in the partial differential equations of motion. Nonlinearity results because the 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 behavior 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 due to the nonquadric energy functional with increasing rigidity and complexity of the rest shapes[11, 14].

3. IMPLEMENTATION OF THE PRIMAL FORMULATION

Steps of animation of deformable models using primal formulation are as follows:

First, we should calculate the total external force for each point of the model that is discretized using body

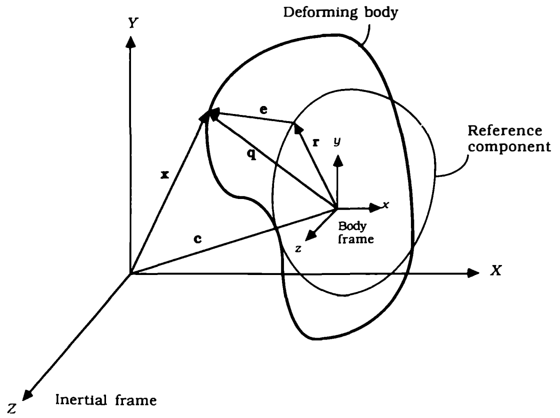

Fig. 2. Geometric representation of deformable models for hybrid formulation. Reprinted with permission from "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture" by D. Terzopoulos and K. Fleischer, *ACM Computer Graphics,* 22 (*Proc. SIGGRAPH* 1988) 269–287. Copyright 1988, Association for Computing Machinery, Inc.

coordinates of the model. In order to achieve this, we should add the forces effecting a point, which are 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 for bringing it back to its original position is calculated and added to the total external force for that point. Each constrained point has an effect on the total external force for all points in the model depending on the difference between the body coordinates of the points. This effect is calculated according to an exponential distribution function. This method for calculating constraint forces gives good results for small time steps. For larger time steps, the model points make small oscillations since this approach corresponds to a corrective action.

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 (Fig. 3). In other words, any point on a model could 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

$$\mathbf{f}_s(u, t) = k(\mathbf{p}_0 - \mathbf{x}(u_0, t))\delta(u - u_0), \qquad (9)$$

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

More complicated constraints are being studied for future versions of our implementation (such as point-to-point constraints, for connecting two points on different models to simulate articulated figures, point-to-line constraints, for controlling the motion of the models, *etc.*). However, some implementation problems occur with the current formulation.

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 that can be calculated as a function of the obstacle's implicit function such that the force grows quickly if the model attempts to penetrate the obstacle. This is achieved by creating a potential function $c \exp(\nabla f(\mathbf{x})/\epsilon)$ around each obstacle, where f is the obstacle's implicit function, ∇ denotes the gradient, and c and ϵ 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, hyperboloids are possible choices for an obstacle. The repulsive force due to an impenetrable obstacle is

$$\mathbf{f}_c(u, t) = -c((\nabla f(\mathbf{x})/\epsilon)\exp(-f(\mathbf{x})/\epsilon)\cdot \mathbf{n})\mathbf{n}, \qquad (10)$$

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

The second step, after calculating the total external force for each point on the discrete model and specifying the value of the weighting functions, is the discretization of the partial differential equations using finite difference methods on the discrete mesh of nodes. For this purpose, we calculate the finite differences that are used to calculate the stiffness matrix $\mathbf{K}$. Expressing the grid functions $\mathbf{x}[m, n]$ and $\epsilon[m, n]$ as $\underline{\mathbf{x}}$ and $\underline{\epsilon}$ in grid vector notation, which are denoting the 3D po-

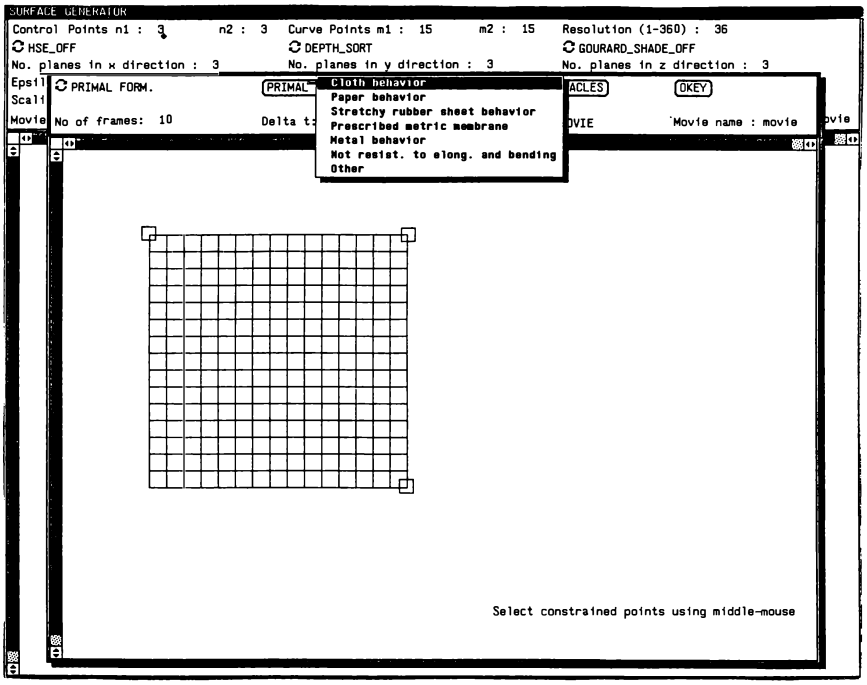
```
SURFACE GENERATOR
Control Points n1 :  3        n2 :  3    Curve Points m1 :  15      m2 :  15    Resolution (1-360) :  36
HSE_OFF                                  DEPTH_SORT                             GOURARD_SHADE_OFF
No. planes in x direction :  3           No. planes in y direction :  3         No. planes in z direction :  3
PRIMAL FORM.
No of frames:  10                        Delta t:                               Movie name : movie
Select constrained points using middle-mouse
```
Epsil
Scali
Movie
PRIMAL
Cloth behavior
Paper behavior
Stretchy rubber sheet behavior
Prescribed metric membrane
Metal behavior
Not resist. to elong. and bending
Other
ACLES
OKEY
OVIE
ovie

Fig. 3. Screen dump during the specification of the parameters for an animation.

sitions of model points and elastic force for each model point stored in $M \times N$ vector for an $M \times N$ discrete grid of a deformable model, elastic force can be written in vector form as

$$\underline{\epsilon} = \mathbf{K}(\underline{\mathbf{x}}) \cdot \underline{\mathbf{x}}, \qquad (11)$$

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 second-order ordinary differential equations. The band structure of the matrix $\mathbf{K}$ is shown in Fig. 4.

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,* an $MN \times MN$ diagonal matrix with the $\mu[m, n]$ variables as diagonal elements, and $\mathbf{C}$ 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 second-order ordinary differential equations

$$\mathbf{M}\frac{d^2\underline{\mathbf{x}}}{dt^2} + \mathbf{C}\frac{d\underline{\mathbf{x}}}{dt} + \mathbf{K}(\underline{\mathbf{x}})\underline{\mathbf{x}} = \underline{\mathbf{f}}(\underline{\mathbf{x}}), \qquad (12)$$

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

$$\frac{d^2\underline{\mathbf{x}}}{dt^2} \approx (\underline{\mathbf{x}}_{t+\Delta t} - 2\underline{\mathbf{x}}_t + \underline{\mathbf{x}}_{t-\Delta t})/\Delta t^2, \qquad (13)$$

$$\frac{d\underline{\mathbf{x}}}{dt} \approx (\underline{\mathbf{x}}_{t+\Delta t} - \underline{\mathbf{x}}_{t-\Delta t})/2\Delta t \qquad (14)$$

into Eq. (12), we obtain the semi-implicit integration procedure

$$\mathbf{A}_t\underline{\mathbf{x}}_{t+\Delta t} = \underline{\mathbf{g}}_t, \qquad (15)$$

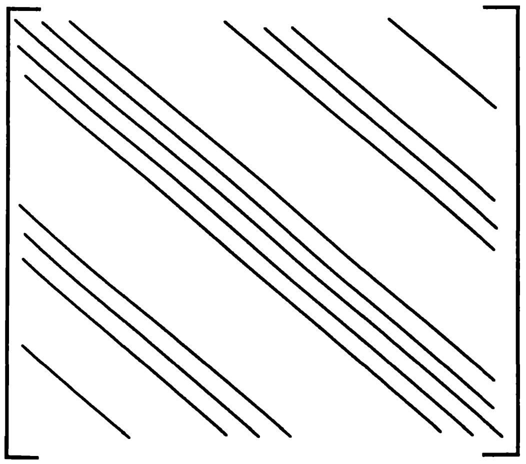
Fig. 4. The band structure of the stiffness matrix K.

where the $MN \times MN$ matrix

$$\mathbf{A}_t(\underline{\mathbf{x}}_t) = \mathbf{K}(\underline{\mathbf{x}}_t) + \left(\frac{1}{\Delta t^2}\mathbf{M} + \frac{1}{2\Delta t}\mathbf{C}\right) \qquad (16)$$

and the effective force vector

$$\underline{\mathbf{g}}_t = \underline{\mathbf{f}}_t + \left(\frac{1}{\Delta t^2}\mathbf{M} + \frac{1}{2\Delta t}\mathbf{C}\right)\underline{\mathbf{x}}_t + \left(\frac{1}{\Delta t}\mathbf{M} - \frac{1}{2}\mathbf{C}\right)\dot{\underline{\mathbf{x}}}_t, \qquad (17)$$

with

$$\dot{\underline{\mathbf{x}}}_t = (\underline{\mathbf{x}}_t - \underline{\mathbf{x}}_{t-\Delta t})/\Delta t. \qquad (18)$$

Applying the above semi-implicit 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 [Eq. (15)] for the instantaneous configuration $\underline{\mathbf{x}}_{t+\Delta t}$ using the preceding solution $\underline{\mathbf{x}}_t$ and $\dot{\underline{\mathbf{x}}}_t$[12].

Implementation of the hybrid formulation follows the same steps described for the primal formulation. The only difference is the sparse, banded stiffness matrix $\mathbf{K}$ 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 is concentrated at its center of mass, and a system of ordinary differential equations whose size is proportional with the size of the discrete model. These equations are solved in tandem for each time step with respect to the initial conditions given[14].

4. 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 linear system of equations obtained by a semi-implicit time integration procedure as explained in the previous section can be solved by different linear system solvers. Since the stiffness matrix is sparse and has a special band structure, Conjugate Gradient method for solving linear systems as described in [17] can be used. This method uses the sparsity property to gain efficiency in solving the equations. We have also used LU decomposition and back substitution to solve the equations.

In Fig. 5, we have used the primal formulation and the material properties are adjusted to simulate a membrane not resistant to elongation or contraction, and not resistant to bending ($w^1 = 0$, $w^2 = 0$). In this example, a discrete model of size 16×16 is constrained

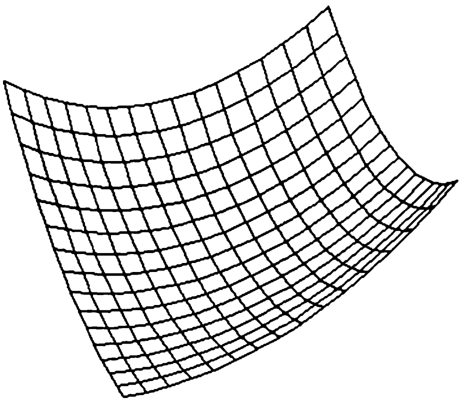

Fig. 5. A highly nonrigid surface constrained from 3 corner falls (initially, the surface is flat).

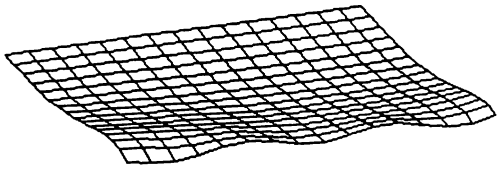

Fig. 6. A moderately rigid surface constrained from different points on its edges exerted a downward force (Initially, the surface is flat).

from three corners and falls by the effect of the gravitational force.

In Fig. 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.

In Fig. 7, an elastic model not resistant to elongation or contraction and not resistant to bending falls on an impenetrable obstacle that is an ellipsoid. The deformable model takes the shape of the obstacle when it collides with it as it is seen. To get better results in collision simulations, either we should take a very small time step or we should use an adaptive time stepping. Otherwise, we may detect collisions very late, namely after the model points penetrate the obstacle too much.

Shaded versions of these simulations are given in Figs. 8, 9, and 10.

5. CONCLUSION

In creating good computer animation, the focus is not on the problem of completing a given motion task, but more importantly on how this task is to be performed by the animated character. All the elements involved in an animated character must cooperate in synchronized harmony. Most of the animation systems generated up to now leave the burden of generating realistic animation to the animator. To remedy this problem, fundamental principles of traditional animation, such as *squash and stretch, exaggeration, follow through,* and *overlapping action*[18] should be formalized as high level constructs.

Physically-based modeling has emerged as a means of creating realistic animation. It proposes methods to create active models that 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 modeling adds new levels of representation to object description 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 Lagrangian equations, constraint methods, the principle of virtual work, can be used. Constraint methods, which are highly suitable for this purpose, unify the creation of complex models with the control of the motion of the models. 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 rigid and deformable models. 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 considering the advantages and disadvantages of each formulation. Our aim as a future work is to modify the equations of motion proposed for deformable models[12] in such a way that constraint forces will be taken into account as external forces. This approach allows modeling and animating articulated bodies consisting of rigid and nonrigid parts by creating complex models from sim-

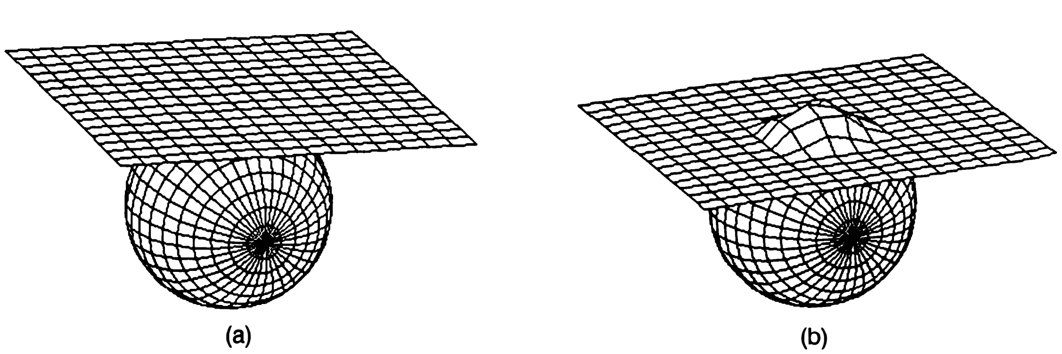

Fig. 7. A highly nonrigid surface collides with an ellipsoid (a) Initial frame; (b) final frame.

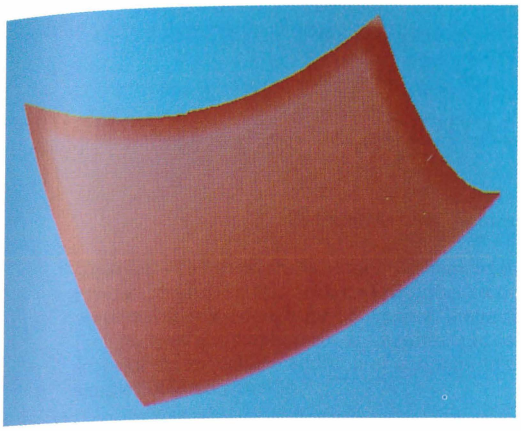
Fig. 8. Shaded version of the simulation in Fig. 5.

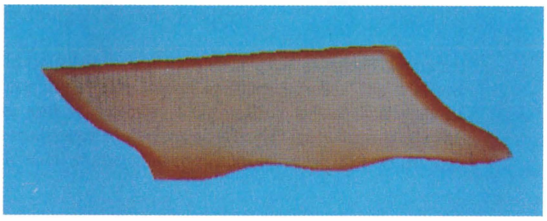
Fig. 9. Shaded version of the simulation in Fig. 6.

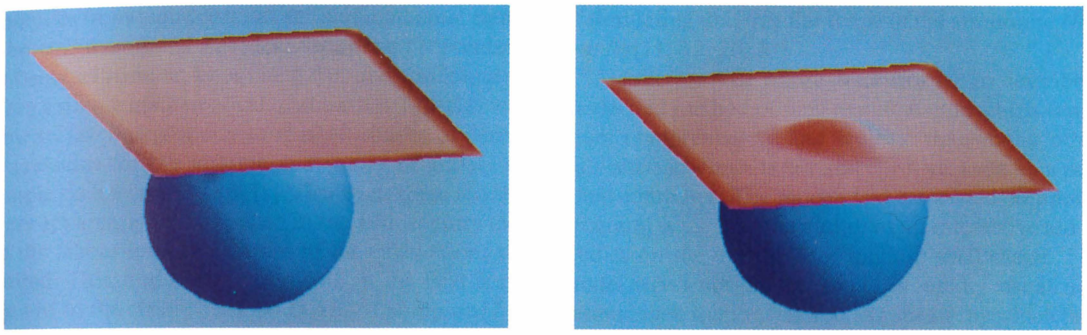
Fig. 10. Shaded version of the simulation in Fig. 7.

pler primitives using point-to-point constraint. Also, other constraints, such as point-to-path *etc.*, can be used to control the motion of the models.

REFERENCES

1. A. H. Barr, Superquadrics and angle-preserving transformations. *IEEE Comp. Graph. Appl.* **1**(1) 11–23 (January 1981).
2. P. Bézier, *Numerical Control—Mathematics and Applications,* John Wiley and Sons, London (1972).
3. A. H. Barr, Global and local deformations of solid primitives. *Comp. Graph.* **18**(3) 21–30 (July 1984).
4. T. W. Sederberg and S. R. Parry, Free-form deformation of solid geometric models, *ACM Computer Graphics* (*Proc. SIGGRAPH*), **20**(4) 151–160 (August 1986).
5. U. Güdükbay and B. Özgüç, Free-form solid modeling using deformations, *Computers & Graphics,* **14**(3) 491–500 (1990).
6. B. W. Kernighan and D. M. Ritchie, *The C Programming Language,* Prentice Hall, Inc., Englewood Cliffs, NJ (1978).
7. Sun Microsystems, *Sun View Programmer's Guide,* Mountain View, CA (1986).
8. J. E. Chadwick, D. R. Haumann, and R. E. Parent, Layered construction for deformable animated characters, *ACM Computer Graphics,* **23**(3) (*Proc. SIGGRAPH 1989*), 243–252 (July 1989).
9. J. Platt and A. Barr, Constraint methods for flexible models, *ACM Computer Graphics,* **22**(4) (*Proc. SIGGRAPH 1988*) 279–288 (August 1988).
10. A. Pentland and J. Williams, Good vibrations: Modal dynamics for graphics and animation *ACM Computer Graphics,* **23**(3) (*Proc. SIGGRAPH* 1989) 215–222 (July 1989).
11. D. Terzopoulos and K. Fleischer, Modeling inelastic deformation: Viscoelasticity, plasticity, fracture, *ACM Computer Graphics* **22**(4) (*Proc. SIGGRAPH* 1988) 269–278 (August 1988).
12. D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, Elastically deformable models, *ACM Computer Graphics* **21**(4) (*Proc. SIGGRAPH* 1987) 205–214 (July 1987).
13. A. Witkin, K. Fleischer, and A. Barr, Energy constraints on parameterized models, *ACM Computer Graphics* **21**(4) (*Proc. SIGGRAPH* 1987) 225–232 (July 1987).
14. D. Terzopoulos and A. Witkin, Physically based models with rigid and deformable components, *IEEE Comp. Graph. Appl.* **8**(6) 41–51 (November 1988).
15. J. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture,* Halstead Press, Horwood, NY (1981).
16. M. P. Do Carmo, *Differential Geometry of Curves and Surfaces,* Prentice-Hall, Englewood Cliffs, NJ (1974).
17. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing,* Cambridge University Press, Cambridge, U.K. (1986).
18. J. Lasseter, Principles of traditional animation applied to 3D computer animation, *ACM Computer Graphics* **21**(4) (*Proc. SIGGRAPH* 1987) 35–44 (July 1987).