DyRT: Dynamic Response Textures for Real Time Deformation Simulation with Graphics Hardware

Doug L. James and Dinesh K. Pai
Department of Computer Science, University of British Columbia

Abstract

In this paper we describe how to simulate geometrically complex, interactive, physically-based, volumetric, dynamic deformation models with *negligible main CPU costs.* This is achieved using a ***Dy**namic **R**esponse **T**exture*, or **DyRT**, that can be mapped onto any conventional animation as an optional rendering stage using commodity graphics hardware. The DyRT simulation process employs precomputed modal vibration models excited by rigid body motions. We present several examples, with an emphasis on bone-based character animation for interactive applications.

1 Introduction

In this paper we present an efficient rendering technique for simulating real time dynamic deformations for applications such as character animation. This is achieved using a ***Dy**namic **R**esponse **T**exture*, or **DyRT**, that can be mapped onto any conventional animation (motion capture or keyframe or rigid body dynamics simulation) as an optional rendering stage. This is because the complexity of rendering deformations using DyRT is comparable to lighting the object. Therefore, *every* deformable object, large or small, can be rendered with realistic dynamic deformation responses, in real time, on commodity graphics hardware.

The physical realism of DyRT is due to the use of precomputed modal analyses [22] of dynamic elastic models computed using, e.g., the Finite Element Method (FEM) [29]. These systems typically have a few clearly dominant dynamic deformation modes that enable us to produce convincing realizations on commodity graphics cards. This is achieved using vertex programs [14] that perform the per-vertex linear superpositions necessary to compute displacement and normal vectors.

A second key component of a DyRT is the use of *rigid motion transfer functions* for rigid (bone) motion input dependence, so that DyRTs respond physically and are not just "canned vibrations." This is in contrast to, e.g., several nVidia vertex program demos [14], such as "warp," which, although extremely useful in context, have limited physical foundations.

In the remainder of this paper we describe the foundations for and the process of applying DyRT to objects, with a particular emphasis on bone-based character animation.

1.1 Related Work

Significant work has been done on simulating dynamic deformable objects, in areas such as human body modeling and interactive simulation. Despite the large amount of pioneering work on deformation [25, 28, 15, 1], there continue to be exciting new applications [18, 17] and improvements in simulation efficiency [2, 6].

Numerous examples of human body modeling exist in the literature with particular areas of interest being deformations of skin and muscles [27, 9], faces [12], and layered models [5]. Support exists in commercial animation packages, such as Maya, for simulating tissue dynamics. There have also been significant recent developments for interactive dynamic tissue simulation, especially for force feedback applications such as surgical simulation [6, 20]. Despite these advances, the simulation of transient vibration responses for secondary animation remains largely absent from the traditional character animation pipeline, and especially so in video games.

Of particular interest for graphics hardware are data-driven deformation models based on linear superposition of precomputable *global deformation bases* [3], which include space warping methods such as FFD. While such models can provide fast simulation and constraint handling for physically-based dynamic [19, 28] and also static [11, 10] deformable models, we are primarily interested in their amenability to graphics hardware simulation [14].

For simulating *free vibrations* of elastic models with modest amplitudes, global deformation bases based on Karhunen-Loeve expansions from modal analysis provide the optimal description [22, 8]. First introduced to the graphics community by the pioneering work of Pentland and Williams [19, 8], more recently they have been used for interactive applications involving precomputed or measured modal data: stochastic simulation of tree-like structures [23], force feedback [4], and contact sound simulation [26].

Our contribution: This is the first paper to show how to simulate geometrically complex, interactive, physically-based, volumetric, dynamic deformation models in real time with *negligible main CPU costs.* We do so with precomputed modal vibration models stored in graphics hardware memory and driven by a handful of inputs defined by rigid body motion.

2 Background on Modal Vibration Models

We briefly summarize the necessary background on modal vibration analysis here, and refer the reader to a suitable text [22]. The linear elastodynamic equation for a finite element model [29],

$$\mathsf{M}\ddot{\mathsf{u}} + \mathsf{C}\dot{\mathsf{u}} + \mathsf{K}\mathsf{u} = \mathsf{F}, \qquad (1)$$

describes the displacements $\mathsf{u}=\mathsf{u}(t)$ of N nodes within a volume. The displacement field u is expanded in a modal displacement basis

$$\mathsf{u}(t) = \Phi\,\mathsf{q}(t) \qquad (2)$$

where Φ denotes the model's *modal matrix,* a matrix whose i^{th} column $\Phi_{:i}$ represents the i^{th} *mode shape*, and $\mathsf{q}=\mathsf{q}(t)$ are the corresponding *modal amplitudes*, i.e., q_i is the modal amplitude of mode shape $\Phi_{:i}$. An important property is that the modal matrix Φ is independent of time, and completely characterized by values at mesh vertices.

Substituting (2) into (1) and premultiplying by Φ^{T} yields

$$\mathsf{M_q}\ddot{\mathsf{q}} + \mathsf{C_q}\dot{\mathsf{q}} + \mathsf{K_q}\mathsf{q} = \mathsf{Q} \qquad (3)$$

in which

$$\begin{aligned} \mathsf{M_q} &= \Phi^{\mathsf{T}}\mathsf{M}\Phi = \mathrm{diag}(m_i) && (4)\\ \mathsf{K_q} &= \Phi^{\mathsf{T}}\mathsf{K}\Phi = \mathrm{diag}(k_i) && (5)\\ \mathsf{C_q} &= \Phi^{\mathsf{T}}\mathsf{C}\Phi && (6)\\ \mathsf{Q} &= \Phi^{\mathsf{T}}\mathsf{F} && (7) \end{aligned}$$

where all of $\mathsf{M_q}$ and $\mathsf{K_q}$ are diagonal matrices, but for general damping $\mathsf{C_q}$ is dense. If we make the common assumption of proportional (Rayleigh) damping

$$\mathsf{C} = \alpha\mathsf{M} + \beta\mathsf{K} \quad \Rightarrow \quad \mathsf{C_q} = \mathrm{diag}(\alpha m_i + \beta k_i)$$

then the system of ODEs are completely decoupled by the modal transformation. This allows the motions due to individual modes to be computed independently and combined by linear superposition.

The system of decoupled ordinary differential equations may be written as

$$\ddot{\mathsf{q}}_i + 2\xi_i\omega_i\dot{\mathsf{q}}_i + \omega_i^2\mathsf{q}_i = \frac{\mathsf{Q}_i}{m_i}, \qquad i = 1..n, \qquad (8)$$

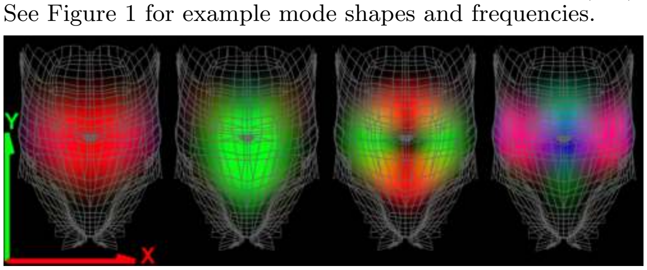
where the undamped *natural frequency of vibration* is

$$\omega_i = \sqrt{\frac{k_i}{m_i}} \quad \text{(in radians)} \qquad (9)$$

and the dimensionless *modal damping factor* is

$$\xi_i = \frac{c_i}{2m_i\omega_i} = \frac{1}{2}\left(\frac{\alpha}{\omega_i} + \beta\omega_i\right). \qquad (10)$$

We are interested in *underdamped systems* for which visible damped vibration occurs, and this corresponds to $\xi_i \in (0,1)$. See Figure 1 for example mode shapes and frequencies.



$\Phi_{:1}$ ($\omega_1 = 1.00$) $\Phi_{:2}$ ($\omega_2 \approx 1.12$) $\Phi_{:3}$ ($\omega_3 \approx 1.25$) $\Phi_{:4}$ ($\omega_4 \approx 1.44$)

Figure 1: *Dominant low frequency mode shapes* of the belly model represent bulk translation and rotation. RGB colors correspond to XYZ displacement magnitudes.

Finally, for a system starting from rest at $t=0$ the solution for the i^{th} mode due to forcing $\mathsf{Q}_i(t)$ is

$$\mathsf{q}_i(t) = \int_0^t e^{-\xi_i\omega_i(t-\tau)} \sin\omega_{di}(t-\tau)\frac{\mathsf{Q}_i(\tau)}{m_i\omega_{di}}d\tau \qquad (11)$$

where the observed *damped natural frequency* is

$$\omega_{di} = \omega_i\sqrt{1-\xi_i^2}. \qquad (12)$$

3 Exciting Modes with Rigid Motions

Our goal is to produce realistic modal deformations automatically from a conventional bone-based animation specification, for instance using motion capture data or rigid body dynamics simulation. Suppose the motion of a rigid body, the "bone," is specified as a homogenous transformation matrix $R(t) = \begin{pmatrix} \Theta & p \\ 0 & 1 \end{pmatrix}$, where Θ is a rotation matrix. We now describe how to compute the correct modal forcing function $\mathsf{Q}_i(t)$ for a deformable object, the "flesh," attached to a bone such as depicted in Figure 2. We describe how to deal with joints between bones in Sec. 4.2.

Figure 2: *Modeling of a thigh finite element model* using a skeleton and CSG operations.

The velocity of rigid body is represented by its linear velocity ν and angular velocity ω. We can therefore view velocity as a 6×1 *twist* or *spatial velocity*[1] vector $\psi = \left(\omega^T \nu^T\right)^T$.

The velocity, $\dot{r}_j$, of a material point at r_j is then given by

$$\dot{r}_j = [\omega]r_j + \nu = \left(\, -[r_j] \;\; I \,\right)\psi, \qquad (13)$$

where $[\omega]$ is the standard skew-symmetric matrix of the cross product $\omega\times$, and I is a 3-by-3 identity matrix. Using a simple Euler discretization, with constant time step size h, the acceleration of the material point at discrete time step k is

$$\ddot{r}_j^{(k)} \approx \frac{1}{h}(\dot{r}_j^{(k)} - \dot{r}_j^{(k-1)}) = \frac{1}{h}\left(\, -[r_j] \;\; I \,\right)(\psi^{(k)} - \psi^{(k-1)}). \quad (14)$$

Higher order discretizations are similar. Defining

$$\Gamma = \begin{pmatrix} -[r_1] & I \\ -[r_2] & I \\ & \vdots \\ -[r_p] & I \end{pmatrix}$$

we have the acceleration of points on the body as

$$\ddot{\mathsf{r}}^{(k)} \approx \frac{1}{h}\Gamma(\psi^{(k)} - \psi^{(k-1)}). \qquad (15)$$

When viewed relative to a coordinate frame attached to the bone, which is accelerating, the D'Alembert force is[2]

$$\mathsf{F}^{(k)} = \mathsf{M}\ddot{r}_j^{(k)} = \frac{1}{h}\mathsf{M}\Gamma(\psi^{(k)} - \psi^{(k-1)}). \qquad (16)$$

This is the forcing function for the vibration in (1). The primary modal forcing term Q_i/m_i in (11) is therefore

$$\begin{aligned} \mathsf{M}_q^{-1}\mathsf{Q}^{(k)} &= \frac{1}{h}\Phi^{-1}\Gamma(\psi^{(k)} - \psi^{(k-1)}), && (17)\\ &\stackrel{\text{def}}{=} \mathsf{H}(\psi^{(k)} - \psi^{(k-1)}). && (18) \end{aligned}$$

[1] Here we use the traditional kinematic terminology from screw theory. We refer the reader to any standard mathematical treatment on kinematics, such as [16], for more details

[2] Coriolis forces are negligible here and have been omitted.

We call $\mathsf{H} = (1/h)\Phi^{-1}\Gamma$ the *rigid motion transfer matrix*. It maps changes in spatial velocity to modal forces that lead to modal vibrations. It can be precomputed in advance of a simulation and stored. In practice, the forces may be filtered, e.g., scaled and clamped, to avoid extremely large excitations from abrupt motion changes or resonant forcing.

Finally, we need to perform the time-domain convolution of (11). This can be performed efficiently in discrete time using a small IIR digital filter [24, 26]:

$$\begin{aligned} \mathsf{q}_i^{(k)} &= 2\varepsilon_i \cos\theta_i \mathsf{q}_i^{(k-1)} - \varepsilon_i^2 \mathsf{q}_i^{(k-2)} \\ &+ \frac{2[\varepsilon_i \cos(\theta_i + \gamma_i) - \varepsilon_i^2 \cos(2\theta_i + \gamma_i)]}{3\omega_i \omega_{di}} \frac{\mathsf{Q}_i^{(k-1)}}{m_i} \end{aligned} \quad (19)$$

where $\varepsilon_i = \exp(-\xi_i \omega_i h)$, $\theta_i = \omega_{di} h$ and $\gamma_i = \arcsin \xi_i$.

4 Special Considerations

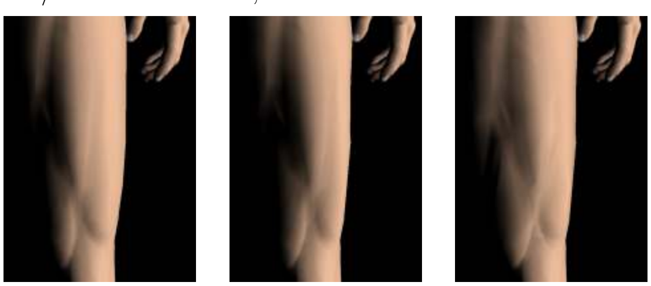
4.1 Normal Calculation

Unlike the displaced vertex positions which can be computed in parallel on a per-vertex basis, vertex normals are complicated by the requirement of neighbouring vertex information. Therefore DyRT objects include an approximate vertex normal correction obtained by linearizing the i^{th} vertex's deformed normal n'_i about the undeformed value $\hat{n}_i$,

$$n'_i = \hat{n}_i + \sum_m \mathsf{N}_{im} \mathsf{q}_m \quad (20)$$

where N_{im} is the i^{th} vertex's normal correction for mode m. Details are given in Appendix A.

While corrected normals can further increase visual realism (see Figure 3), the added cost of per-vertex memory for each mode's normal correction should be weighed against other vertex memory requirements. In practice, correcting normals only for particular modes, such as the dominant and/or torsional modes, is a fair trade-off.



Undeformed   Deformed without   Deformed with

Figure 3: *Normal correction benefits* are illustrated using the lowest torsional deformation mode of the thigh model: (Left) undeformed, (Middle) deformed without normal correction, and (Right) with normal correction computed.

4.2 Matrix Palette Skinning with DyRT

DyRT provides minimal complications for traditional hardware character animation. Using vertex program hardware for indexed matrix palette skinning vertex programs, as in [14] (see their "jester" example), *static display lists are used for each DyRT mapped object*. In our examples, per-vertex data exists not only for vertex position, normal, color, texture coords, and 4 matrix index/weight pairs, but also for DyRT values: one for each mode's displacement and any normal correction. Due to current vertex memory constraints, each vertex is vibrated by only one DyRT object (with multiple modes, as described in §5.1), but multiple layered (or blended) DyRTs could be used in the future, or at the cost of fewer modes or normal corrections per DyRT.

In the vertex program, modal deformations are performed before the vertex blending stage, and require at most $(2m + 2)$ extra instructions for m modes (using all normal corrections); in our "DyRT Man" example $m = 5$ so that only 12 instructions are added and the vertex program remains fast (see Appendix B).

5 Process Details

5.1 Precomputation

1. Acquire articulated character geometry.
2. For each deformable body part, e.g., thigh,
   - Use surface model to define a closed volume to be filled with elastic material.
   - Generate a volumetric finite element mesh, e.g., using a tetrahedral mesh generation package such as NETGEN [21].
   - Fix the finite element model's boundary vertices where you do not desire deformation, e.g., along bones and seams.
   - Define material properties such as stiffness, compressibility and density.
   - Compute and save the dominant modes' frequencies and volumetric mode shapes Φ using a modal analysis package, e.g., CalculiX [7] uses the excellent ARPACK eigenvalue solver [13].
   - *Build an m-mode DyRT object* consisting of
     - m modal model natural frequencies ω_i;
     - m modal shape functions $\Phi_{:1..m}$ interpolated onto the original character geometry;
     - m normal perturbation maps $\mathsf{N}_{:1..m}$ computed on the character geometry;
     - m IIR digital convolution filters from (19);
     - the m-by-6 transfer matrix H from (18).

5.2 Runtime Computations

For each animation time step, k, and each DyRT object:

1. Obtain the rigid bone transform and estimate the spatial velocity twist, $\psi^{(k-1)}$.
2. For each mode $i = 1..m$:
   - Compute the modal forcing term $\mathsf{Q}_i^{(k-1)}/m_i$ using the rigid motion transfer matrix from (18).
   - Perform the time domain IIR filter convolution of (19) to obtain $\mathsf{q}_i^{(k)}$.
3. Bind and enable appropriate DyRT vertex program and set vertex program constants: modal coefficients, $\mathsf{q}_i^{(k)}$, and current bone transforms (See Appendix B).
4. Call static display list for this body part.

6 Results

Our first example applies DyRT to a character animated using indexed matrix palette skinning vertex programs. The humanoid mesh used was exported from Curious Labs Poser and converted to 17,980 quadrilateral faces and 17,953 vertices. Following the described process, we constructed 3 DyRT objects: matching thigh models based on a 10,000 10-node tetrahedral element finite element model, and an abdominal model with 30,000 elements. Precomputation times were only a couple of minutes for each DyRT, and much larger models could be used. The final character was animated with House of Moves motion capture.

In our second example, we apply DyRT to secondary tissue in a laparoscopic surgical simulation. In this setting, DyRT helps increase scene realism while allowing the main CPU to focus on simulating more complex tissue models involved in user contact interactions.

Dynamic deformations are inherently difficult to portray in paper format, however examples in the accompanying video (see Figure 4) illustrate the subtle yet significant impact DyRT can have on scene realism. All examples run in real time, at approximately 60 FPS, on a PC with a GeForce3 graphics card; throughout the simulation *the runtime cost to the main CPU is negligible.*

Figure 4: *Examples from video:* (Left) A jumping motion that leads to significant thigh and belly vibrations; (Right) DyRT applied to tissue in a surgical simulation.

7 Summary and Conclusion

We have illustrated the process by which DyRT can be used to simulate geometrically complex, volumetric, physically-based, dynamic deformation models with *negligible main CPU costs* by exploiting commodity graphics hardware. Given our results, we believe that DyRT-based secondary animation is an efficient technique to increase the level of realism in modern real time applications.

Acknowledgements: We would like to thank the reviewers for their helpful suggestions, and Edward M. Lichten M.D. for the texture map image in our laparoscopic surgery example.

A Computation of Normal Correction

We first show how to approximate the face normal for a deformed triangle. Consider an undeformed triangle with vertices (p_0, p_1, p_2), a single mode m with amplitude q_m and shape function vertex displacements (u_0, u_1, u_2), so that the deformed triangle has coordinates $(p_0 + \mathsf{q}_m u_0, p_1 + \mathsf{q}_m u_1, p_2 + \mathsf{q}_m u_2)$. Let $U = (p_1 - p_0)$, $V = (p_2 - p_0)$, $\delta U = (u_1 - u_0)$, $\delta V = (u_2 - u_0)$, $U' = U + \delta U$ and $V' = V + \delta V$. For sufficiently small values of q_m the face normal is

$$n' = \frac{U' \times V'}{\|U' \times V'\|} \approx \frac{U \times V}{\|U \times V\|} + \mathsf{q}_m \left[\frac{\delta U \times V + U \times \delta V}{\|U \times V\|} \right] \quad (21)$$

where the quantity in square brackets is the flat-shaded normal correction. For smooth shading, normals can be averaged over vertex adjacent faces to obtain the i^{th} per-vertex normal correction N_{im} from (20). Alternate approaches using finite differences are also possible.

B Vertex Program for DyRT

```
# Load vertex p_i into R1 and add 5 modal corrections:
MOV R1, v[OPOS];                        # R1  = p_i
MAD R1, c[DyRT  ].xxxw, v[5], R1;       # R1 += q_1 Φ_i1
MAD R1, c[DyRT  ].yyyw, v[6], R1;       # R1 += q_2 Φ_i2
MAD R1, c[DyRT  ].zzzw, v[7], R1;       # R1 += q_3 Φ_i3
MAD R1, c[DyRT+1].xxxw, v[8], R1;       # R1 += q_4 Φ_i4
MAD R1, c[DyRT+1].yyyw, v[9], R1;       # R1 += q_5 Φ_i5

# Load normal n_i into R2 and add 5 modal corrections:
MOV R2, v[NRML];                        # R2  = n_i
MAD R2, c[DyRT  ].xxxw, v[10], R2;      # R2 += q_1 N_i1
MAD R2, c[DyRT  ].yyyw, v[11], R2;      # R2 += q_2 N_i2
MAD R2, c[DyRT  ].zzzw, v[12], R2;      # R2 += q_3 N_i3
MAD R2, c[DyRT+1].xxxw, v[13], R2;      # R2 += q_4 N_i4
MAD R2, c[DyRT+1].yyyw, v[14], R2;      # R2 += q_5 N_i5

# Bone-weighted Vertex Blending: ....
# Transform and Lighting: ....
```

References

[1] D. Baraff and A. Witkin. Dynamic Simulation of Non-penetrating Flexible Bodies. In *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, pages 303–308, 1992.

[2] D. Baraff and A. Witkin. Large Steps in Cloth Simulation. In *SIGGRAPH 98 Conference Proceedings*, pages 43–54, 1998.

[3] A. Barr. Global and Local Deformations of Solid Primitives. In *Computer Graphics (SIGGRAPH 84 Conference Proceedings)*, volume 18, pages 21–30, 1984.

[4] C. Basdogan. Real-time Simulation of Dynamically Deformable Finite Element Models Using Modal Analysis and Spectral Lanczos Decomposition Methods. In *Medicine Meets Virtual Reality (MMVR'2001)*, pages 46–52, 2001.

[5] J.E. Chadwick, D.R. Haumann, and R.E. Parent. Layered Construction of Deformable Animated Characters. In *Computer Graphics (SIGGRAPH 89 Conference Proceedings)*, volume 23, pages 243–252, 1989.

[6] G. Debunne, M. Desbrun, A. Barr, and M.-P. Cani. Dynamic real-time deformations using space and time adaptive sampling. In *SIGGRAPH 01 Conference Proceedings*, pages 31–36, 2001.

[7] G. Dhondt and K. Wittig. CalculiX: A Free Software Three-Dimensional Structural Finite Element Program.

[8] M. Friedmann and A. Pentland. Distributed physical simulation. In *Third Eurographics Workshop on Animation and Simulation*, pages 1–17, 1992.

[9] J. Gourret, N. Magnenat-Thalmann, and D. Thalmann. Simulation of Object and Human Skin Deformations in a Grasping Task. In *Computer Graphics (SIGGRAPH 89 Conference Proceedings)*, volume 23, pages 21–29, 1989.

[10] D.L. James. *Multiresolution Green's Function Methods for Interactive Simulation of Large-scale Elastostatic Objects and Other Physical Systems in Equilibrium.* PhD thesis, Institute of Applied Mathematics, University of British Columbia, Vancouver, British Columbia, Canada, 2001.

[11] D.L. James and D.K. Pai. ArtDefo: Accurate Real Time Deformable Objects. In *SIGGRAPH 99 Conference Proceedings*, pages 65–72, 1999.

[12] Y. Lee, D. Terzopoulos, and K. Walters. Realistic Modeling for Facial Animation. In *SIGGRAPH 95 Conference Proceedings*, pages 55–62, 1995.

[13] R. Lehoucq, D. Sorensen, and C. Yang. ARPACK Users' Guide: Solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods. Technical report, Comp. and Applied Mathematics, Rice Univ., 1997.

[14] E. Lindholm, M.J.Kilgard, and H. Moreton. A User-Programmable Vertex Engine. In *SIGGRAPH 2001 Conference Proceedings*, pages 149–158, 2001.

[15] D. Metaxas and D. Terzopoulos. Dynamic Deformation of Solid Primitives with Constraints. In *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, volume 26, pages 309–312, 1992.

[16] R.M. Murray, Z. Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Inc., 1994.

[17] J. O'Brien, P. Cook, and G. Essl. Synthesizing Sounds from Physically Based Motion. In *SIGGRAPH 01 Conference Proceedings*, pages 529–536, 2001.

[18] J.F. O'Brien and J.K. Hodgins. Graphical modeling and animation of brittle fracture. In *SIGGRAPH 99 Conference Proceedings*, pages 111–120, 1999.

[19] A. Pentland and J. Williams. Good Vibrations: Modal Dynamics for Graphics and Animation. In *Computer Graphics (SIGGRAPH 89 Conference Proceedings)*, volume 23, pages 215–222, 1989.

[20] G. Picinbono, H. Delingette, and N. Ayache. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *ICRA2001: IEEE International Conference on Robotics and Automation*, Seoul Korea, May 2001.

[21] J. Schoberl. NETGEN - An advancing front 2D/3D-mesh generator based on abstract rules. *Comput.Visual.Sci*, 1:41–52, 1997.

[22] A.A. Shabana. *Theory of Vibration, Volume II: Discrete and Continuous Systems.* Springer–Verlag, New York, NY, first edition, 1990.

[23] J. Stam. Stochastic Dynamics: Simulating the Effects of Turbulence on Flexible Structures. *Computer Graphics Forum*, 16(3), 1997.

[24] K. Steiglitz. *A Digital Signal Processing Primer with Applications to Digital Audio and Computer Music.* Addison-Wesley, New York, 1996.

[25] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4:306–331, 1988.

[26] K. van den Doel, P.G. Kry, and D.K. Pai. FoleyAutomatic: Physically-based Sound Effects for Interactive Simulations and Animations. In *SIGGRAPH 01 Conference Proceedings*, 2001.

[27] J. Wilhelms and A.V. Gelder. Anatomically Based Modeling. In *SIGGRAPH 97 Conference Proceedings*, pages 173–180, 1997.

[28] A. Witkin and W. Welch. Fast Animation and Control of Nonrigid Structures. In *Computer Graphics (SIGGRAPH 90 Conference Proceedings)*, pages 243–252, 1990.

[29] O. C. Zienkiewicz. *The Finite Element Method.* McGraw-Hill Book Company (UK) Limited, Maidenhead, Berkshire, England, 1977.