Synthesizing Sounds from Rigid-Body Simulations

James F. O'Brien Chen Shen Christine M. Gatchalian

EECS, Computer Science Division
University of California, Berkeley

Abstract

This paper describes a real-time technique for generating realistic and compelling sounds that correspond to the motions of rigid objects. By numerically precomputing the shape and frequencies of an object's deformation modes, audio can be synthesized interactively directly from the force data generated by a standard rigid-body simulation. Using sparse-matrix eigen-decomposition methods, the deformation modes can be computed efficiently even for large meshes. This approach allows us to accurately model the sounds generated by arbitrarily shaped objects based only on a geometric description of the objects and a handful of material parameters. We validate our method by comparing results from a simulated set of wind chimes to audio measurements taken from a real set.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation; H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing—Signal analysis, synthesis, and processing

Keywords: Sound modeling, physically based modeling, simulation, surface vibrations, dynamics, animation techniques, finite element method, modal synthesis, modal analysis.

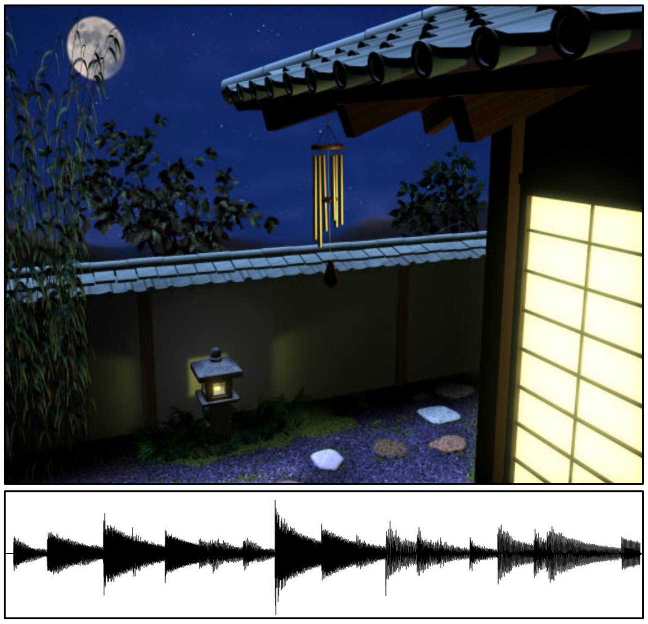

Figure 1: A synthetic environment containing a set of simulated wind chimes. Both the motion of the chimes and the corresponding audio can be computed at interactive speeds.

1 Introduction

One of the central goals for the field of computer graphics is the compelling portrayal of realistic synthetic environments. However, generating convincing animations of scenes such as that shown in figure 1 requires depicting not only the visual aspects of the scene, but its audio components as well. While constructing a soundtrack by hand often provides a feasible option for animations that are generated off line, interactive applications increasingly rely on physically based simulation techniques to generate animated motions in real-time and these applications require methods for generating the corresponding audio in real-time as well.

One class of simulation method that has found widespread use in real-time applications is rigid-body simulations. Because rigid bodies are made up of incompliant materials, they experience only small-amplitude deformations during interactions with their environment. Explicitly discarding these small deformations allows rigid-body simulators to model a system's remaining degrees of freedom efficiently. However, although visually insignificant, it is the vibration of these small-amplitude deformations that generates the sounds heard from these objects.

This paper describes a real-time technique for generating realistic and compelling sounds that correspond to the motions generated by rigid-body simulation methods. Precomputing the shape and frequencies of an object's deformation modes allows that object's vibrational response to contact forces to be efficiently computed at runtime. The vibrational response is then used directly to compute the corresponding audio. Our technique computes an object's deformation modes numerically by performing an eigen-decomposition of the system matrices from a finite element model of the object. This approach allows us to accurately model the sounds generated by arbitrarily shaped objects based on a geometric description of the object and a handful of material parameters. The diagram in figure 2 provides an overview of this process.

job@cs.berkeley.edu, csh@cs.berkeley.edu, tine@cs.berkeley.edu

2 Background

The technique presented in this paper is closely related to previous methods developed by van den Doel, Kry, and Pai. The concept of using the vibrational modes of an object for generating sound was originally introduced to the graphics community in [van den Doel

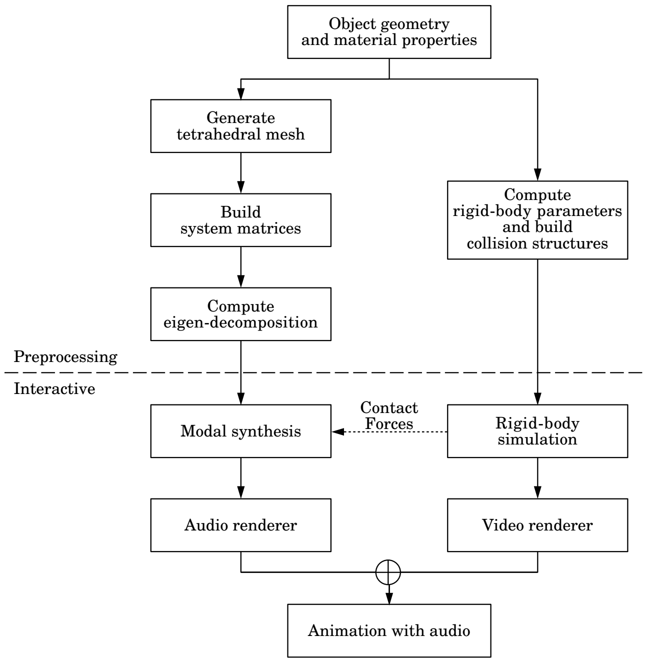

Figure 2: This diagram illustrates both the preprocessing steps that are used to construct the audio/visual model for an object, and the processes that subsequently generate sound and motion from this description at interactive speeds.

and Pai, 1996] and [van den Doel and Pai, 1998]. They showed that the analytically computed vibrational modes of simply shaped objects, such as square plates or cylindrical rods, could be used to generate sound maps over the object surfaces. They constructed a system that computed realistic contact sounds when an interactive user indicated a point on the surfaces of the modeled objects. Because the maps were generated from the mode shapes, the system correctly captured the variations that arise when objects are struck at different locations. More recently, [van den Doel et al., 2001] described a method that uses recorded data to construct sound maps over the surface of an object. In addition to allowing a user to interactively generate sounds by tapping the object surfaces, they used contact forces from a real-time rigid-body simulation to excite the sampled modes.

Our work builds on the ideas of these two previous methods by extending the range of objects that can be modeled to include ones that are neither simple shapes nor available to be measured. The analytically computed modes used in [van den Doel and Pai, 1996] and [van den Doel and Pai, 1998] are the continuous equivalent of the numerically computed discretized modes described in Section 3.1 of this paper. Numerical computation allows determining the modes for essentially arbitrary shapes as opposed to a few simple shapes, and it makes fewer assumptions about the underling differential equations. Our method for driving the sound generation from a rigid-body simulation is essentially identical to the method used in [van den Doel et al., 2001], but we have found that useful results can be generated using “off-the-shelf” rigid-body simulators and that a special contact method is not necessarily required unless one wishes to produce rubbing or scraping sounds.

Another related method for generating sound has been described in [O’Brien et al., 2001]. Their approach uses a nonlinear finite element method to explicitly model the response of an object to external forces. Audio is generated by analyzing the computed surface behavior and then applying a set of filters to the computed motion for extracting frequency components that fall within the audible range. Unlike the method detailed in this paper and the previously described methods of van den Doel and his colleagues, the use of a nonlinear finite element method allows them to model sounds that arise from nonlinear behaviors such as buckling. The main limitation of their method is that it requires large amounts of computation. In contrast, our method can accurately model only sounds produced by linear phenomena, but it can compute these sounds in real-time.

In addition to the above physically motivated work on sound generation, other prior work in the graphics community has focused on sound propagation and heuristic approaches to sound generation. Producing synchronized soundtracks for animations was addressed in [Takala and Hahn, 1992] and [Hahn et al., 1995]. For modeling tearing cloth, [Terzopoulos and Fleischer, 1988] generated soundtracks by playing a pre-recorded sound whenever a connection in a spring mesh failed. Work described in [Savioja et al., 1997] focused on creating virtual musical performances in virtual spaces using physically derived models of musical instruments and acoustic ray-tracing for spatialization of the sound sources. Other researchers have developed methods for correctly modeling reflections and transmissions within the sonic environment [Funkhouser et al., 1998; Funkhouser et al., 1999; Min and Funkhouser, 2000; Tsingos et al., 2001].

The method described in this paper is also related to previous work in the graphics community on modeling deformable objects. The idea of decoupling an object’s rigid-body behavior from it’s elastic deformation was proposed in [Terzopoulos and Fleischer, 1988] as an efficient method for modeling deformable objects. This idea was extended in [Pentland and Williams, 1989] by using modal analysis for modeling deformable objects, although instead of actually using the object’s vibrational modes they approximated them with arbitrary linear and quadratic deformation fields.

Outside the field of computer graphics, an extensive amount of research on sound modeling has been conducted in the digital sound and music communities. There the focus has been primarily on accurately modeling the sounds generated by musical instruments, including the fine subtleties that distinguish high-quality instruments. A comprehensive review of the work that has been done in those areas can be found in [Cook, 2002].

3 Methods

The mechanical dynamics of a solid physical object can be decomposed into two components: idealized rigid-body motions and elastic deformations. An object is referred to as being rigid, or incompliant, if its response to typical interactions includes only negligible deformations. For example, when a person taps the side of a drinking glass it flexes slightly but the amplitude of this deformation is small enough to be unobservable by sight. However, this small deformation may be observable by hearing. In particular, if the elastic properties of the glass are such that the small deformation induced by the tap results in vibrations at frequencies between approximately 20 and $20,000$ Hz, then the small pressure fluctuations caused by the oscillating deformation will be heard as sound. For further information on the physical process of sound generation we refer the reader to [Kinsler et al., 2000].

3.1 Modal Analysis

Our method for modeling the sounds generated by rigid objects makes use of a well studied technique known as modal analysis. This section presents a brief overview of modal analysis and provides the framework for describing our methods. We refer the reader to [Cook et al., 1989] for additional information on modal analysis.

A physical system that has been discretized using a finite element, finite differencing, or other similar method can be expressed in the following general form:

$$\boldsymbol{\mathcal{K}}(\boldsymbol{d}) + \boldsymbol{\mathcal{C}}(\boldsymbol{d}, \dot{\boldsymbol{d}}) + \boldsymbol{\mathcal{M}}(\ddot{\boldsymbol{d}}) = \boldsymbol{f} \tag{1}$$

where $\boldsymbol{d}$ is the vector of node displacements, an overdot indicates a derivative with respect to time, $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{\mathcal{C}}$ are nonlinear functions that respectively determine the internal forces due to node displacements and node velocities, $\boldsymbol{\mathcal{M}}$ maps node accelerations to node momenta, and $\boldsymbol{f}$ represents any other (*e.g.* external) forces. Typically, the forces determined by $\boldsymbol{\mathcal{K}}$ are internal elastic forces and $\boldsymbol{\mathcal{C}}$ determines damping forces.

In general, equation (1) is nonlinear, however if we assume that the displacements are small then we may linearize about the system's rest configuration giving:

$$\boldsymbol{K}\boldsymbol{d} + \boldsymbol{C}\dot{\boldsymbol{d}} + \boldsymbol{M}\ddot{\boldsymbol{d}} = \boldsymbol{f} \tag{2}$$

where $\boldsymbol{K}$, $\boldsymbol{C}$, and $\boldsymbol{M}$ are respectively known as the system's stiffness, damping, and mass matrices. For the physical systems corresponding to solid objects, all three matrices are real and symmetric. Both $\boldsymbol{K}$ and $\boldsymbol{C}$ are positive semi-definite, and $\boldsymbol{M}$ is positive definite. Linearizing in this fashion is consistent with our goal of modeling the small-amplitude, high-frequency vibrations in solid objects that produce sound. Unfortunately, the linearized system cannot model the rotational components of rigid-body motion. We will put this issue aside for now, but later we will return to it and show how the rigid-body modes can be decoupled from all other modes.

Once we have the linearized system, the next step in the modal analysis is to perform a series of manipulations that will diagonalize equation (2). To facilitate this process, we will first assume that $\boldsymbol{C} = \alpha_1\boldsymbol{K} + \alpha_2\boldsymbol{M}$ for some α_1 and α_2. Expressing the damping matrix as a linear combination of the stiffness and mass matrices is known as Raleigh damping. Although this assumption simplifies diagonalization while still producing good results, it is not strictly necessary. A more general assumption, known as proportional damping, that expresses the damping matrix as a linear combination of powers of the stiffness and mass matrices would also be diagonalized by the process described below but the equations would be more cumbersome. Additionally, even if for some reason $\boldsymbol{C}$ must be arbitrary, then other, slightly more complicated, methods are available for decoupling equation (2) [Anderson et al., 1999; Bai et al., 2000].

Replacing $\boldsymbol{C}$ with $\alpha_1\boldsymbol{K} + \alpha_2\boldsymbol{M}$ gives:

$$\boldsymbol{K}(\boldsymbol{d} + \alpha_1\dot{\boldsymbol{d}}) + \boldsymbol{M}(\alpha_2\dot{\boldsymbol{d}} + \ddot{\boldsymbol{d}}) = \boldsymbol{f}\ . \tag{3}$$

Since $\boldsymbol{M}$ is symmetric and positive definite, it may be decomposed using a Cholesky factorization so that $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}$. If we introduce another variable, $\boldsymbol{y} = \boldsymbol{L}^{\mathsf{T}}\boldsymbol{d}$, and then rewrite equation (3) in terms of $\boldsymbol{y}$ after pre-multiplying by $\boldsymbol{L}^{-1}$ we then have:

$$\boldsymbol{L}^{-1}\boldsymbol{K}\boldsymbol{L}^{-\mathsf{T}}(\boldsymbol{y} + \alpha_1\dot{\boldsymbol{y}}) + (\alpha_2\dot{\boldsymbol{y}} + \ddot{\boldsymbol{y}}) = \boldsymbol{L}^{-1}\boldsymbol{f}\ . \tag{4}$$

The real and symmetric matrix $\boldsymbol{L}^{-1}\boldsymbol{K}\boldsymbol{L}^{-\mathsf{T}}$ can be decomposed into $\boldsymbol{L}^{-1}\boldsymbol{K}\boldsymbol{L}^{-\mathsf{T}} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathsf{T}}$ where $\boldsymbol{V}$ is the orthogonal matrix whose columns are the eigenvectors of $\boldsymbol{L}^{-1}\boldsymbol{K}\boldsymbol{L}^{-\mathsf{T}}$ and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues. Introducing another variable, $\boldsymbol{z} = \boldsymbol{V}^{\mathsf{T}}\boldsymbol{y}$, and pre-multiplying by $\boldsymbol{V}^{\mathsf{T}}$ transforms equation (4) into:

$$\boldsymbol{\Lambda}(\boldsymbol{z} + \alpha_1\dot{\boldsymbol{z}}) + (\alpha_2\dot{\boldsymbol{z}} + \ddot{\boldsymbol{z}}) = \boldsymbol{V}^{\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{f} \tag{5}$$

which can be rearranged to give:

$$\boldsymbol{\Lambda}\boldsymbol{z} + (\alpha_1\boldsymbol{\Lambda} + \alpha_2\boldsymbol{I})\dot{\boldsymbol{z}} + \ddot{\boldsymbol{z}} = \boldsymbol{g} \tag{6}$$

where $\boldsymbol{g} = \boldsymbol{V}^{\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{f}$.

At this point the original linear system of equation (3) has been diagonalized into a set of decoupled oscillators. The i'th row of equation (6) is the scalar second-order differential equation:

$$\lambda_i z_i + (\alpha_1\lambda_i + \alpha_2)\dot{z}_i + \ddot{z}_i = g_i \tag{7}$$

where λ_i is the i'th entry of the diagonal matrix $\boldsymbol{\Lambda}$. Equation (7) may be solved by numerical integration or it may be solved more efficiently using the analytic solution:

$$z_i = c_1 e^{t\omega_i^+} + c_2 e^{t\omega_i^-} \tag{8}$$

where c_1 and c_2 are arbitrary (complex) constants, and ω_i is the complex frequency given by

$$\omega_i^{\pm} = \frac{-(\alpha_1\lambda_i + \alpha_2) \pm \sqrt{(\alpha_1\lambda_i + \alpha_2)^2 - 4\lambda_i}}{2}\ . \tag{9}$$

The absolute value of the imaginary part of ω_i is the frequency (in radians/second, not Hertz) of the mode, and the real part is the mode's decay rate.

The decoupled system of equation (6) is *not an approximation* of the original linear system in equation (3), it is *exactly* the same as the original linear system. Of course the linear system was an approximation of the original nonlinear one, but any problem that could be solved using equation (3) could also be solved with equation (6).

The columns of $\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{V}$ are the vibrational modes of the object being modeled. (See figure 3.) Each mode has the property that a displacement or velocity over the object that is a scalar multiple of the mode will produce an acceleration that is also a scalar multiple of the mode. This property means that the modes do not interact with each other, which is why decoupling the system into a set of independent oscillators was possible. The eigenvalue for each mode is the ratio of the mode's elastic stiffness to the mode's mass, and it is the square of the mode's natural frequency (in radians per second). In general the eigenvalues will be nonzero, but for each free body in the system there will be six zero eigenvalues that correspond to the body's six rigid-body modes. The rigid-body eigenvalues are zero because a rigid-body displacement will not generate any elastic forces.

3.2 Rigid Body Simulation

As discussed previously, the rigid-body modes for an object do not interact with the object's deformation modes provided the amount of elastic deformation experienced by the object is small.[1] Additionally, small-amplitude elastic deformations will not significantly effect the rigid-body collisions between objects. These observations allow us to model the rigid-body behavior of the objects in almost the same way as if we were not interested in generating audio. The only change that must be made to the rigid-body simulation is that information about contact forces must be gathered and exported to another process that will generate the audio. Of course, hearing the results of the rigid-body simulation, in addition to seeing them, may reveal previously unnoticed inadequacies of the simulator, but we have not found this to be a problem with the simulation engines we have worked with.

We have implemented our system using two existing rigid-body simulation engines that were not originally designed for generating audio. Our choice of engines was motivated by what systems were readily available and how well they were able to model the scenarios we wished to test. The first is a commercial software package, Vortex, sold by Critical Mass Labs. The second system we are using had been previously written by Okan Arikan, a graduate student

[1]Actually, the requirement was that all displacements be small, including displacements corresponding to the rigid-body modes. The translation modes are inherently linear so they cannot interact with the elastic modes regardless of their magnitude, but for a rapidly rotating body there will be some coupling between the rotation modes and the elastic ones. Unless the object is rotating very rapidly or experiencing large angular accelerations, the coupling between rotation and elastic modes with frequencies in the audible range will be negligible, so we ignore this interaction.

not involved in this project. No special changes were made to either package other then instrumenting them to allow reporting collision forces.

3.3 Deformation Model

Once the task of modeling the rigid-body modes has been delegated to a rigid-body simulator, the remaining elastic deformation modes can be used for generating audio. Because we are interested in modeling sounds from incompliant objects, we can use the modal decomposition methods described in Section 3.1 to compute their behavior efficiently. However before we can perform a modal decomposition, we must first select a deformable modeling method that can be used to generate the $\boldsymbol{K}$, $\boldsymbol{C}$, and $\boldsymbol{M}$ matrices.

The method we are using for modeling deformable behavior is the tetrahedral finite element method described by [O'Brien and Hodgins, 1999] for modeling fracture propagation, and subsequently used in [O'Brien et al., 2001] for modeling nonlinear audio generation. As discussed by O'Brien, Cook, and Essl, a variety of methods could be used, including spring/mass systems or finite differences methods. We selected this finite element method because their previous results show that it is accurate enough for generating compelling audio.

Computing the global stiffness and mass matrices proceeds by first computing individual 12×12 stiffness and mass matrices for each element and then assembling the results to form the global matrices. From [O'Brien and Hodgins, 1999] the nonlinear node forces are given by:

$$f_{[i]a} = -\frac{\mathrm{vol}}{2} \sum_{j=1}^{4} p_{[j]a} \sum_{k=1}^{3} \sum_{l=1}^{3} \beta_{jl} \beta_{ik} \sigma_{kl} \tag{10}$$

where $f_{[i]a}$ is the a'th component of the force exerted on the i'th node of the element, vol is the volume of the element, p are the node positions, β is the element basis matrix, and σ is the stress tensor within the element. Details for computing β and σ appear in [O'Brien and Hodgins, 1999].

The element stiffness matrix, $\boldsymbol{k}$, is computed by taking the partials of $\boldsymbol{f}$ and evaluating them at zero displacement:

$$\begin{aligned} k_{[ij]ab} &= \left. \frac{\partial f_{[i]a}}{\partial p_{[j]b}} \right|_{\boldsymbol{p}=\boldsymbol{p}_{\mathrm{rest}}} & (11) \\ &= -\frac{\mathrm{vol}}{2} \left(\lambda \beta_{ia} \beta_{jb} + \mu \beta_{ib} \beta_{ja} + \mu \sum_{k=1}^{3} \beta_{ik} \beta_{jk} \delta_{ab} \right) & (12) \end{aligned}$$

where δ is the Kronecker delta, and λ and μ are the material's Lamé constants.[2] This is the exactly the same matrix that would have resulted if Cauchy's infinitesimal strain had been used in place of Green's strain, however with Cauchy's strain the partials would be constant with respect to node position so that it would not matter where they were evaluated.

The element mass matrix, $\boldsymbol{m}$ is computed by taking the second partials of the kinetic energy within the element with respect to the node velocities, which turns out to be constant with respect to node position and velocity:

$$\begin{aligned} \boldsymbol{m}_{[ij]ab} &= \frac{\partial^2 \kappa}{\partial \dot{p}_{[i]a} \partial \dot{p}_{[j]b}} & (13) \\ &= \frac{\rho \, \mathrm{vol}}{20} (1 + \delta_{ij}) \delta_{ab} & (14) \end{aligned}$$

[2]Unfortunately, the symbol λ is conventionally used both to indicate one of the system eigenvalues and the first Lamé constant. In this paper it should be clear from context (and the presence or absence of a subscript) what the symbol is referring to.

where κ is the kinetic energy within the element, an overdot represents a derivative with respect to time (*i.e.* $\dot{p}$ are node velocities), and ρ is the material's density.

The global stiffness and mass matrices, $\boldsymbol{K}$ and $\boldsymbol{M}$, are built by assembling the element matrices. Assuming that we are working with three-dimensional objects, each of the global matrices will be $3N \times 3N$ where N is the number of nodes in the finite element mesh. Each entry in each of the 12×12 element matrices is accumulated into the corresponding entry of the global matrix.

Since each node in a tetrahedral mesh will share an element with only a small number of the other nodes, the global matrices will be very sparse. This sparseness means that an eigen decomposition of $\boldsymbol{K}$ can be performed efficiently using sparse matrix algorithms. Unfortunately, the Cholesky decomposition tends to generate a dense $\boldsymbol{L}$ matrix even when $\boldsymbol{M}$ is originally sparse, and as a result computing $\boldsymbol{L}^{-1}$ may be costly and $\boldsymbol{L}^{-1}\boldsymbol{K}\boldsymbol{L}^{-\mathsf{T}}$ will be densified.

Dense matrix algorithms can be used for systems up to approximately 1000 nodes, but beyond that we suggest using an alternate mass matrix that does not generate a dense Cholesky decomposition. The alternate mass matrix, known as a lumped mass matrix, simply shifts the sum of each row onto the diagonal:

$$\boldsymbol{m}^{\mathrm{lumped}}_{[ij]ab} = \frac{\rho \, \mathrm{vol}}{4} \delta_{ij} \delta_{ab} \,. \tag{15}$$

Because the element mass matrices are diagonal, the global mass matrix will be as well, and its Cholesky decomposition will also be diagonal: $\boldsymbol{L}$ will be a diagonal matrix whose entries are simply the square root of the entries of the lumped $\boldsymbol{M}$. For small systems generated by coarse meshes, the errors introduced by mass lumping may be significant. However, as the mesh gets finer the errors introduced by lumping quickly become insignificant [Cook et al., 1989]. Luckily, the large systems corresponding to fine meshes are precisely the ones that require the sparse solvers facilitated by mass lumping. Our implementation includes both dense and sparse decomposition routines and we use whichever is appropriate to the size of a particular system. For dense decompositions, we use the routines from LAPACK [Anderson et al., 1999], and for sparse decompositions we use the TRLan package [Wu and Simon, 1999]. The method used for each of our examples, along with computation times and the number of nodes, is listed in table 1.

The use of Raleigh damping was another simplification that we made to facilitate decoupling equation (2). In [O'Brien and Hodgins, 1999] they used a nonlinear stiffness-proportional damping term based on the strain rate with parameters ϕ and ψ. Raleigh damping is equivalent to a linearization of this damping term with the additional constraint that $\frac{\lambda}{\phi} = \frac{\mu}{\psi}$, and the Raleigh parameter α_1 should be set to this ratio to generate equivalent results. O'Brien and Hodgins did not discuss a mass proportional damping term, but setting α_2 to a non-zero value would be equivalent to including a $(-\alpha_2 \dot{d}_i m_i)$ damping force on each node.

Even with sparse matrix methods, computing a system decomposition still requires a significant amount of time, so it is worth noting that certain changes may be made without recomputing the decomposition. The damping parameters, α_1 and α_2, have no effect on the decomposition, so the only work involved when changing them is re-evaluating equation (9). Changing the material's density does not change the mode shapes, it only scales the eigenvalues by the inverse of the scale factor applied to the density. Similarly, scaling the Lamé constants both by the same scale factor (*i.e.* so that the ratio between λ and μ is preserved) only scales the eigenvalues by the same ratio. Changing the ratio between the Lamé constants, changing the shape of the object, or modifying the mesh all require recomputing the decomposition.

3.4 Sound Generation

Once all of the computational machinery described above is available, the actual process of computing audio matching the motion

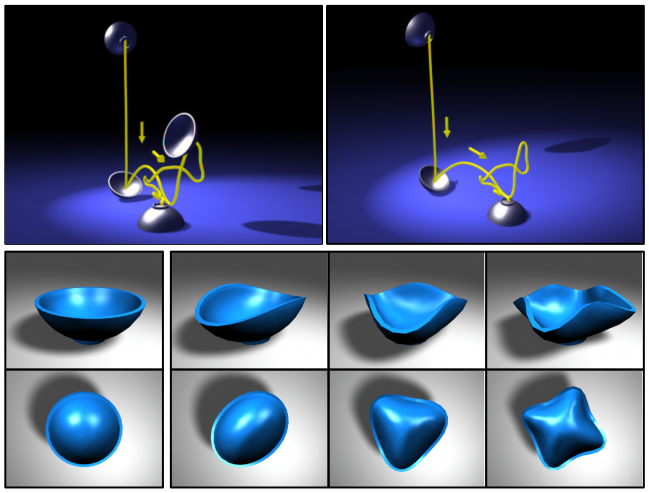

Figure 3: The top shows a multi-exposure image from an animation of a bowl falling onto a hard surface with the path of the bowl's center traced by a yellow curve. Only the bowl is sounding. The two bottom rows show a side and top view of the bowl along with three of the bowl's first vibrational modes. (The modes selected for the illustration are the first three non-rigid ones with distinct eigenvalues that are excited by a transverse impulse to the bowl's rim.)

from a rigid-body simulation is both straightforward to implement and computationally efficient:

1. A rigid-body simulation is set up for the desired scenario.
2. For each object in the simulation, the system matrices are assembled and decomposed into their vibrational modes (*i.e.* the columns of $\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{V}$).
3. For each object in the simulation, only the columns of $\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{V}$ corresponding to $|\mathrm{Im}(\omega_i)|$ in the range $3.18 \ldots 3{,}180\,\mathrm{Rad/s}$ (20...20,000 Hz) are retained, the rest are discarded (or if the sparse method is used, never computed).
4. As the rigid body simulation runs, collision forces are projected onto the retained modes. The response of each mode is modeled using equation (8).
5. Each mode response is scaled according to how it moves the objects surface and the scaled responses are then summed together.
6. Finally, the result is output to the computer's audio device.

In practice, not all of the modes in the audible range need to be retained. As discussed in [van den Doel et al., 2001], high-quality results can easily be obtained using only the first 800 or fewer modes.

A mode's response to a projected impulse is given by equation (8) with

$$c_1 = \frac{2\Delta t g_i}{\omega_i^+ - \omega_i^-} \qquad (16)$$

$$c_2 = \frac{2\Delta t g_i}{\omega_i^- - \omega_i^+} \qquad (17)$$

where Δt is the interval over which the projected force is applied, and t is time relative to when the impulse was applied. Substituting these values of c_1 and c_2 into equation (8), recalling that only modes with $|\mathrm{Im}(\omega_i)|$ in the range $3.18 \ldots 3{,}180\,\mathrm{Rad/s}$ are used, and then simplifying yields

$$z_i = \frac{2\Delta t g_i}{|\mathrm{Im}(\omega_i)|} e^{t\mathrm{Re}(\omega_i)} \sin(t|\mathrm{Im}(\omega_i)|) \ . \qquad (18)$$

Evaluating equation (18) for every audio sample is inefficient. By noting that $e^{\omega(t+s)} = e^{\omega t}e^{\omega s}$, the value of the oscillator at one audio sample can be computed from the previous value using only a single complex multiply. Additionally, as a mode is excited at subsequent times by different contact forces, the additional excitations can be modeled by simply adding the new value to the oscillator's current value. Because the cost of modeling additional impulses is essentially zero, the forces from the rigid-body simulation may be convolved with a Gaussian kernel to model the effect of soft collisions, or with a noise function to model small-scale roughness that is below the resolution of the rigid-body simulator [van den Doel et al., 2001]. Our results were generated using the former.

A method for modeling the coupling between vibrations in an object and vibrations in the surrounding air is described in [O'Brien et al., 2001]. Unfortunately, their method is too slow for real-time use. We compute an approximate coupling coefficient for each mode by summing the amount of normal displacement generated by that mode over the surface of the object multiplied by the mode's frequency. The coupling coefficient for each mode multiplies the result computed by that mode's oscillator and the sum of the scaled oscillators is the final sound generated by the system. A result of this simplification is all objects are treated as omni-directional sources.

4 Results and Discussion

We have built a system that implements the methods described above and used it to generate a number of demonstrative examples. Table 1 lists the parameters that were used in each of the examples, and the video tape accompanying this paper contains animations that exhibit the sounds and motions produced.

To test how well the computed results match real objects, we generated the wind chimes shown in figure 1. These chimes were modeled based on measurements from a real set of chimes. Each tube is a hollow cylinder $1.25\,\mathrm{cm}$ in radius with a nominal wall thickness of $1\,\mathrm{mm}$. The measured lengths of the chimes are listed in table 2. We computed the modal decomposition for each chime using reference parameters for aluminum. The resulting base frequencies matched measured ones to within 2% error. However, the real chimes were slightly out of tune, so we tuned the simulated set by adjusting the tube lengths so that they were within $\pm 1\,\mathrm{Hz}$ of the correct (D scale) tuning.

Figure 3 shows a bowl model that was used for two of the examples. The modal decomposition of the bowl was computed once with material parameters for aluminum and again with material parameters for wood (oak). Two animations were created, both with the same rigid-body motion but with the two sound tracks generated from the two different modal decompositions. The resulting audio (refer to video tape) captures the general characteristics of both materials as well as details such as the sound produced as the bowl rolls on its edge. Figure 3 also illustrates the mode-shapes for three of the bowl's vibrational modes by showing the results of applying the mode as a displacement over the bowl's original shape.

An example generated using a more complex model consists of bunny figurines falling through a chute. (See figure 4.) Both the bunny and the shelves in the chute generate sounds when struck. The shelves are made of plastic, metal, and wood. The bunny is ceramic. The tetrahedral bunny model was generated by meshing the region between the surface of the Stanford Bunny model and an interior offset surface to create a hollow figure with finite thickness walls, as shown on the right side of figure 4. The right side of

Example	Figure	λ (Pa)	μ (Pa)	α_1	α_2	ρ (Kg/m^3)	Base Freq. (Hz)	Decay	Num. Nodes	Method	Precompute
Chime(D3)	1	4.98×10^{10}	2.57×10^{10}	1×10^{-7}	0	2700	587.4	0.6	18796	Sparse	2h 24min
Bowl #1	3	4.98×10^{9}	2.57×10^{9}	1×10^{-7}	30	2700	551.3	15.6	387	Dense	4min 12sec
Bowl #2	–	5.00×10^{8}	1.00×10^{8}	8×10^{-6}	50	750	216.7	22.4	387	Dense	4min 12sec
Bunny (Ceramic)	4	3.99×10^{9}	2.05×10^{9}	1×10^{-6}	10	2700	855.9	19.5	37114	Sparse	4h 40min
Plastic Shelf	4	2.49×10^{10}	1.28×10^{10}	1×10^{-6}	50	2700	488.9	29.7	361	Sparse	30sec
Aluminum Shelf	4	4.98×10^{10}	2.56×10^{10}	1×10^{-7}	0	2700	691.5	0.9	361	Sparse	30sec
Wood Shelf	4	5.00×10^{8}	1.00×10^{8}	8×10^{-6}	50	750	154.6	28.8	361	Sparse	30sec
Bunny (Metal)	–	4.99×10^{10}	2.56×10^{10}	1×10^{-7}	0	2700	855.9	19.5	37114	Sparse	4h 40min
Blocks	5	5.00×10^{8}	1.00×10^{8}	8×10^{-6}	50	550	1596.2	428.1	1160	Dense	5h 28min
Boxes	5	5.00×10^{8}	1.00×10^{8}	8×10^{-6}	50	550	159.1	49.0	1160	Dense	5h 28min
The End (T)	6	1.49×10^{9}	7.70×10^{8}	2×10^{-7}	30	2700	247.7	15.2	71	Dense	42sec

Table 1: This table lists parameters that were used for each example object, the resulting frequency and decay for the object's primary mode, the number of nodes in the tetrahedral mesh, the method used for the modal decomposition, and the amount of time required to compute the decomposition. Once the model decomposition has been computed, all of the above examples can generate audio in real-time. For "Chimes" and "The End," the information listed is for the D3 tube and the letter T.

	Ideal	Measured		Computed	Adjusted	
Note	Freq.	Length	Freq.	Freq.	Length	Freq.
D3	587.33	.505	585.8	589.17	.5061	587.40
E3	659.26	.475	656.0	665.03	.4770	659.27
G3	783.99	.435	781.8	787.01	.4366	784.06
A4	880.00	.410	877.5	884.70	.4115	879.36
B4	987.77	.388	982.5	984.75	.3878	987.32
D4	1174.66	.353	1167.0	1186.88	.3548	1174.67

Table 2: The notes and ideal frequencies listed indicate the values specified by the manufacturer of the real wind chimes. The measured values were taken from the real wind chimes. The computed frequencies are what our model produced using the parameters from table 1 and the measured lengths. The adjusted values indicate the length and resulting frequency of the simulated chimes after tuning. Lengths are in meters and frequencies in Hertz.

figure 4 also shows the results of projecting a pair of impulses onto the retained modes of the bunny model.

The blocks and boxes shown in figure 5 illustrate how scale can effect the resulting audio. Both the boxes and blocks are geometrically similar: hollow cubes with a wall thickness of 5% their width. However, the boxes are 10× the size of the blocks. While the different scales are subtly revealed by the rigid-body motions (by the rate of acceleration with respect to the object sizes), the sounds produced by the two sets of objects are distinctly different, and the difference provides a clear cue as to the size of the objects.

As we discussed previously, similarities exist between the approach we have presented here and that presented in [van den Doel et al., 2001]. The main difference between the two methods is that we synthesize audio from only geometry and material properties whereas their system makes use of extensive measurements of a given object's response to impacts. Each of these methods presents distinct advantages: by relying on recorded data their method may easily match a given object, but our method is applicable when no real object or no robotic measuring devices are available. One direction that might be worth pursuing would be using their measured data for a given object to infer material parameters that could then be applied to the geometry of a different object. This approach might allow audio models for an entire set of cooking pots, for example, to be generated from measurements of a single pot in the set. It might also allow us to determine the sound made by a novel bell design, based on data from bells of similar materials, before we actually make the bell. Based on the good correspondence between our synthetic chimes and the physical set, we are optimistic about this direction of future work.

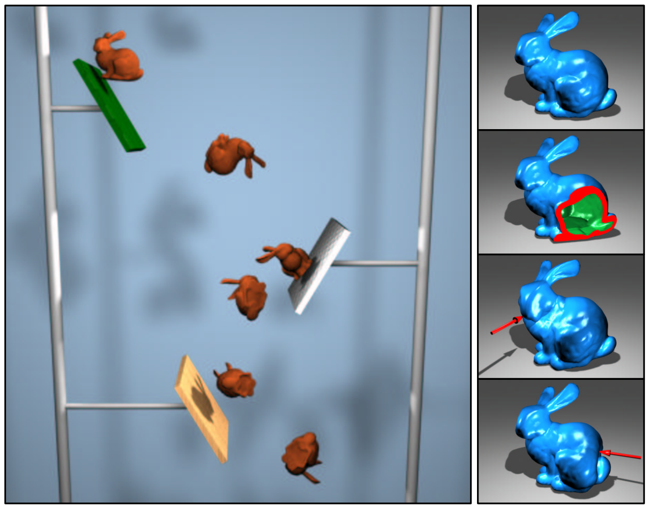
Figure 4: The left side of this figure shows an image from animation of several bunnies falling through a chute. Both the bunnies and the shelves are sounding. The images on the right show in order from top to bottom: the exterior of the bunny model, a cut-away revealing the wall thickness and hollow interior, modal response to an impulse on the bunny's nose, and the modal response to an impulse on the bunny's back. The impulse responses are greatly exaggerated for illustration.

Although the resolution of the mesh can affect the resulting audio, we have found that even very coarse meshes may be used for generating acceptable results. The meshes used for each of the letters shown in figure 6 are very coarse, yet the resulting audio is still acceptable. We have found that low mesh resolution tends to shift frequencies higher and may add a "hollow" quality to the sound. The frequency shifting may be partially compensated for by simply modifying the material parameters (*e.g.* raising the density) to compensate, so it will only be a problem if one is attempting to match a particular object (as we were for the wind chimes).

Although the modal decompositions may require up to a few hours of computation, this work needs only to be done once for a given object and audio can then be generated interactively. By precomputing the modal decomposition and storing it with an object, the approach we have presented could easily be applied to interactive applications such as video games that already employ rigid-body simulation methods. Additionally, because our method

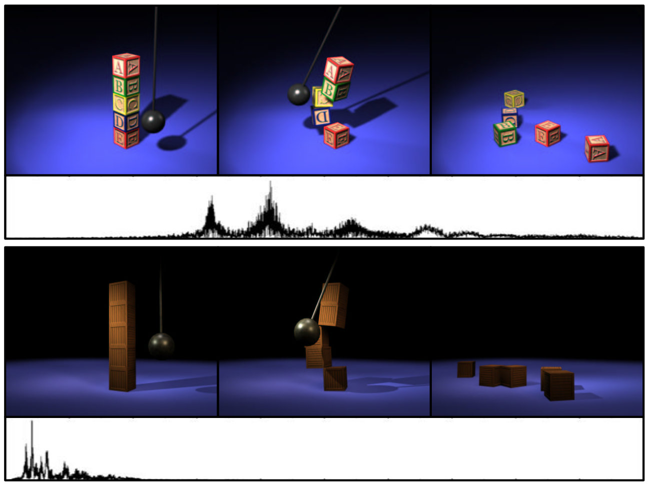

Figure 5: The top images show a stack of 5 cm blocks being knocked over. The images on the bottom show a stack of 50 cm boxes being knocked over. Only the blocks and boxes are sounding. Other than the 10× scale, both models are identical. The plots below each sequence show the frequency content of the resulting audio, indicating a significant difference in the sounds. (The horizontal axis ranges from 0 to 5000 Hz, the vertical axes are auto-scaled independently.)

requires only a geometric model and a handful of material parameters, the extra effort required to generated the audio model of a given object is minimal.

Acknowledgments

The authors thank Okan Arikan for allowing us to use his rigid body simulation code, and the members of the Berkeley Graphics Group for their helpful criticism and comments. This work was supported in part by Pixar Animation Studios, Intel Corporation, Sony Computer Entertainment America, a research grant from the Okawa Foundation, and NSF grant CCR-0204377.

References

ANDERSON, E., BAI, Z., BISCHOF, C., BLACKFORD, S., DEMMEL, J., DONGARRA, J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., MAKENNEY, A., AND SORENSEN, D. 1999. *LAPACK Users' Guide*, third ed. Siam, Philadelphia.

BAI, Z., DEMMEL, J., DONGARRA, J., RUHE, A., AND VAN DER VORST, H., Eds. 2000. *Templates for the solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia.

COOK, R. D., MALKUS, D. S., AND PLESHA, M. E. 1989. *Concepts and Applications of Finite Element Analysis*, third ed. John Wiley & Sons, New York.

COOK, P. R. 2002. *Real Sound Synthesis for Interactive Applications*. A. K. Peters, Ltd., Natick, Massachusetts.

FUNKHOUSER, T., CARLBOM, I., ELKO, G., PINGALI, G., SONDHI, M., AND WEST, J. 1998. A beam tracing approach to acoustic modeling for interactive virtual environments. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, 21–32.

FUNKHOUSER, T., MIN, P., AND CARLBOM, I. 1999. Real-time acoustic modeling for distributed virtual environments. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 365–374.

HAHN, J., GEIGEL, J., LEE, J., GRITZ, L., TAKALA, T., AND MISHRA, S. 1995. An integrated approach to sound and motion. *Journal of Visualization and Computer Animation 6*, 2, 109–123.

KINSLER, L. E., FREY, A. R., COPPENS, A. B., AND SANDERS, J. V. 2000. *Fundamentals of Acoustics*, fourth ed. John Wiley & Sons, New York.

MIN, P., AND FUNKHOUSER, T. 2000. Priority-driven acoustic modeling for virtual environments. *Computer Graphics Forum 19*, 3 (Aug.).

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 137–146.

O'BRIEN, J. F., COOK, P. R., AND ESSL, G. 2001. Synthesizing sounds from physically based motion. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 529–536.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: Modal dynamics for graphics and animation. In *Proceedings of SIGGRAPH 89*, Computer Graphics Proceedings, Annual Conference Series, 215–222.

SAVIOJA, L., HUOPANIEMI, J., LOKKI, T., AND VÄÄNÄNEN, R. 1997. Virtual environment simulation - advances in the DIVA project. In *Proceedings of the International Conference on Auditory Display (ICAD)*, 43–46.

TAKALA, T., AND HAHN, J. 1992. Sound rendering. In *Proceedings of SIGGRAPH 92*, Computer Graphics Proceedings, Annual Conference Series, 211–220.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Deformable models. *The Visual Computer 4*, 6, 306–331.

TSINGOS, N., FUNKHOUSER, T., NGAN, A., AND CARLBOM, I. 2001. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 545–552.

VAN DEN DOEL, K., AND PAI, D. K. 1996. Synthesis of shape dependent sounds with physical modeling. In *Proceedings of the International Conference on Auditory Display (ICAD)*.

VAN DEN DOEL, K., AND PAI, D. K. 1998. The sounds of physical shapes. *Presence 7*, 4, 382–395.

VAN DEN DOEL, K., KRY, P. G., AND PAI, D. K. 2001. Foley automatic: Physically-based sound effects for interactive simulation and animation. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 537–544.

WU, K., AND SIMON, H. 1999. TRLAN user guide. Tech. Rep. LBNL-42953, Lawrence Berkeley National Laboratory.

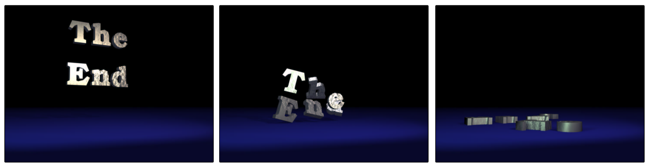

Figure 6: Select frames from an animation of the words "The End" falling onto a hard surface. Both the letters and the surface are sounding.