Synthesizing Sounds from Physically Based Motion

James F. O'Brien
University of California, Berkeley

Perry R. Cook
Princeton University

Georg Essl
Princeton University

Abstract

This paper describes a technique for approximating sounds that are generated by the motions of solid objects. The technique builds on previous work in the field of physically based animation that uses deformable models to simulate the behavior of the solid objects. As the motions of the objects are computed, their surfaces are analyzed to determine how the motion will induce acoustic pressure waves in the surrounding medium. Our technique computes the propagation of those waves to the listener and then uses the results to generate sounds corresponding to the behavior of the simulated objects.

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.

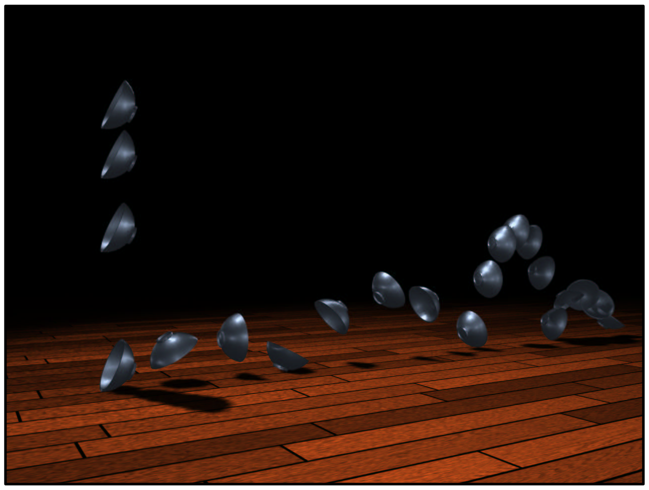
1 Introduction

One of the ultimate goals for computer graphics is to be able to create realistic synthetic environments that are indistinguishable from reality. While enormous progress has been made in areas such as modeling and rendering, realizing this broad objective requires more than just creating realistic static appearances. We must also develop techniques for making the dynamic aspects of the environment compelling as well.

Physically based animation techniques have proven to be a highly effective means for generating compelling motion. To date, several feature films have included high quality simulated motion for a variety of phenomena including water and clothing. Interactive applications, such as video games and training simulations, have also begun to make use of physically simulated motion. While computational limitations currently restrict the types of simulations that can be included in real-time applications, it is likely that these limitations will eventually be overcome through a combination of faster hardware and more efficient algorithms.

job@cs.berkeley.edu, prc@cs.princeton.edu, gessl@cs.princeton.edu

ACM SIGGRAPH 2001, 12–17 August 2001, Los Angeles, CA
©2001 ACM 1-58113-374-X/01/08...$5.00



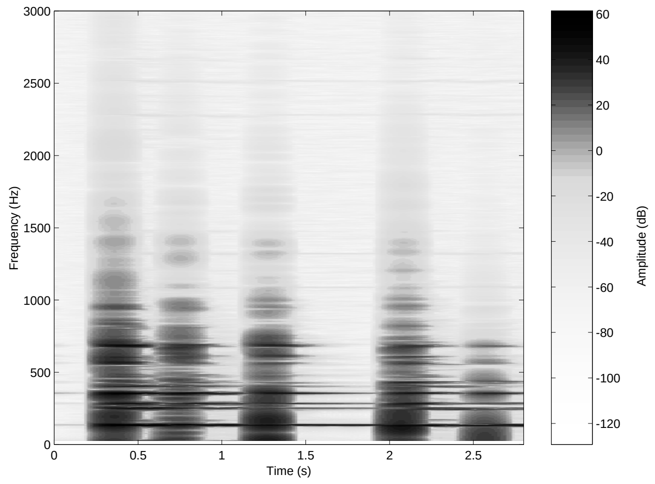


Figure 1: The top image shows a multi-exposure image from an animation of a metal bowl falling onto a hard surface. The lower image shows a spectrogram of the resulting audio for the first five impacts.

Although the field of computer graphics traditionally focuses on generating visuals, our perception of an environment encompasses other modalities in addition to visual appearance. Because these other modalities play an integral part in forming our impression of real-world environments, the graphics goal of creating realistic synthetic environments must also encompass techniques for modeling the perception of an environment through our other senses. For example, sound plays a large role in determining how we perceive events, and it can be critical to giving the user/viewer a sense of immersion.

The work presented in this paper addresses the problem of automatically generating physically realistic sounds for synthetic environments. Rather than making use of heuristic methods that are specific to particular objects, our approach is to employ the same simulated motion that is already being used for generating animated video to also generate audio. This task is accomplished by analyzing the surface motions of objects that are animated using a deformable body simulator, and isolating vibrational components that correspond to audible frequencies. The system then determines how these surface motions will generate acoustic pressure waves in the surrounding medium and models the propagation of those waves to the listener. For example, a finite element simulation of a bowl dropping onto the floor was used to compute both the image shown in Figure 1 and the corresponding audio.

Assuming that the computational cost of physically based animation is already justified for the production of visuals, the additional cost of computing the audio with our technique is negligible. The technique does not make use of specialized heuristics, assumptions about the shape of the objects, or pre-recorded sounds. The audio is generated automatically as the simulation runs and does not require any user interaction. Although we feel that the results generated with this technique are suitable for direct use in many applications, nothing precludes subsequent modification by another process or a Foley artist in situations where some particular effect is desired.

The remaining sections of this paper provide a detailed description of the sound generation technique we have developed, a review of related prior work, several examples of the results we have obtained, and a discussion of potential areas for future work. Presenting audio in a printed medium poses obvious difficulties. We include plots that illustrate the salient features of our results, and the proceedings video tape and DVD include animations with the corresponding audio.

2 Background

Prior work in the graphics community on sound generation and propagation has focussed on efficiently producing synchronized soundtracks for animations [18, 30], and on correctly modeling reflections and transmissions within the sonic environment [14, 15, 21]. In their work on modeling tearing cloth, Terzopoulos and Fleischer generated soundtracks by playing a pre-recorded sound whenever a connection in a spring mesh failed [31]. The DIVA project endeavored to create virtual musical performances in virtual spaces using physically derived models of musical instruments and acoustic ray-tracing for spatialization of the sound sources [27]. Funkhouser and his colleagues used beam tracing algorithms and priority rules to efficiently compute the direct and reflected paths from sound sources to receivers [15]. Van den Doel and Pai mapped analytically computed vibrational modes onto object surfaces allowing interactive sound generation for simple shapes [34]. Richmond and Pai experimentally derived modal vibration responses using robotic measurement systems, for interactive re-synthesis using modal filters [26]. More recent work by van den Doel, Kry, and Pai uses the output of a rigid body simulation to drive re-synthesis from the recorded data obtained with their robotic measurement system [33].

Past work outside of the graphics community on simulating the acoustics of solids for the purpose of generating sound has centered largely on the study of musical systems, such as strings, rigid bars, membranes, piano sound boards, and violin and guitar bodies. The techniques used include finite element and finite difference methods, lower dimensional simplifications, and modal/sinusoidal models of the eigenmodes of sound-producing systems.

Numerical simulations of bars and membranes have used either finite difference [3, 8, 10] or finite element methods [4, 5, 24]. Finite differencing approaches have also been used to model the behavior of strings [6, 7, 25].

Many current real-time techniques model the modes of acoustical systems, using resonant filters [1, 9, 35, 37] or additive sinusoidal synthesis [28]. In essence, modal modeling achieves efficiency by removing the spatial dynamics, and by replacing the actual physical system by an equivalent mass-spring system which models the same spectral response. However, the dynamics (in particular the propagation of disturbances) of the original system are lost. If the modal shapes are known, the spatial information can be maintained and spatial interactions can remain meaningful.

If certain assumptions are made, some systems can be modeled in reduced dimensionality. For example, if it is assumed that a drum head or the top-plate of a violin is very thin, a two-dimensional mesh can be used to simulate the transverse vibration [36].

In many systems such as strings and narrow columns of air, vibration can be considered one-dimensional, with the principal modes oriented along only one axis. McIntyre, Schumacher and Woodhouse's time-domain modeling technique has proven to be very useful in simulating musical instruments with a resonant system which is well approximated by the one-dimensional wave equation [20]. Such systems exhibit the d'Alembert solution, which is a decomposition of the one-dimensional wave equation into left-going and a right-going traveling wave components. Smith introduced extensions to the idea taken from scattering filter theory and coined the term "Waveguide Filters" for simulations based on this one-dimensional signal processing technique [29]. The waveguide wave-equation formulation can be modified to account for frequency dependent propagation speed due to stiffness, as described in [11]. In this technique, the propagation speeds around the eigenmodes of the system are modeled accurately, with errors introduced in damping at frequencies other than the eigenmodes.

When considering that the various efficient techniques described above are available, it should be noted that the approximations are flawed from a number of perspectives. For example, except for strings and simple bars, most shapes are not homogeneous. It is important to observe that for even moderate inhomogeneity, reflections at the points of changing impedance have to be expected that are not captured in a straightforward way. Further, the wave-equation and Euler-Bernoulli equation derivations also assume that the differential equations governing solid objects are linear. As a result the above methods can produce good results but only under very specific conditions. The method that we describe in the next section requires more computation than most of the above techniques, but it is much more general.

3 Sound Modeling

When solid objects move in a surrounding medium, they induce disturbances in the medium that propagate outward. For most fluid media, such as air or water, the significance of viscous effects decreases rapidly with distance, and the propagating disturbance has the characteristics of a pressure wave. If the magnitude of the pressure wave is of moderate intensity so that shock waves do not form and the relationship between pressure fluctuation and density change is approximately linear, then the waves are acoustic waves described by the equation

$$\frac{\partial^2 p}{\partial t^2} = c^2 \nabla^2 p \tag{1}$$

where t is time, p is the acoustic pressure defined as the difference between the current pressure and the fluid's equilibrium pressure, and c is the acoustic wave speed (speed of sound) in the fluid. Finally, if the waves reach a listener at a frequency between about $20\,\mathrm{Hz}$ and $20{,}000\,\mathrm{Hz}$, they will be perceived as sound. (See Chapter five of the text by Kinsler *et al.* [19] for a derivation of Equation (1).)

The remainder of this section describes our technique for approximating the sounds that are generated by the motions of solid objects. The technique builds on previous work in the field of physically based animation that uses deformable models to simulate the

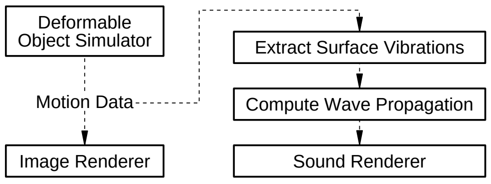


Figure 2: A schematic overview of joint audio and visual rendering.

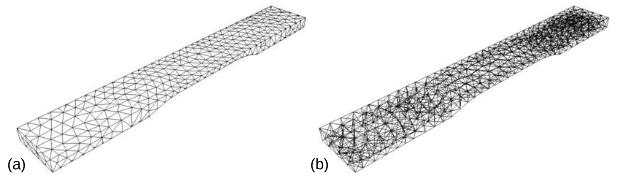


Figure 3: Tetrahedral mesh for an $\mathrm{F}^\sharp 3$ vibraphone bar. In (a), only the external faces of the tetrahedra are drawn; in (b) the internal structure is shown. Mesh resolution is approximately $1\,\mathrm{cm}$.

behavior of the objects. As the motion of the solid objects is computed, their surfaces are analyzed to determine how the motion will induce acoustic pressure waves in the surrounding media. The system computes the propagation of those waves to the listener and then uses the results to generate sounds corresponding to the simulated behavior. (See Figure 2.)

3.1 Motions of Solid Objects

The first step in our technique requires computing the motions of the animated objects that will be generating sounds. As these motions are computed, they will be used to generate both the audio and visual components of the animation.

Our system models the motions of the solid objects using a non-linear finite element method similar to the one developed by O'Brien and Hodgins [22, 23]. This method makes use of tetrahedral elements with linear basis functions to compute the movement and deformation of three-dimensional, solid objects. (See Figure 3.) Green's non-linear finite strain metric is used so that the method can accurately handle large magnitude deformations. A volume-based penalty method computes collision forces that allow the objects to interact with each other and with other environmental constraints. For the sake of brevity, we omit the details of this method for modeling deformable objects which are adequately described in [22].

We selected this particular method because it is reasonably fast, reasonably accurate, easy to implement, and treats objects as solids rather than shells. However, the sound generation process is largely independent of the method used to generate the object motion. So long as it fulfills a few basic criteria, another method for simulating deformable objects could be selected instead. These criteria are

- *Temporal Resolution* — Vibrations at frequencies as high as about 20,000 Hz generate audible sounds. If the simulation uses an integration time-step larger than approximately $10^{-5}\,\mathrm{s}$, then it will not be able to adequately model high frequency vibrations.

- *Dynamic Deformation Modeling* — Most of the sounds that an object generates as it moves arise from vibrations driven by elastic deformation. These vibrations will not be present with techniques that do not model deformation (*e.g.* rigid body simulators). Similarly, these vibrations will not be present with inertia-less techniques.

- *Surface Representation* — Because the surfaces of the objects are where vibrations transition from the objects to the surrounding medium, the simulation technique must contain some explicit representation of the object surfaces.

- *Physical Realism* — Simulation techniques used for physically based animation must produce motion that is visually acceptable for the intended application. Generating sounds from the motion will reveal additional aspects of the motion that may not have been visibly apparent, so a simulation method used to generate audio must compute motion that is accurate enough to sound acceptable in addition to looking acceptable.

The tetrahedral finite element method we are using meets all of these criteria, but so do many other methods that are commonly used for physically based animation. For example, a mass and spring system would be suitable provided the exterior faces of the system were defined.

3.2 Surface Vibrations

Once we have a method for computing the motion of the objects, the next step in the process requires analyzing the surface's motions to determine how it will affect the pressure in the surrounding fluid. Let Ω be the surface of the moving object(s), and let ds be a differential surface element in Ω with unit normal $\hat{\boldsymbol{n}}$ and velocity $\boldsymbol{v}$. If we neglect viscous shear forces then the acoustic pressure, p, of the fluid adjacent to ds is given by

$$p = z\boldsymbol{v} \cdot \hat{\boldsymbol{n}}. \qquad (2)$$

where $z = \rho c$ is the fluid's specific acoustic impedance. From [19], the density of air at 20°C under one atmosphere of pressure is $\rho = 1.21\,\mathrm{kg/m}^3$, and the acoustic wave speed is $c = 343\,\mathrm{m/s}$, giving $z = 415\,\mathrm{Pa}\cdot\mathrm{s/m}$.

Representing the pressure field over Ω requires some form of discretization. We will assume that a triangulated approximation of Ω exists (denoted Ω^*) and we will approximate the pressure field as being constant over each of the triangles in Ω^*.

Each triangle is defined by three nodes. The position, $\boldsymbol{x}$, and velocity, $\dot{\boldsymbol{x}}$, of each node are computed by a physical simulation method as discussed in Section 3.1. We will refer to the nodes of a given triangle by indexing with square brackets. For example, $\boldsymbol{x}_{[2]}$ is the position in world coordinates of the triangle's second node. The surface area of each triangle is given by

$$a = ||(\boldsymbol{x}_{[2]} - \boldsymbol{x}_{[1]}) \times (\boldsymbol{x}_{[3]} - \boldsymbol{x}_{[1]})||/2, \qquad (3)$$

and its unit normal by

$$\hat{\boldsymbol{n}} = \frac{(\boldsymbol{x}_{[2]} - \boldsymbol{x}_{[1]}) \times (\boldsymbol{x}_{[3]} - \boldsymbol{x}_{[1]})}{2a}. \qquad (4)$$

The average pressure over the triangle is computed by substituting the triangle's normal and average velocity, $\bar{\boldsymbol{v}}$, into Equation (2) so that

$$\bar{p} = z\bar{\boldsymbol{v}} \cdot \hat{\boldsymbol{n}} = z\left(\frac{1}{3}\sum_{i=1}^{3}\dot{\boldsymbol{x}}_{[i]}\right) \cdot \hat{\boldsymbol{n}}. \qquad (5)$$

The variable $\bar{p}$ tells us how the pressure over a given triangle fluctuates, but we are only interested in fluctuations that correspond to frequencies in the audible range. Frequencies above this range will need to be removed or they will cause aliasing artifacts.[1] Lower

[1]We assume that the simulation time-step is smaller than the audio sampling period. This is the case for our examples which used an integration time-step between 10^{-6} s and 10^{-7} s.

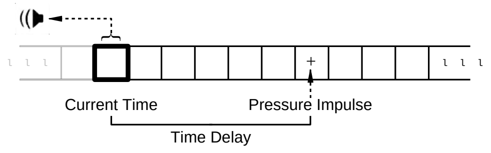


Figure 4: One-dimensional accumulation buffer used to account for travel time delay.

frequencies will not cause aliasing problems, but they will interact with later computations to create other difficulties. For example, an object moving at a constant rate will generate a large, constant pressure in front of it. The corresponding constant term will show up as an inconvenient offset in the final audio samples. More importantly, it may interact with latter visibility computations to create unpleasant artifacts. To remove undesirable frequency components, we make use of two filters that are applied to the pressure variable at each triangle in Ω^*.

First, a low-pass filter is applied to $\bar{p}$ to remove high frequencies. The low-pass filter is implemented using a normalized kernel, K, built from a windowed sinc function given by

$$K_i = \operatorname{sinc}(i\Delta t)\cdot \operatorname{win}(i/w) \quad , \ i \in [-w,\ldots,w] \tag{6}$$

$$\operatorname{sinc}(t) = \frac{\sin(2\pi f_{\max} t)}{\pi t} \tag{7}$$

$$\operatorname{win}(u) = \begin{cases} 1/2 + \cos(\pi u)/2 & : \ |u| \le 1 \\ 0 & : \ |u| > 1 \end{cases} , \tag{8}$$

where $f_{\max}$ is the highest frequency to be retained, Δt is the simulation time-step, and w is the kernel's half-width. The low-pass filtered pressure, g, is obtained by convolving $\bar{p}$ with K and sub-sampling the result down to audio rate.

The second filter is a DC-blocking filter that will remove any constant component and greatly attenuate low-frequency ones. It works by differentiating a signal and then re-integrating the signal using a "lossy integrator." The final filtered pressure, $\tilde{p}$, after application of the DC-blocking filter is given by

$$\tilde{p}_i = (1-\alpha)\tilde{p}_{i-1} + (g_i - g_{i-1}), \tag{9}$$

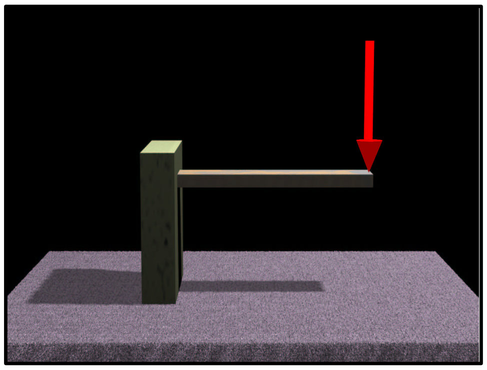
where α is a loss constant between zero and one, g is the low-pass filtered pressure, and the subscripts index time at audio rate.

For the examples presented in this paper, $f_{\max}$ was $22{,}050$ Hz and we sub-sampled to an audio rate of $44{,}100$ Hz. The low-pass filter kernel's half-width was three times the wavelength of $f_{\max}$ ($w = \lceil 3/(f_{\max}\Delta t)\rceil$). The value of α was selected by trial and error with $\alpha = 0.1$ yielding good results.

3.3 Wave Radiation and Propagation

Once we know the pressure distribution over the surface of the objects we must compute how the resulting wave propagates outward towards the listener. The most direct way of accomplishing this task would involve modeling the region surrounding the objects with Equation (1), and using the pressure field over Ω as prescribed boundary conditions. This approach would lead to a coupled solid/fluid simulation. Unfortunately, the additional cost of the fluid simulation would not be trivial. Instead, we can make a few simplifying assumptions and use a much more efficient solution method.

Huygen's principle states that the behavior of a wavefront may be modeled by treating every point on the wavefront as the origin of a spherical wave, which is equivalent to stating that the behavior of a complex wavefront can be separated into the behavior of a set of simpler ones [17]. Using this principle, we can approximate the



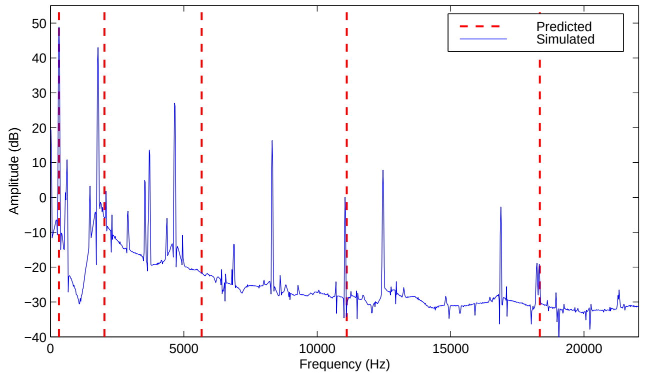


Figure 5: Spectrum generated by plucking the free end of a clamped bar. Predicted values are taken from [19].

result of propagating a single pressure wave outward from Ω by summing the results of a many simpler waves, each propagating from one of the triangles in Ω^*.

If we assume that the environment is anechoic (no reflections) and we ignore the effect of diffraction around obstacles, then a reasonable approximation for the effect on a distant receiver of the pressure wave generated by a triangle in Ω^* is given by

$$s = \frac{\tilde{p}\, a\, \delta_{\bar{\boldsymbol{x}}\to r}}{||\bar{\boldsymbol{x}} - \boldsymbol{r}||}\cos(\theta), \tag{10}$$

where $\boldsymbol{r}$ is the location of the receiver, $\bar{\boldsymbol{x}}$ is the center of the triangle, θ is the angle between the triangle's surface normal and the vector $\boldsymbol{r} - \bar{\boldsymbol{x}}$, and $\delta_{\bar{x}\to r}$ is a visibility term that is one if an unobstructed ray can be traced from $\bar{\boldsymbol{x}}$ to $\boldsymbol{r}$ and zero otherwise.[2] The $\cos(\theta)$ is a rough approximation to the first lobe of the frequency-dependent beam function for a flat plate [19].

Equation (10) is nearly identical to similar equations that are used in image rendering with local lighting models, and the decision to ignore reflected and diffracted sound waves is equivalent to ignoring secondary illumination. A minor difference is that the falloff term is inversely proportional to distance, not to distance squared. The sound intensity, measured in energy per unit time and area, does falloff with distance squared, but eardrums and microphones react to pressure which is proportional to the square-root of intensity [32].

[2]The center of a triangle is computed by averaging the locations of its vertices and low-pass filtering the result using the sinc kernel from Equation (6). Likewise, the normal is obtained from low-pass filtered vertex locations. The filtering is necessary because the propagation computations are performed at audio rate, not simulation rate, and incorrectly sub-sampling the triangle centers or normals will result in audible aliasing artifacts.

A more significant difference arises because sound travels several orders of magnitude slower than light, and we cannot assume that sound propagation is instantaneous. In fluids such as air, sound does travel rapidly enough ($c = 343\,\mathrm{m/s}$) that we may not directly notice the delay except over large distances, but we do notice indirect effects even at small distances. For example, very small delays are responsible for both Doppler shifting and the generation of interference patterns. Additionally, if we wish to compute stereo sound by using multiple receiver locations, then delay differences between a human listener's ears as small as $20\,\mu\mathrm{s}$ provide important cues for localizing sounds [2].

To account for propagation delay we make use of a one-dimensional accumulation buffer that stores audio samples. All entries in the buffer are initially set to zero. When we compute s for each of the triangles in Ω^* at a given time, we also compute a corresponding time delay

$$d = \frac{||\bar{\boldsymbol{x}} - \boldsymbol{r}||}{c} \ . \tag{11}$$

The s values are then added to the entries of the buffer that correspond to the current simulation time plus the appropriate delay. (See Figure 4.)

In general, d will not be a multiple of the audio sampling rate. If we were to round to the nearest entry in the accumulation buffer, the resulting audio would contain artifacts akin to the "jaggies" that occur when scan-converting lines. These artifacts will manifest themselves in the form of an unpleasant buzzing sound as if a saw-tooth wave had been added to the result. To avoid this problem, we add the s values into the buffer by convolving the contribution with a narrow (two samples wide) Gaussian and "splatting" the result into the accumulation buffer.

As the simulation advances forward in time, values are read from the entry in the accumulation buffer that corresponds to the current time. This value is treated as an audio sample that is sent to the output sink. If stereo sound is desired we compute this propagation step twice, once for each ear.

4 Results

We have implemented the described technique for generating audio and tested the system on several examples. For all of the examples, two listener locations were used to produce stereo audio. The locations are centered around the virtual viewpoint and separated by $20\,\mathrm{cm}$ along a horizontal axis that is perpendicular to the viewing direction. The sound spectra shown in the following figures follow the convention of plotting frequency amplitude using decibels, so that the vertical axes are scaled logarithmically.

Figure 1 shows an image taken from an animation of a bowl falling onto a hard surface and a spectrogram of the resulting audio. In this example, only the surface of the bowl is used to generate the audio, and the floor is modeled as rigid constraint. The spectrogram reveals the bowl's vibrational modes as darker horizontal lines. Variations in the degree to which each of the modes are excited occur because different parts of the bowl hit the surface on each bounce.

The bowl's shape is arbitrary and there is no known analytical solution for its vibrational modes. While being able to model arbitrary shapes is a strength of the proposed method, we would like to verify its accuracy by comparing its results to known solutions. Figure 5 shows the results computed for a rectangular bar that is clamped at one end and excited by applying a vertical impulse to the other. The accompanying plot shows the spectrum of the resulting sound with vertical lines indicating the mode frequencies predicted by a well known analytical solution [19]. Although the

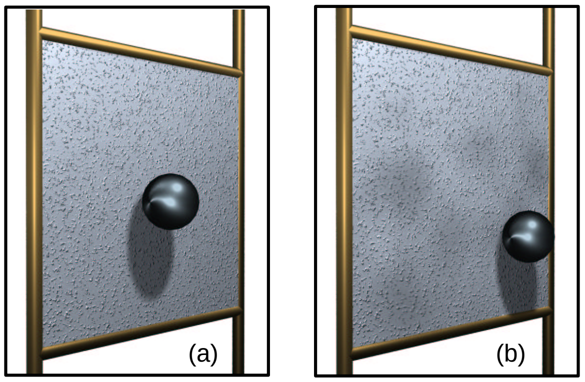
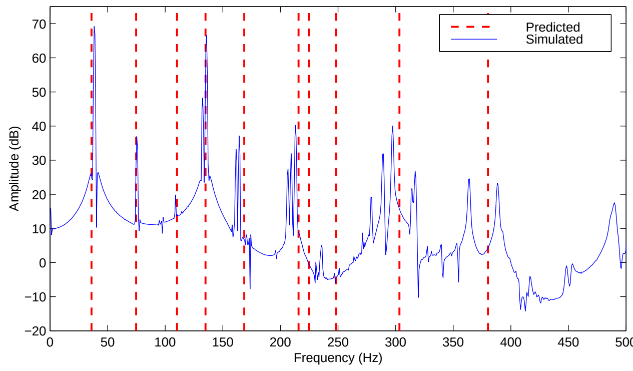


Figure 6: A square plate being struck (a) on center and (b) off center.



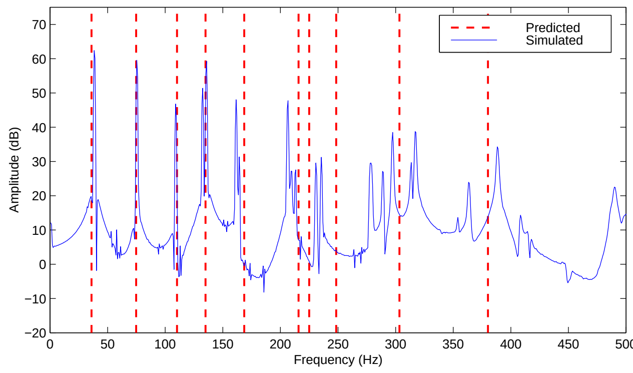


Figure 7: Spectrum generated by striking the square plate shown in Figure 6 in the center (top) and off from center (right). Predicted values are taken from [12].

results correspond reasonably well, the simulated results are noticeably flatter than the theoretical predictions. One possible explanation for the difference is that the analytical solution assumes the bar to be perfectly elastic, while our simulated bar experiences internal damping.

Figure 6 shows two trials from a simulation of a square plate (with finite thickness) that is held fixed along its border while being struck by a weight. In the first trial the weight hits the plate on-center, while in the second trial the weight lands off-center horizontally by 25% of the plate's width and vertically by 17%. The frequencies predicted by the analytical solution given by Flecher and Rossing [12] are overlaid on the spectra of the two resulting sounds. As with a real plate, the on-center strike does not significantly excite vibrational modes that have nodes lines passing

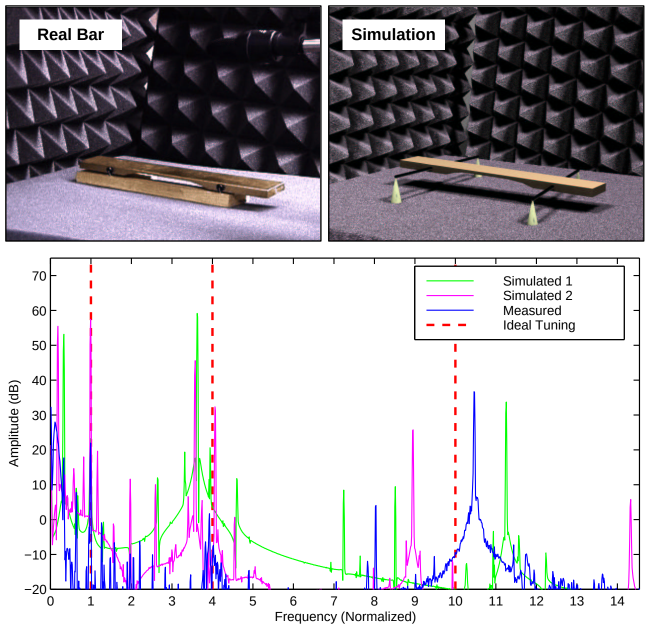


Figure 8: The top image plots a comparison between the spectra of a real vibraphone bar (*Measured*), and simulated results for a low-resolution (*Simulated 1*) and high-resolution mesh (*Simulated 2*). The vertical lines located at 1, 4, and 10 show the tuning ratios reported in [12].

through the center of the plate (*e.g.* the modes indicated by the second and third dashed, red lines). The off-center strike does excite these modes and a distinct difference can be heard in the resulting audio. The simulation's lower modes match the predicted ones quite well, but for the higher modes the correlation becomes questionable. Poor resolution of the higher modes is to be expected as they have a lower signal-to-noise ratio and are more easily affected by discretization and other error sources.

The bars in a vibraphone are undercut so that the first three partials are tuned according to a $1{:}4{:}10$ ratio. While this modification makes the instrument sound pleasant, the change in transverse impedance between the thin and thick portions of the bar prevent an analytical solution. We ran two simulations of a $36\,\mathrm{cm}$ long bar with mesh resolutions of $1\,\mathrm{cm}$ and $2\,\mathrm{cm}$, and compared them to a recording of a real bar being struck. (The $1\,\mathrm{cm}$ mesh is shown in Figure 3.) To facilitate the comparison, the simulated audio was warped linearly in frequency to align the first partials to that of the real bar at $187\,\mathrm{Hz}$ ($\mathrm{F}^\sharp3$), which is equivalent to adjusting the simulated bar's density so that is matches the real bar. The results of this comparison are shown in Figure 8. Although both the simulated and real bars differ slightly from the ideal tuning, they are quite similar to each other. All three sounds also contain a low frequency component below the bar's first mode that is created by the interaction with the real or simulated supports.

The result of striking a pendulum with a fast moving weight is shown in Figure 9. Because of Doppler effects, the pendulum's periodic swinging motion should modulate both the amplitude and the frequency of the received sound. Because our technique accounts for both distance attenuation and travel delay, it can model these phenomena. The resulting modulation is clearly visible in the spectrogram (particularly in the line near $500\,\mathrm{Hz}$) and can be heard in the corresponding audio.

Because this sound generation technique does not make additional assumptions about how waves travel in the solid objects, it can be used with non-linear simulation methods to generate sounds for objects whose internal vibrations are not modeled by the linear wave equation. The finite element method we are using employs a non-linear strain metric that is suitable for modeling large deformations. Figure 10 shows frames from two animation of a ball dropping onto a sheet. In the first one, the sheet is nearly rigid and the ball rolls off. In the second animation, the sheet is highly compliant and it undergoes large deformations as it interacts with the ball. Another example demonstrating large deformations is shown in Figure 11 where a slightly bowed sheet is being bent back and forth to create crinkling sound. Animations containing the audio for these, and other, examples have been included on the proceedings video tape and DVD. Simulations times are listed in Table 1.

5 Conclusions and Future Work

In this paper, we have described a general technique for computing physically realistic sounds that takes advantage of existing simulation methods already used for physically based animation. We have also presented a series of examples that demonstrate the results that can be achieved when our technique is used with a particular, finite element based simulation method.

One area for future work is to combine this sound generation technique with other simulation methods. As discussed in Section 3.1, it should be possible to generate audio from most deformable body simulation methods such as mass and spring systems or dynamic cloth simulators. It may also be possible to generate audio from some fluid simulation methods, such as the method developed by Foster and Metaxas for simulating water [13].

Of the criteria we listed in Section 3.1, we believe that the required temporal resolution is most likely to pose difficulties. If the simulation integrator uses time-steps that are larger than about $10^{-5}\,\mathrm{s}$, the higher frequencies of the audible spectrum will not be sampled adequately so that, at best, the resulting audio will sound dull and soggy. Unfortunately, small time-steps result in slow simulations, and as a result a significant amount of research has focused on finding ways to allow numerical integrators to take larger steps while remaining stable. In general, the larger time-steps are achieved by removing the high-frequency components from the motions being computed. While removing these high-frequency components will at some point create visually objectionable artifacts, it is likely that sound quality will be adversely affected first.

Rigid body simulations are also excluded by our criteria because they do not model the deformations that drive most of the vibrations that produce sound. This limitation is particularly unfortunate because rigid body simulations are widely used, particularly in interactive applications. Because they are so commonly used, developing general methods for computing sounds for rigid body and large time-step simulation methods is an important area for future work.

Although our sound propagation model is relatively cheap to compute, it is also quite limited. Reflected and diffracted sound transport often play a significant role in determining what we hear in an environment. To draw an analogy with image rendering, our current method is roughly equivalent a local illumination model and adding reflection and diffraction would be equivalent to stepping up to global illumination. In some ways global sound computations would actually be more complex than global illumination because one cannot assume that waves propagate instantaneously. Other researchers have investigated techniques for modeling acoustic reflections, for example [14], and combining our work with theirs would probably be useful.

Our listener model could also be improved. As currently implemented, a listener receives pressure waves equally well from all directions. While this behavior is ideal for an omni-directional microphone, human ears and most real microphones behave quite differently. One obvious effect is that the human head acts as a blocker so that high frequency sounds from the side tend to be heard better

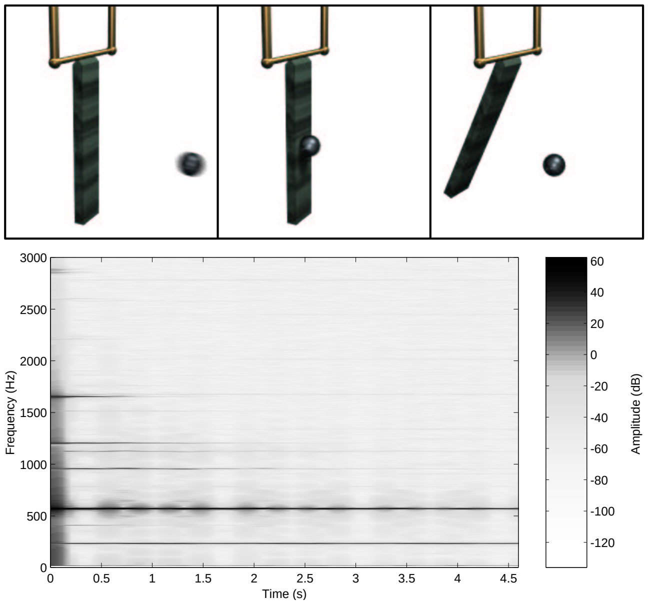


Figure 9: The spectrogram produced by a swinging bar after it is struck by a weight.

by that ear while low frequencies diffract around the head and are heard by both ears. Other, more subtle effects, such as the pattern of reflections off of the outer ear, also contribute to allowing us to localize the sounds we hear. Other researchers have taken extensive measurements to determine how sounds are affected as they enter a human ear, and used the compiled data to build head-related transfer functions [2, 16]. Filters derived from these transfer functions have been used successfully for generating three-dimensional spatialized audio. We are currently working on adding this functionality to our existing system.

Our primary goal in this work has been to generate a tool that is useful for generating audio. However, we have also noticed that the audio produced by a simulation makes an excellent debugging tool. For example, we have observed that symptoms of a simulation going unstable often can be heard before they become visible. Other common simulation errors, such as incorrect collision response, are also evidenced by distinctively odd sounds. As physically based animation continues to becomes more commonly used, sound generation could become useful not only for its own sake but also as standard tool for working with and debugging simulations.

Acknowledgments

The images in this paper were generated using Alias—Wavefront's Maya and Pixar Animation Studios' RenderMan software running on Silicon Graphics hardware. This research was supported, in part, by a research grant from the Okawa foundation, by NSF grant number 9984087, and by the State of New Jersey Commission on Science and Technology Grant Number 01-2042-007-22.

References

[1] J. M. Adrien. The missing link: Modal synthesis. In De Poli, Piccialli, and Roads, editors, *Representations of Musical Signals*, chapter 8, pages 269–297. MIT Press, Cambridge, 1991.

[2] D. R. Begault. *3-D Sound for Virtual Reality and Multimedia*. Academic Press, 1994.

[3] I. Bork. Practical tuning of xylophone bars and resonators. *Applied Acoustics*, 46:103–127, 1995.

[4] I. Bork, A. Chaigne, L.-C. Trebuchet, M. Kosfelder, and D. Pillot. Comparison between modal analysis and finite element modeling of a marimba bar. *Acoustica united with Acta Acustica*, 85(2):258–266, March 1999.

[5] J. Bretos, C. Santamaría, and J. A. Moral. Finite element analysis and experimental measurements of natural eigenmodes and random responses of wooden bars used in musical instruments. *Applied Acoustics*, 56:141–156, 1999.

[6] A. Chaigne and A. Askenfelt. Numerical simulations of piano strings: I. a physical model for a struck string using finite difference methods. *Journal of the Acoustical Society of America*, 95(2):1112–1118, February 1994.

[7] A. Chaigne and A. Askenfelt. Numerical simulations of piano strings: II. comparisons with measurements and systematic exploration of some hammer-string parameters. *Journal of the Acoustical Society of America*, 95(3):1631–1640, March 1994.

[8] A. Chaigne and V. Doutaut. Numerical simulations of xylophones: I. time-domain modeling of the vibrating bars. *Journal of the Acoustical Society of America*, 101(1):539–557, January 1997.

[9] P. R. Cook. Physically informed sonic modeling (PhISM): Synthesis of percussive sounds. *Computer Music Journal*, 21(3):38–49, 1997.

[10] V. Doutaut, D. Matignon, and A. Chaigne. Numerical simulations of xylophones: II. time-domain modeling of the resonator and of the radiated sound pressure. *Journal of the Acoustical Society of America*, 104(3):1633–1647, September 1998.

[11] G. Essl and P. R. Cook. Measurements and efficient simulations of bowed bars. *Journal of the Acoustical Society of America*, 108(1):379–388, 2000.

[12] N. H. Fletcher and T. D. Rossing. *The Physics of Musical Instruments*. Spreinger–Verlag, New York, second edition, 1997.

[13] N. Foster and D. Metaxas. Realistic animation of liquids. In *Proceeding Graphics Interface '96*, pages 204–212. Canadian Human-Computer Communications Society, May 1996.

[14] T. Funkhouser, I. Carlbom, G. Elko, G. Pingali, M. Sondhi, and J. West. A beam tracing approach to acoustic modeling for interactive virtual environments. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 21–32, July 1998.

[15] T. Funkhouser, P. Min, and I. Carlbom. Real-time acoustic modeling for distributed virtual environments. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 365–374, August 1999.

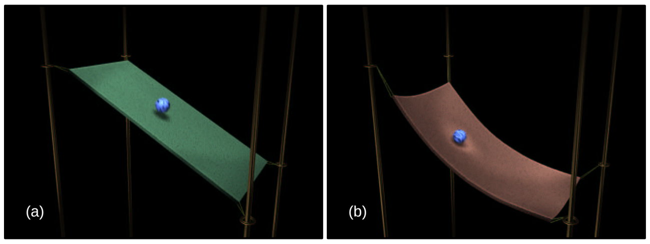


Figure 10: These figures show a round weight being dropped onto two different surfaces. The surface shown in (a) is rigid while the one shown in (b) is more compliant.

Example	Figure	Simulation Δt	Nodes	Elements	Surface Elements	Total Time	Audio Time	
Bowl	1	1×10^{-6} s	387	1081	742	91.3 min	4.01 min	4.4%
Clamped Bar	5	1×10^{-7} s	125	265	246	240.4 min	1.26 min	0.5%
Square Plate (on center)	6.a	1×10^{-6} s	688	1864	1372	245.0 min	8.14 min	3.3%
Square Plate (off center)	6.b	1×10^{-6} s	688	1864	1372	195.7 min	7.23 min	3.7%
Vibraphone Bar	8	1×10^{-7} s	539	1484	994	1309.7 min	5.31 min	0.4%
Swinging Bar	9	3×10^{-7} s	130	281	254	88.4 min	1.42 min	1.6%
Rigid Sheet	10.a	6×10^{-7} s	727	1954	1438	1041.8 min	7.80 min	0.7%
Compliant Sheet	10.b	2×10^{-6} s	727	1954	1438	313.1 min	7.71 min	2.5%
Bent Sheet	11	1×10^{-7} s	678	1838	1350	1574.2 min	6.45 min	0.4%

Table 1: The total times indicate the total number of minutes required to compute one second of simulated data, including graphics and file I/O. The audio times listed indicate the amount of the total time that was spent generating audio, including related file I/O. The percentages listed indicate the time spent generating audio as a percentage of the total simulation time. Timing data were measured on an SGI Origin using one 350 MHz MIPS R12K processor while unrelated processes were running on the machine's other processors.

[16] W. G. Gardner and K. D. Martin. HRTF measurements of a KEMAR dummy head microphone. Technical Report 280, MIT Media Lab Perceptual Computing, 1994.

[17] J. W. Goodman. *Introduction to Fourier Optics*. McGraw-Hill Higher Education, New York, second edition, 1996.

[18] J. Hahn, J. Geigel, J. Lee, L. Gritz, T. Takala, and S. Mishra. An integrated approach to sound and motion. *Journal of Visualization and Computer Animation*, 6(2), 1995.

[19] L. E. Kinsler, A. R. Frey, A. B. Coppens, and J. V. Sanders. *Fundamentals of Acoustics*. John Wiley and Sons, Inc., New York, fourth edition, 2000.

[20] M. E. McIntyre, R. T. Schumacher, and J. Woodhouse. On the oscillation of musical instruments. *Journal of the Acoustical Society of America*, 74(5):1325–1344, November 1983.

[21] P. Min and T. Funkhouser. Priority-driven acoustic modeling for virtual environments. *Computer Graphics Forum*, 19(3), August 2000.

[22] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 137–146, August 1999.

[23] J. F. O'Brien and J. K. Hodgins. Animating fracture. *Communications of the ACM*, 43(7):68–75, July 2000.

[24] F. Orduña-Bustamante. Nonuniform beams with harmonically related overtones for use in percussion instruments. *Journal of the Acoustical Society of America*, 90(6):2935–2941, December 1991. Erratum: J. of Acous. Soc. Am. 91(6):3582–3583.

[25] M. Palumbi and L. Seno. Physical modeling by directly solving wave PDE. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 325–328, Beijing, China, October 1999. International Computer Music Association (ICMA).

[26] J. L. Richmond and D. K. Pai. Robotic measurement and modeling of contact sounds. In *Proceedings of the International Conference on Auditory Display (ICAD)*, 2000.

[27] L. Savioja, J. Huopaniemi, T. Lokki, and R. Väänänen. Virtual environment simulation - advances in the DIVA project. In *Proceedings of the International Conference on Auditory Display (ICAD)*, pages 43–46, 1997.

[28] X. Serra. A computer model for bar percussion instruments. In *Proc. International Computer Music Conference (ICMC)*, pages 257–262, The Hague, 1986. International Computer Music Association (ICMA).

[29] J. O. Smith. Efficient simulation of the reed-bore and bowstring mechanisms. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 275–280, The Hague, 1986. International Computer Music Association (ICMA).

[30] T. Takala and J. Hahn. Sound rendering. In *Proceedings of SIGGRAPH 92*, Computer Graphics Proceedings, Annual Conference Series, pages 211–220, July 1992.

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

[32] B. Truax, editor. *Handbook for Acoustic Ecology*. Cambridge Street Publishing, second edition, 1999.

[33] K. van den Doel, P. G. Kry, and D. K. Pai. Foley automatic: Physically-based sound eects for interactive simulation and animation. In *Proceedings of SIGGRAPH 2001*, August 2001.

[34] K. van den Doel and D. K. Pai. Synthesis of shape dependent sounds with physical modeling. In *Proceedings of the International Conference on Auditory Display (ICAD)*, 1996.

[35] K. van den Doel and D. K. Pai. The sounds of physical shapes. *Presence*, 7(4):382–395, 1998.

[36] S. van Duyne and J. O. Smith. Physical modeling with the 2D waveguide mesh. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 40–47, The Hague, 1993. International Computer Music Association (ICMA).

[37] J. Wawrzynek. VLSI models for sound synthesis. In M. Mathews and J. Pierce, editors, *Current Directions in Computer Music Research*, chapter 10, pages 113–148. MIT Press, Cambridge, 1989.

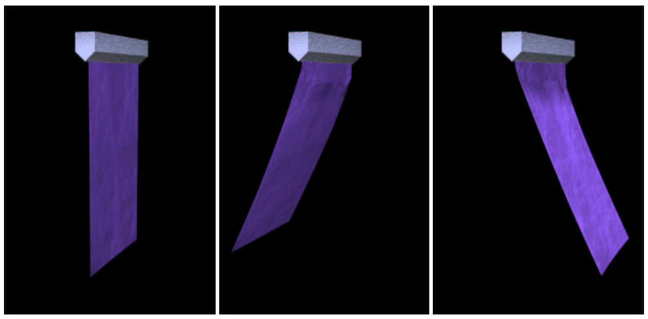


Figure 11: A slightly bowed sheet being bent back and forth.