北京理工大學

本科生毕业设计（论文）外文翻译

外文原文题目： Interactive Sound Synthesis for Large Scale Environment

中文翻译题目： 大型场景交互声音合成

毕业设计（论文）题目： 基于物理模型的碰撞与滚动声音实时模拟

姓　　名： 王铎暾

学　　院： 计算机学院

班　　级： 07111506

指导教师： 刘庆晖

Interactive Sound Synthesis for Large Scale Environments

Nikunj Raghuvanshi[1] Ming C. Lin[†]

Department of Computer Science
University of North Carolina at Chapel Hill

Abstract

We present an interactive approach for generating realistic physically-based sounds from rigid-body dynamic simulations. We use spring-mass systems to model each object's local deformation and vibration, which we demonstrate to be an adequate approximation for capturing physical effects such as magnitude of impact forces, location of impact, and rolling sounds. No assumption is made about the mesh connectivity or topology. Surface meshes used for rigid-body dynamic simulation are utilized for sound simulation without any modifications. We use results in auditory perception and a novel priority-based quality scaling scheme to enable the system to meet variable, stringent time constraints in a real-time application, while ensuring minimal reduction in the perceived sound quality. With this approach, we have observed up to an order of magnitude speed-up compared to an implementation without the acceleration. As a result, we are able to simulate moderately complex simulations with upto hundreds of sounding objects at over 100 frames per second (FPS), making this technique well suited for interactive applications like games and virtual environments. Furthermore, we utilize OpenAL and EAX™ on Creative Sound Blaster Audigy 2™ cards for fast hardware-accelerated propagation modeling of the synthesized sound.

CR Categories: H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing—Modeling, Methodologies and techniques; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism— Animation;

Keywords: Sound Synthesis, Rigid-Body Simulation, OpenAL

1 Introduction

Most interactive applications today employ recorded sound clips for providing sounds corresponding to object interactions in a scene. Although this approach has the advantage that the sounds are realistic and the sound-generation process is quite fast, there are many physical effects which cannot be captured by such a technique. For instance, in a typical collision between two objects, the loudness and timbre of the sound is determined by the magnitude and location of the impact forces – a plate sounds very differently when struck on the edge compared to when it is struck in the middle. Consequently, if the collision scenario changes slightly, the sound exhibits a corresponding change. Such subtle effects can add substantial realism to a typical scene by avoiding the repetitiveness common to recorded sound clips. However, developing a system which produces sound using physically-based principles in real time poses

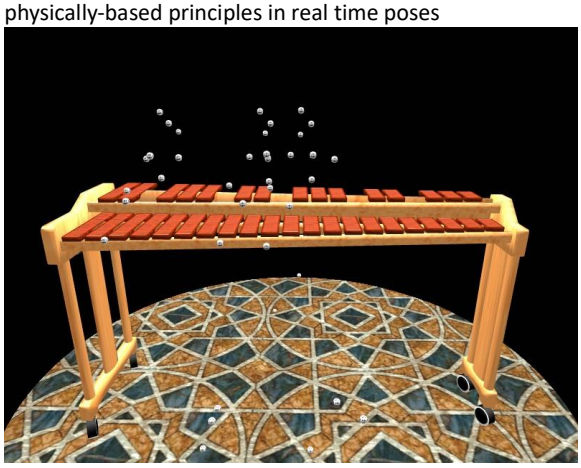

Figure 1: Numerous dice fall on a three-octave xylophone in close succession, playing out the song "The Entertainer" (see the video). Our algorithm is able to produce the corresponding musical tones at more than 500 FPS for this complex scene, with audio generation taking 10% of the total CPU time, on a 3.4GHz Pentium-4 Laptop with 1GB RAM.

substantial difficulties. The foremost requirement is the presence of an efficient dynamics engine which informs the sound system of object collisions and the forces involved. For this work, we have developed a fast and robust rigid-body simulator, but many present day games meet this requirement. Given a dynamics simulator, the main challenge is to synthesize the sound efficiently enough to play in real time while taking only a small portion of the total running time, which is usually dominated by graphics and rigid-body simulation. Typically the sound system can only afford a few hundred CPU cycles per object per sound sample for many interactive applications.

Main Results: In this paper, we present an approach which meets the interactive performance requirements outlined above, while ensuring high realism and fidelity of the sound produced. Given an object's geometry and a few material parameters, we construct a spring-mass model approxima-

[1] nikunj@cs.unc.edu [†]lin@cs.unc.edu

ting the object's surface. We show that although a spring-mass system is a coarser approximation than FEM models used in prior approaches [Chaigne and Doutaut 1997; O'Brien et al. 2001; O'Brien et al. 2002], it is an adequate model to capture the small-scale surface vibrations that lead to the generation of sound in nature. We show how this formulation yields an analytical solution to the equation of motion for the surface of the object.

However, a naive implementation of such an approach can handle only a few (less than ten) sounding objects in real time. We also present several acceleration techniques. The increased computational efficiency is achieved by exploiting auditory perception, which ensures that the resulting degradation in perceived quality is minimal. In addition, the sound quality and the associated computational cost for each object is scaled dynamically in a priority-based scheme which guarantees that the total sound production meets stringent time constraints, while preserving the overall aural experience. Our approach has the following characteristics:

- It is based on a discretized physically-based representation that offers simplicity of formulation and ease of implementation;
- It makes no assumptions about the input mesh topology – surface meshes used for physics can be used directly for sound synthesis;
- It is capable of yielding both impact and rolling sounds naturally, without any special-case treatment;
- It enables rich environments consisting of numerous sounding objects, with insignificant difference in the overall audio quality.

We also use OpenAL and EAX™ to provide hardwareaccelerated propagation modeling of the synthesized sounds on Creative Sound Blaster Audigy 2™ audio cards which easily produce spatial and environmental sound effects such as distance attenuation and room acoustics. To the best of our knowledge, with the possible exception of methods that rely on physical measurements, no prior work has been demonstrated to handle complex scenarios (e.g. see Figs. 1 and 7) in real time.

Organization: The rest of the paper is organized as follows. We review related work in Section 2. We present the mathematical formulation developed to model the surface vibrations for sound synthesis in Section 3 and describe various acceleration techniques to enable real-time sound generation for a large-scale environment consisting of hundreds of sounding objects in Section 4. In Section 5, we discuss implementation issues and demonstrate the results of our system on complex scenes. Finally, we conclude with possible future research directions.

2 Previous Work

The concept of modeling the surface vibrations of objects using discretized physical models in real time was first proposed by Florens and Cadoz [1991], who used a system of masses and damped springs to model 3D shapes and developed the CORDIS-ANIMA system for physically-based sound synthesis. More recently, numerical integration with a finite element approach was proposed as a more accurate technique for modeling vibrations [Chaigne and Doutaut 1997; O'Brien et al. 2001]. These methods had the advantage that the simulation parameters corresponded directly to physically measurable quantities and the results were more accurate. The main drawback was the complexity of formulation and implementation and the low speed of the resulting simulation.

To remedy the performance issue of the above methods, van den Doel and Pai suggested [1996; 1998] using the analytically computed vibrational modes of an object, instead of numerical integration, leading to considerable speedups and enabling real-time sound synthesis. But, since the PDEs governing the vibration of arbitrary shapes are very complicated, the proposed system could only handle simple systems, such as plates, for which the analytical solutions were known. To handle more complex systems which do not admit direct analytical solution, two approaches have been proposed in literature. The first approach, [van den Doel et al. 2001] uses physical measurements on a given shape to determine its vibration modes and their dependence on the point of impact. Later, these modes may be appropriately mixed in a real-time application to generate realistic synthetic sounds. But, arbitrary 3D models have to be physically procured, in order to find their aural properties. In [2002], O'Brien et al. address this problem and propose a method for handling arbitrarily-shaped objects by discretizing them into tetrahedral volume elements. They show that the corresponding finite element equations can be solved analytically after suitable approximations. Consequently, they are able to model arbitrarily shaped objects and simulate realistic sounds for a few objects at interactive rates.

Our work shares some common themes with [O'Brien et al. 2002]. However, we propose a simpler system of point-masses and damped springs for modeling the surface vibrations of the object and it also submits to an analytical solution in a similar fashion, while offering much greater simplicity of formulation and ease of implementation. Furthermore, the complexity of scenes demonstrated in [O'Brien et al. 2002] is low, containing less than 10 sounding objects and the interactions captured are mainly due to impacts. As we will demonstrate in Section 5, our method extends to handling hundreds of objects in real time and is also capable of producing realistic *rolling* sounds in addition to impact sounds.

Often, immersive environments are both visually and aurally complex. The problem of scheduling multiple objects for sound synthesis was first addressed in [Fouad et

al. 1997]. They exploited a model of imprecise computation proposed previously in [Chung et al. 1987], and proposed a

Thus, the only avenue left is to make suitable discrete approximations of the problem to reduce the PDEs to ODEs,

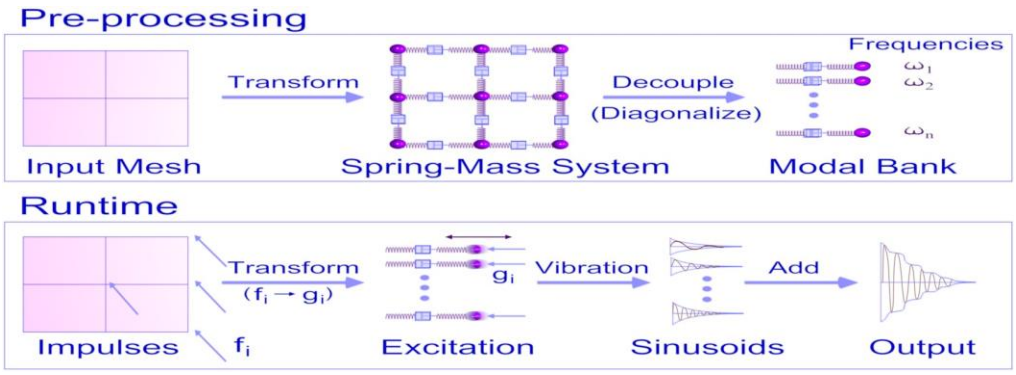

Figure 2: This diagram gives an overview of our approach. In the pre-processing step, each input surface mesh is converted to a spring-mass system by replacing the mesh vertices with point masses and the edges with springs, and the force matrices are diagonalized to yield its characteristic mode frequencies and damping parameters. At runtime, the rigid-body simulator reports the force impulses f_i on a collision event. These are transformed into the mode gains, g_i with which the corresponding modes are excited. These yield damped sinusoids which

system in which the objects are iteratively assigned time quotas depending on the availability of resources and priorities of the objects. As described in Section 4, our approach to prioritization and time-quota assignment exploits properties specific to our sound-generation technique, and thus achieves better results using a much simpler scheme. Recently, van den Doel et al. [2004] proposed techniques to synthesize sounds in real time for scenes with a few sounding rigid bodies and numerous particles, by exploiting frequency masking. At runtime, they find emitted frequencies which are masked out by other neighboring frequencies with higher amplitude and do not mix the masked frequencies. We use a different perceptual observation presented in [Sek and Moore 1995], which report that humans are incapable of distinguishing frequencies that are very close to each other. As we will discuss in Section 4, this can be used to prune out frequencies from an object's frequency spectrum as a *pre-processing* step. Our technique leads to better performance and much lesser memory consumption at runtime while ensuring minimal loss in auditory quality.

3 Methodology

Sound is produced by surface vibrations of an elastic object under an external impulse. These vibrations disturb the surrounding medium to result in a pressure wave which travels outwards from the object. If the frequency of this pressure wave is within the range 20 to 22000 Hz, it is sensed by the ear to give us the subjective perception of sound. The most accurate method for modeling these surface vibrations is to directly apply classical mechanics to the problem, while treating the object as a continuous (as opposed to discrete) entity. This results in PDEs for which analytical solutions are not known for arbitrary shapes. which are more amenable to analysis. In this section, we show how a spring-mass system corresponding to a physical object may be constructed to model its surface deformation and how it may be analyzed to extract the object's modes of vibration. For ease of illustration, we assume a homogeneous object; inhomogeneous objects may be handled by a simple extension of the approach presented here. Further, we assume that the input object is in the form of a thin shell and is hollow inside. This assumption is motivated by practical concerns since most of the geometry today is modeled for rendering and is invariably only a surface representation with no guarantees on surface connectivity. If a volumetric model is available, the approach outlined in this paper applies with minor modifications. Figure 2 gives an overview of our approach.

3.1 Input Processing

Given an input mesh consisting of vertices and edges, we construct an equivalent spring-mass system by replacing the mesh vertices with point masses and the edges with damped springs. We now are suitably combined to yield the output sound signal.

discuss how to assign the spring constants and masses based on the material properties of the object so that the discrete system closely approximates the physical object. The spring constant, k and the particle masses, m_i are given by:

$$\begin{aligned} k &= Yt \\ m_i &= \rho t a_i \end{aligned} \qquad (1)$$

where Y is the Young's Modulus of elasticity for the material, t is the thickness of the object surface, ρ is the material

density and a_i is the area "covered" by a particle, which is calculated by dividing the area of each mesh face equally amongst all its constituent vertices and summing all the face contributions for the vertex corresponding to the mass in consideration. Note that we did not discuss fixing the spring damping parameters above, which we will return to shortly.

3.2 Deformation Modeling

Once the particle system has been constructed as above, we need to solve its equation of motion in order to generate the corresponding sound. Unfortunately, the resulting system of equations is still mathematically complex because the interaction forces between the masses are non-linear in their positions. However, by making the reasonable assumption that the deformation is small and linearizing about the rest positions, this problem can be cast in the form of a coupled linear system of ODEs:

$$M\frac{d^2r}{dt^2} + (\gamma M + \eta K)\frac{dr}{dt} + Kr = f \tag{2}$$

where M is the mass matrix, K is the elastic force matrix, γ and η are the fluid and viscoelastic damping constants for the material respectively. The matrix M is diagonal with entries on the diagonal corresponding to the particle masses, m_i. The elastic force matrix K is real symmetric, with entries relating two particles if and only if they are connected by a spring. The variable r is the displacement vector of the particles with respect to their rest position and f is the force vector. Intuitively, the terms in the above equation correspond to inertia, damping, elasticity and external force respectively. The specific form of damping used above, which expresses the overall damping matrix as a linear combination of K and M is known as Raleigh damping and works well in practice. For a system with N particles in three dimensional space, the dimensions of all the matrices above is $3N \times 3N$.

This formulation of the problem is well known and is similar to the one presented in [O'Brien et al. 2002]. The main difference in our approach is that the force and inertia matrices are assembled from a spring-mass system which makes the formulation much simpler. The solution to Equation (2) can be obtained by diagonalizing K so that:

$$K = GDG^{-1} \tag{3}$$

where G is a real matrix consisting of the eigenvectors of K and D is a diagonal matrix containing the eigenvalues. For reasons we will explain later, we will henceforth call G the "gain matrix". Plugging the above expression for K into Equation (2) and multiplying by G^{-1} throughout, we obtain:

$$G^{-1}M\frac{d^2r}{dt^2} + \left(\gamma G^{-1}M + \eta DG^{-1}\right)\frac{dr}{dt} + DG^{-1}r = f$$

Observing that since M is diagonal, $G^{-1}M = MG^{-1}$ and defining $z = G^{-1}r$ equation(4) may be expressed as:

$$M\frac{d^2z}{dt^2} + (\gamma M + \eta D)\frac{dz}{dt} + Dz = G^{-1}f \tag{5}$$

Since both M and D in the above equation are diagonal, Equation (2) has been decoupled into a set of unrelated differential equations in the variables z_i, which correspond to individual modes of vibration of the object. The equation for each mode is the standard equation of a damped oscillator and has the following solution for the i'th mode:

$$\begin{aligned} z_i(t) &= c_i e^{\omega_i^+ t} + \overline{c_i} e^{\omega_i^- t} \\ \omega_i^{\pm} &= \frac{-(\gamma\lambda_i + \eta) \pm \sqrt{(\gamma\lambda_i + \eta)^2 - 4\lambda_i}}{2} \end{aligned} \tag{6}$$

where the constant c_i, called the gain for the mode, is found by considering the impulses applied as we will discuss shortly. We use c_i to denote the complex conjugate of c_i. The constant λ_i is the i'th eigenvalue in the diagonal matrix, D. The real part of $\omega_i^{\pm}$ gives the damping coefficient for the mode, while the imaginary part, if any, gives the angular frequency of the mode.

3.3 Handling Impulsive Forces

Once an input mesh has been processed as above and the corresponding modes extracted as outlined in Equations (2)-(6), we have all the information needed regarding the aural properties of the object. The sound produced by an object is governed by the magnitude and location of impulsive force on its surface. We model short-duration impulsive contacts by dirac-delta functions. Given an impulse vector f containing the impulses applied to each vertex of an object, we compute the transformed impulse, $g = G^{-2}f$ in order to evaluate the right-hand side of Equation (5). Once this is done, the equation for the i'th mode is given by:

$$m_i\frac{d^2z_i}{dt^2} + (\gamma m_i + \eta\lambda_i)\frac{dz_i}{dt} + \lambda_i z_i = g_i\delta(t - t_0) \tag{7}$$

where t_0 is the time of collision and $\delta()$ is the dirac delta function. Integrating the above equation from a time just before t_0 to a time just after t_0 and noting that $\int_{t_0^-}^{t_0^+}\delta(t - t_0)dt = 1$, we obtain:

$$m_i\Delta\left(\frac{dz_i}{dt}\right) + (\gamma m_i + \eta\lambda_i)\Delta z_i + z_i\Delta t = g_i \tag{8}$$

Assuming that Δt is very small, and using the fact that the deformation is small compared to the change in velocities,

[2] The constant of proportionality is determined based on the geometry of the object and takes the fact into account that vibrations in the direction of the surface normal contribute more to the resulting pressure wave than vibrations perpendicular to the normal. We do not describe it in detail here as it is not critical to the approach being presented.

we can neglect the last two terms on the left-hand side to obtain:

$$\Delta\left(\frac{dz_i}{dt}\right) = \frac{g_i}{m_i} \tag{9}$$

The above gives a very simple rule which relates the change in the time derivative of the mode to the transformed impulse. Referring to Equation (6) and requiring that z_i should stay the same just before and after the collision while $\frac{dz_i}{dt}$ should increase as in Equation (9), the update rule for the mode gain c_i can be shown to be:

$$c_i \leftarrow c_i + \frac{g_i}{m_i\left(\omega_i^+ - \omega_i^-\right)e^{\omega_i^+ t_0}}. \tag{10}$$

Initially, c_i is set to 0 for all modes.

4 Real-time Sound Synthesis

In this section, we describe how the mathematical formulation presented in the previous section is utilized to efficiently generate sound in real time. First, we describe a naive implementation and then discuss techniques to increase its efficiency.

Assume that there exists a rigid-body simulator which can handle all the dynamics. During a collision event, the sound system is informed of the object that undergoes the impact and the magnitude and location of impact. This impact is processed as described in Section 3.3 to result in the gains for the different modes of vibration of the object, where the gain for the *i*'th mode being c_i. The equation for a mode from the time of collision onwards is given by (6). The amplitude contribution of a mode at any moment is proportional[1] to its *velocity* (and not position). This is because the pressure contribution of a particle is determined by its velocity and the mode velocities are linearly related to the physical velocities of the particles. The mode velocity is found by taking a differential of Equation (6) with respect to time:

$$v_i = \frac{dz_i}{dt} = c_i\omega_i^+ e^{\omega_i^+ t} + \overline{c_i}\omega_i^- e^{\omega_i^- t} \tag{11}$$

For generating each audio sample, we need to evaluate the above equation for all vibration modes of the object, which is quite inefficient. As mentioned in [O'Brien et al. 2002], the simple observation that $e^{i\omega(t+\Delta t)} = e^{i\omega t}e^{i\omega\Delta t}$ offers some gain in performance since generating a new audio sample just requires a single complex multiply with the previous value. However, the efficiency is still not sufficient to handle a large number of objects in real time. We

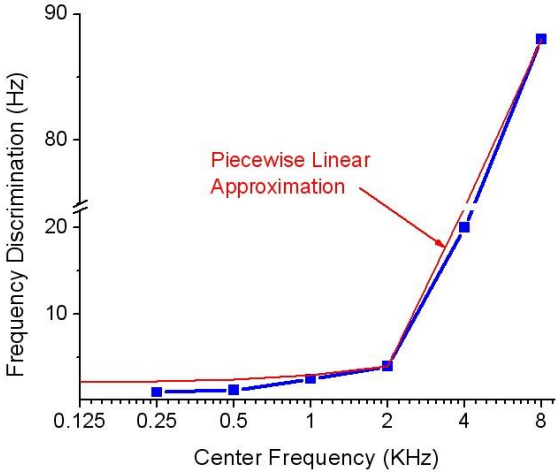

Figure 3: This plot shows frequency discrimination in humans as a function of the center frequency. Note that the human capacity to discriminate between frequencies degrades considerably for frequencies in the range 2-22 KHz, which forms a bulk of the human audible range. We use this fact to guarantee that no more than 1000 modes need to be mixed for any object in the worst case, *irrespective* of its geometric complexity. In most cases the actual number is much smaller, in the range of a few hundreds. The red curve shows the piecewise linear approximation of this curve that we use.

may estimate the running time of the system as follows: A simple spring-mass system with N particles has $3N$ modes, and the above operation needs to be repeated for each mode for each audio sample. Assuming a sampling rate of 44100 Hz, the number of floating-point operations (FLOPS) needed for this calculation for generating audio samples worth t seconds is:

$$T = 3N \times 4 \times 44100t\ FLOPS. \tag{12}$$

Considering that the typical value of N is about 5000 or higher, producing sound worth 1 second would take 2646 MFLOPS. Since today's fastest processors operate at a few thousand MFLOPS [Dongarra 2005], the above processing would take about a second. Given that this estimated amount of time is for just one object and a typical scene may contain many such objects, such an approach is clearly not fast enough for interactive applications. Furthermore, for many real-time environments such as games and virtual environments, only a very small fraction of the actual time can be allocated for sound production. Thus, in the rest of this section, we will discuss techniques to increase the efficiency of the proposed base system to enhance its capability in handling scenarios with a large number of sounding objects at interactive rates.

From Equation (12), it is clear that the running time is proportional to the number of modes being mixed and the number of objects. Next, we present acceleration

techniques for sound simulation by reducing the number of modes per object: “Mode Compression” and “Mode Truncation”, and by scaling the audio quality of different objects dynamically with little degradation in perceived sound quality.

4.1 Mode Compression

Humans have a limited range of frequency perception, ranging from 20 to 22000 Hz. It immediately follows that modes with frequencies lying outside this range can be clipped out and need not be mixed. However, there is another important fact which can lead to large reductions in the number of modes to be mixed. A perceptual study described in [Sek and Moore 1995] shows that humans have a limited capacity to discriminate between nearby frequencies. Note that this is different from frequency masking [Zwicker and Fastl 1990] in which one of two *simultaneously* played frequencies masks out the other. Rather, this result reports that even if two “close enough” frequencies are played in succession, the listener is unable to tell whether they were two different frequencies or the same frequency played out twice. The authors call the length of the interval of frequencies around a center frequency which sound the same, the “Difference Limens to Change” (DLC). Figure 3 shows a plot of the DLC against center frequencies ranging from .25 to 8 KHz. Interestingly, the DLC shows a large variation over the audible frequency range, getting very large as the center frequency goes beyond 2 KHz. Even at 2 KHz, the DLC is more than 1 Hz. That is, a human subject cannot tell apart 1999 Hz from 2000 Hz.

We use the above fact to drastically reduce the number of modes that are mixed for an object. We linearly approximate the DLC curve with a piecewise linear curve shown as the red line in Figure 3. The approximation has two segments: one from 20 Hz to 2 KHz and another from 2 KHz to 22 KHz. As we show in the figure we overestimate the DLC slightly. This increases the performance further and we have observed minimal loss in quality in all the cases we have tested. The main idea behind our compression scheme is to group together all the modes with perceptually indistinguishable frequencies. It can be easily shown that if the above mentioned linear approximation to the DLC curve is used and indistinguishable modes clustered at the corresponding frequency centers, the maximum number of modes that need to be mixed is less than 1000. It is important to note that this is just the worst case scenario and it happens only when the frequency spectrum of the object consists of all frequencies from 20 to 22,000 Hz, which is very rare. For most objects, the frequency spectrum is discrete and consequently, the number of modes after mode compression is much smaller than 1000, typically in the range of a few hundreds.

We now describe the details of our technique. Recall the gain matrix from Equation (3), G. The gain matrix has a very simple physical interpretation: Rows of the matrix correspond to vertices of the object and columns correspond to the different modes of vibration (with their corresponding frequencies). Each row of G lists the gains for the various modes of vibration of the object, when a unit impulse is applied on the corresponding vertex. It is clear from the above discussion that all the mode gains within a row of G which correspond to modes with close frequencies need to be clustered together. This is achieved by replacing the gain entries for all such modes by a single entry with gain equal to the sum of the constituent gains. Since a mode corresponds to a whole column, this reduces to summing together columns element-wise based on their frequencies. The complete procedure is as follows:

- Sort the columns of G with the corresponding mode frequencies as the key. [3]
- Traverse the modes in increasing order of frequency. Estimate the DLC, Δ at the current frequency using the piecewise linear curve shown in Figure 3. If the current frequency and next frequency are within Δ of each other the two mode frequencies are indistinguishable, replace the two columns by their element-wise sum.

Below, we enumerate the main advantages of this scheme:

1. The running time is constant instead of linear in the numberof vertices in the object. For example, if the input mesh is complex with 5,000 vertices, the number of modes mixed is bounded by 1000 instead of the earlier $3N = 15{,}000$ which is a substantial performance gain.

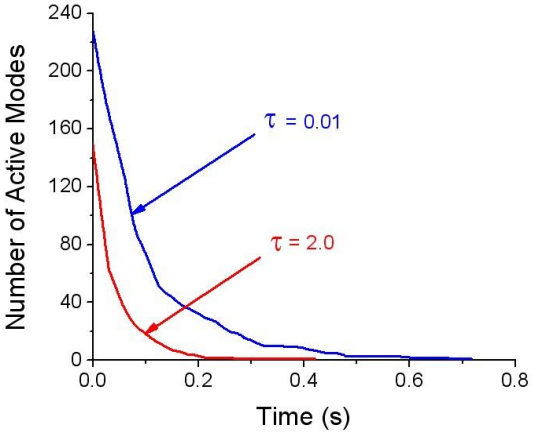

Figure 4: This graph shows the number of modes mixed vs time, for a xylophone bar just after it is struck in the middle. τ is the mode truncation threshold. A higher value of τ leads

[3] This step is usually not needed as most linear algebra packages output the eigenvector matrix sorted on the eigenvalues

to more aggressive truncation of modes with low amplitude, leading to savings in terms of the number of modes mixed. In this case, τ = 2.0 results in about 30% gain in efficiency over τ = 0.01 which only truncates modes with near-zero amplitude. The sound quality for both the cases is nearly identical.

2. Since this scheme requires just the frequencies of the differentmodes, the whole processing can be done as a pre-process without requiring any extra runtime CPU cycles.

3. From the above mentioned procedure, it is clear that the number of columns in the matrix G, which is the same as the number of modes, is now bounded by 1000 instead of the earlier value of $3N$. Since this matrix needs to be present in memory at runtime for transforming impulses to mode gains, its memory consumption is an important issue. Using this technique, for an object with 5000 vertices, the memory requirement has been reduced from 225 MB to less than 15 MB, by more than a factor of 15.

4. Most objects have a discrete frequency spectrum with possibly many degenerate frequencies. Due to numerical inaccuracies while diagonalizing the elastic force matrix and the approximations introduced by the spring-mass discretization, these degenerate frequencies may appear as spurious distinct modes with near-equal frequencies. Obviously, it is wasteful to treat these as distinct modes. It is our observation that most of the times these modes' frequencies are close enough so that they are naturally summed together in this scheme.

4.2 Mode Truncation

The sound of a typical object on being struck consists of a transient response composed of a blend of high frequencies, followed by a set of lower frequencies with low amplitude. The transient attack is essential to the quality of sound as it is perceived as the characteristic "timbre" of the object. The idea behind mode truncation is to stop mixing a mode as soon as its contribution to the total sound falls below a certain preset threshold, τ. Since mode truncation preserves the initial transient response of the object when τ is suitably set, the resulting degradation in quality is minimal. Figure 4 shows a plot of the number of active modes with respect to time for a xylophone bar struck in the middle for two different values of τ: .01 and 2. These values are normalized with respect to the maximum sample value which is 65536 for 16-bit audio. The first case with τ = .01 performs essentially no truncation, only deactivating those modes which have near-zero amplitude. Note that with τ = 2 the number of modes mixed is reduced by more than 30%. Also, the number of active modes floors off much earlier (.2 secs compared to .6 secs). It is important to note that this results in little perceptible loss in quality.

The details of the technique are as follows: Assume that an object has just undergone a collision and the resulting mode gains c_i have been calculated as given by Equation (10). From this time onwards until the object undergoes another collision, Equation (11) gives a closed-form expression for the mode's contribution to the sound of the object. This can be used to predict exactly when the mode's contribution drops below the threshold τ. The required "cutoff time", t^c_i is such that for all times $t > t^c_i$:

$$c_i\omega_i^+ e^{\omega_i^+ t} + \overline{c_i}\omega_i^- e^{\omega_i^- t} < \tau \tag{13}$$

Using the fact that for any two complex numbers x and y, $|x + y| \leq |x| + |y|$ it can be shown that,

$$t_i^c \leq \frac{1}{-Re(\omega_i^+)} \ln\left(\frac{2\,|c_i|\,|\omega_i^+|}{\tau}\right) \tag{14}$$

Using the above inequality, the cutoff times are calculated for all the modes just after a collision happens. While generating the sound samples from a mode, only one floating point comparison is needed to test if the current time exceeds the cutoff time for the mode. In case it does, the mode's contribution lies below τ and consequently, it is not evaluated.

4.3 Quality Scaling

The two techniques discussed above are aimed at increasing the efficiency of sound synthesis for a single object. However, when the number of sounding objects in a scene grows beyond a few tens, this approach is not efficient enough to work in real time and it is not possible to output the sound for all the objects at the highest quality. It is critical in most interactive applications that the sound system have a graceful way of varying quality in response to variable time constraints. We achieve this flexibility by scaling the sound quality for the objects. The sound quality of an object is changed by controlling the number of modes being mixed for synthesizing its sound. In most cases of scenes with many sounding objects, the user's attention is on the objects in the "foreground", that is, the objects which contribute the most to the total sound in terms of amplitude. Therefore, if it is ensured that the foreground sounds are mixed at high quality while the background sounds are mixed at a relatively lower quality, the resulting degradation in perceived aural quality should be reduced.

We use a simple scheme to ensure higher quality for the foreground sounds. At the end of each video frame, we store the sum of the vibrational amplitudes of all modes for each object, which serve to determine the object's priority. At the next video frame, all objects are sorted in decreasing order based on their priority and the total time-quota for sound-generation divided among the objects as a linearly decreasing ramp with a preset slope, S. After this, all objects are processed in their priority order. For each object, its quality is first scaled so that it can finish within its assigned time-quota and then the required modes are mixed for the

given time period. If an object finishes before its time-quota has expired, the surplus is consumed greedily by the next higher priority object. The slope, S of the ramp decides the degree to which the foreground sound quality is favored over a degradation in background sound quality. The case with $S = 0$ corresponds to no priority scheduling at all, with the time-quota being divided equally among all objects. The converse case with $S = \infty$ corresponds to greedy consumption of the time-quota. That is, the whole time-quota is assigned to the highest priority object. After the object is done, the remaining time, if any, is assigned to the next highest priority object and so on.

4.4 Putting Everything Together

To illustrate how all the techniques described above are integrated, we present a summary of our approach.

Pre-processing

- Construct a spring-mass system corresponding to the input mesh. (Section 3.1)
- Process the spring-mass system to extract the gain matrix, G and the (complex) angular frequencies of the object's modes of vibration: ω_i^+ and ω_i^-. (Section 3.2, Eqns. (3) and (6))
- Mode Compression:
  Aggregate columns of G based on frequencies of the corresponding modes, as described in Section 4.1.
- Store the resulting gain matrix along with the (complex) constants ω_i^+ and ω_i^- for modes correspond to the columns of G after compression. Note that ω_i^- need not be stored in case ω_i^+ has a non-zero imaginary part since in that case
  $\omega_{i-} = \omega_{i+}$.

Runtime Processing

- Load the gain matrix and mode data for each object.
- Begin simulation loop:
  1. Run rigid-body simulation
  2. For each object, O:
     – Collision Handling:
       If the rigid-body simulator reports that O undergoes a collision event, update its gain coefficients as per Equation (10) using the collision impulse and its location. (Section 3.3)
     – Mode Truncation:
       Compute cutoff times t_j^c for each mode based on the mode truncation threshold, τ. (Section 4.2, Equation (14))
  3. Quality Scaling:
     Sort objects based on amplitude contribution, assign time-quotas and compute the number of modes to be mixed for each object. (Section 4.3)
  4. Sound Synthesis:
     For each timestep at time t and for each object, O:
     – Consider all modes permitted by the current quality setting which satisfy $t_j^c > t$. Sample and summate all these modes as described at the beginning of this section. This is O's contribution to the sound.
     – Output the sum of all objects' sample contribution as the sound sample for time t.

End simulation loop

5 Implementation and Results

In this section we present results to demonstrate the efficiency and realism achievable with our approach.

5.1 Rigid Body Simulation

We have implemented the algorithm and acceleration techniques presented in this paper using C++ and OpenGL. Our rigid-body simulator extends the technique presented by Guendelman et al. [2003] to incorporate DEEP [Kim et al. 2002] for fast and more accurate penetration depth estimation, instead of sample-based estimation using distance fields. It also uses a more complex friction model presented by Mirtich and Canny [Mirtich and Canny 1995], which results in more robust contact resolution.

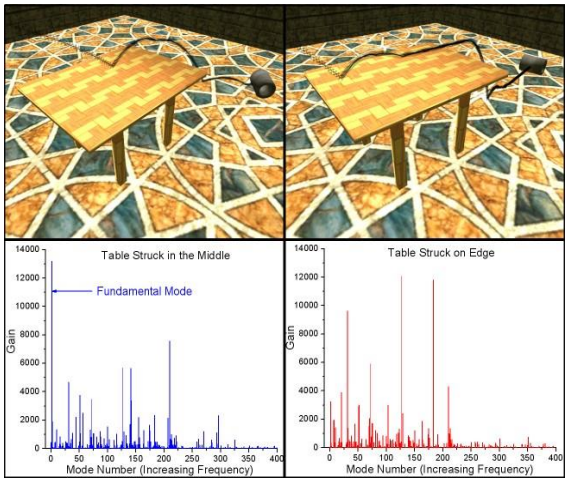

Figure 5: A metallic cylinder falls onto a wooden table, in the middle (left) and on the edge (right) and rolls off. The bottom part shows the corresponding frequency spectra for the two cases. Note that for the case on the left, most of the impulse is transferred to the low frequency fundamental mode while for the case on the right, the impulse is mostly transferred to higher frequency modes.

5.2 Position Dependent Sounds

As discussed earlier, the main advantage of using physically-based sounds over recorded audio is the ability to capture effects such as the magnitude of impacts between objects and more importantly, the subtle shift in sounds on striking an object at different points. Figure 5 shows a scene with a metallic cylinder tossed onto a wooden table. Both the table and cylinder are sounding. The figure contrasts two cases: the first case, shown on the left, depicts the cylinder striking the table near the middle and rolling off, while in the second case it strikes the table near the edge. We discuss the rolling sound in the next subsection, and will discuss the impact sound here. Since the table-top is in the form of a plate, we would expect that striking it on the edge would transfer a larger fraction of the impulse to higher frequencies, while striking it in the middle should transfer most part of the impulse to the fundamental mode of vibration, leading to a deeper sound. To verify this, we plotted the frequency spectra for the two cases just after the cylinder makes first impact with the table. The corresponding plots for the two cases are shown in the lower part of the figure. The case on the left shows a marked peak near the fundamental mode while the peak is completely missing in the second case. Conversely, the second case shows many peaks at higher frequencies which are missing in the first one. This difference clearly demonstrates that the sound for the two cases is markedly different, with the same qualitative characteristics as expected. Another important point to note is that this technique does not require the meshes to be highly tessellated to capture these effects. The table consists of just 600 vertices and the cylinder 128 vertices.

5.3 Rolling Sounds

In addition to handling impact sounds, we are able to simulate realistic rolling sounds without requiring any special-case treatment for sound synthesis. This is in part made possible because of the rigid-body simulator we have developed, which is able to handle contacts in a more graceful manner than most impulse-based simulators. Figure 6 shows the impulses on the cylinder and the corresponding audio for the case shown in the right side of Figure 5. The cylinder rolls on the table after impact, falls to the ground and rolls on the floor for sometime. The initial rolling sound, when the cylinder is on the table, has a much richer quality. The sound of the

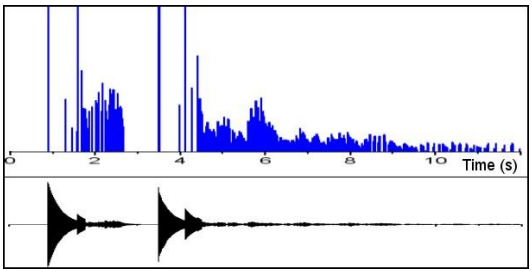

Figure 6: A plot of the impulses on a cylinder versus time for the scene shown on the right in Figure 5 and the corresponding audio samples. The peaks correspond to impacts while the numerous lowamplitude impulses correspond to rolling forces.

table as the cylinder rolls over it conveys a sense of the cylinder's heaviness, which is only partly conveyed by the sound of the impact. The cylinder, although uniformly tessellated, is very coarse, with only 32 circumferential subdivisions. Figure 6 shows the impulses applied on the cylinder against time. The peaks correspond to impacts: when the cylinder falls on the table, and when it falls to the ground from the table. Note that the audio waveform shows the corresponding peaks correctly. The period of time stretching from 6 to 8 seconds consists of the cylinder rolling on the floor and is characterized by many closely-spaced small-magnitude impulses on the cylinder as it strikes the floor again and again due to its tessellation. To test how important the periodicity of these impulses was for the realism of rolling sounds, we found the mean and standard deviation of the interval between these impulses from the data presented in Figure 6. The mean time between the impulses was 17 ms with a standard deviation of 10 ms. The fact that the standard deviation is more than 50% of the mean demonstrates that the impulses show very little periodicity. This suggests that the periodicity of collisions is not critical for the perceived realism of rolling sounds.

5.4 Efficiency

We are able to do audio simulation for complex scenes in real time using our approach. Figure 7 shows a scene with 100 metallic rings falling simultaneously onto a wooden table and undergoing elastic collision. All the rings and the table are sounding. Each ring is treated as a separate object with separate aural properties. The rings consist of 200 vertices each. Figure 8 shows the audio FPS [4] for this simulation against time for the one second interval during which almost *all* the collisions take place. The application frame rate is 100 FPS. Note that this is not the raw data but a moving average so that the short-range fluctuations are absorbed. The plot on the bottom is the base timing

[4] An audio frame is defined as the amount of sound data sufficient to play for a duration equal to the duration of one video frame.

without using any of the acceleration techniques described in Section 4. The audio in this case is very choppy since the audio generation is not able to keep up with the speed of rendering and rigid-body simulation. With mode truncation and mode compression, the performance shows significant improvement. However, after initially starting at about 200 FPS, the frame rate drops in the latter part where the maximum number of collisions happen. With quality scaling in addition to mode compression and truncation (shown by the top-most curve), the frame rate exhibits no such drop, continuing to be around 200 FPS. This is because quality scaling gives priority to sound generation for those rings which just underwent collision, while lowering the quality for other rings which may have collided earlier and are contributing less to the overall sound. This illustrates the importance of quality scaling for scenarios with multiple collisions. It is important to note that although this example sufficiently demonstrates the capability

To illustrate the realistic sounds achievable with our approach, we designed a three-octave xylophone shown in Figure 1. The image shows many dice falling onto the keys of the xylophone to produce the corresponding musical notes. The audio simulation for this scene runs in the range of 500-700 FPS, depending on the frequency of collisions. The dice have been scripted to fall onto the xylophone keys at precise moments in time to play out any set of musical notes. Because of the efficiency of the sound generation process, the overall system is easily able to maintain a steady frame rate of 100 FPS. Also, there are situations in which many dice fall on different keys within a few

milliseconds of each other, but the sound quality exhibits no perceptible degradation. Although we have not tuned

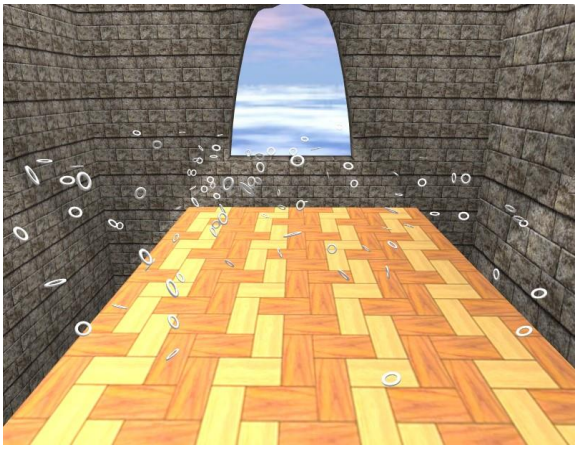

Figure 7: More than 100 metallic rings fall onto a wooden table. All the rings and the table are sounding. The audio simulation runs at more than 200 FPS, the application frame rate being 100 FPS. Quality Scaling ensures that the perceived sound quality does not degrade, while ensuring steady frame rates (See Figure 8) of the system to maintain steady frame rates, it is improbable in a real application, since there are about 100 collisions within a second. This is the reason why the CPU utilization is high (50%). A more common scenario would be as shown in Figure 1, which has a much lower CPU utilization (10%).

the xylophone keys to match the exact frequency spectrum of a real xylophone, the resulting sound is realistic and captures the essential timbre of the instrument. The material parameters for the xylophone were taken from [Chaigne and Doutaut 1997].

5 Conclusion

We have presented a physically-based sound synthesis algorithm with several acceleration techniques for rendering a large-scale scene consisting of hundreds of interacting objects in real time, with little loss in perceived sound quality. This approach requires no special mesh structure, is simple to implement, and further takes advantage of existing hardware acceleration. We plan to extend this framework to auditory display of sliding sounds, explosion noises, breaking sounds, and other more complex audio effects that are difficult to simulate at interactive rates.

References

CHAIGNE, A., AND DOUTAUT, V. 1997. Numerical simulations of xylophones. i. time domain modeling of the vibrating bars. *J. Acoust. Soc. Am. 101*, 1, 539–557.

CHUNG, J. Y., LIU, J., AND LIN, K. J. 1987. Scheduling real-time, periodic jobs using imprecise results. In *Proc. IEEE RTS*.

DONGARRA, J. J. 2005. Performance of various computers using standard linear equations software (linpack benchmark report). Tech. rep., Knoxville, TN, USA.

FLORENS, J. L., AND CADOZ, C. 1991. The physical model: modeling and simulating the instrumental universe. In *Represenations of Musical*

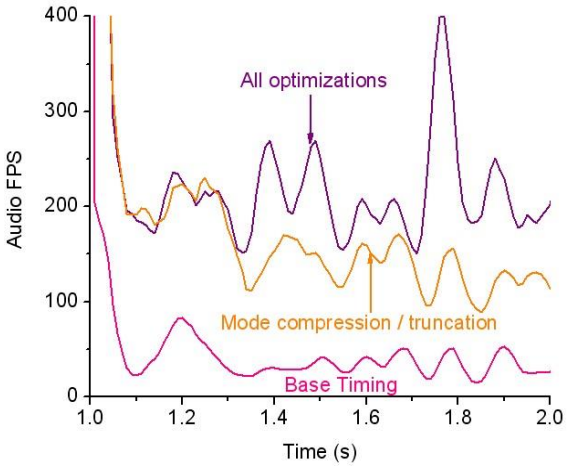

Figure 8: This graph shows the audio simulation FPS for the scene shown in Figure 7 from time 1s to 2s, during which almost all the collisions take place. The bottom-most plot shows the FPS for an implementation using none of the acceleration techniques. The topmost curve shows the FPS with mode compression, mode truncation and quality scaling. Note how the FPS stays near 200 even when the other two curves dip due to numerous collisions during 1.5-2.0s.

FOUAD, H., BALLAS, J., AND HAHN, J. 1997. Perceptually based scheduling algorithms for real-time synthesis of complex sonic environments. In *Proc. Int. Conf. Auditory Display*.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex rigid bodies with stacking. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH) 22*, 871–878.

KIM, Y. J., LIN, M. C., AND MANOCHA, D. 2002. DEEP: an incremental algorithm for penetration depth computation between convex polytopes. *Proc. of IEEE Conference on Robotics and Automation*, 921–926.

MIRTICH, B., AND CANNY, J. 1995. Impulse-based simulation of rigid bodies. In *1995 Symposium on Interactive 3D Graphics*, P. Hanrahan and J. Winget, Eds., ACM SIGGRAPH, 181–188. ISBN 0-89791-736-7.

O'BRIEN, J. F., COOK, P. R., AND ESSL, G. 2001. Synthesizing sounds from physically based motion. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 529–536.

O'BRIEN, J. F., SHEN, C., AND GATCHALIAN, C. M. 2002. Synthesizing sounds from rigid-body simulations. In *The ACM SIGGRAPH 2002 Symposium on Computer Animation*, ACM Press, 175–181.

SEK, A., AND MOORE, B. C. 1995. Frequency discrimination as a function of frequency, measured in several ways. *J. Acoust. Soc. Am. 97*, 4 (April), 2479–2486.

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 Displays*.

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. Foleyautomatic: physicallybased sound effects for interactive simulation and animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 537–544.

VAN DEN DOEL, K., KNOTT, D., AND PAI, D. K. 2004. Interactive simulation of complex audiovisual scenes. *Presence: Teleoper. Virtual Environ. 13*, 1, 99–111.

ZWICKER, E., AND FASTL, H. 1990. In *Psychoacoustics*. Springer-Verlag, Berlin.

Signals, G. D. Poli, A. Piccialli, and C. Roads, Eds. MIT Press, Cambridge, MA, USA, 227–268.

大型场景交互声音合成

摘要

我们提出了一种交互式方法，用于从刚体动态模拟中生成基于物理性质的声音。我们使用了弹簧质点系统来模拟每个物体的局部变形和震动。这被证明是一个对捕捉一些像撞击力大小，撞击位置和滚动声音等物理效应的合适方法。该方法不需要对物体网格的连结性获拓扑结构进行假设，用于于刚体动态模拟的物体表面网格可以直接用于声音的合成。同时，我们使用了基于听觉感知和基于优先级队列的声音质量压缩方法，是的该声音合成方法能够满足实时条件的严格时间限制，同时确保对听者感知到的声音音质降低的程度最小。通过这种方法，与没有实施加速方案相比，我们观察到了高达一个数量级的加速，因此我们能够以 100 帧/秒的速度在交互系统中模拟上百个发声物体，使得该方法非常适合游戏和虚拟环境等交互式应用。此外，我们在 Creative Sound Blaster Audigy 2TM 卡上使用 OpenAL 和 EAXTM，对合成声音进行快速硬件加速传播建模。

关键词：声音合成，刚体模拟，OPENAL

1. 引言

大多数如今的交互应用使用的是已经记录的声音片段来提供对应场景中物体交互的声音。尽管这种方法对声音实时性和声音生成过程是非常快的，但这有有很多物理效果很难被这种技术保留住，比如再一个典型碰撞场景中，声音大小和声音的音质是被碰撞力的位置和大小所控制的 – 当敲击盘子的边缘和中心，一个盘子的声音听起来是非常不同的。总的来说，如果一个碰撞场景有轻微的改变，那么对应展现出俩的声音也应该同样随之变化。这样细微的变化能够给这样的场景增添添加很明显的真实感且避免了重复的已记录声音片段的使用。然后，开发这样的声音系统需要基于物理规则面临着一些实时性处理的困难。对这样的系统最直接的要求就是需要一个高效的动态引擎来对物体碰撞的声音系统和涉及的力进行信息交互。对于这个工作而言，我们开发出了比较搞笑且稳健的刚体模拟器，但是现如今的游戏引擎已经可以满足动态

引擎的高效要求。在这样引擎的支持下，主要的挑战就是需要高效的实时合成声音且同时只占用小部分的运行时间。大多数情况下，声音系统只能对每一个物体的每一个声音占用几百个CPU运行周期来满足交互引用的实时性需求。

主要结果：在这篇文章中，我们提出了一种满足实时交互性能要求，能够保证声音高真实和精确度的方法。在获取物体几何信息和一些材质参数的情况下，我们建立了弹簧模型来近似物体表面信息。这样的方法表明尽管弹簧模型与有限元分析模型而言是一个比较粗糙的近似方法[Chaigne and Doutaut 1997; O'Brien et al. 2001; O'Brien et al. 2002]，这同样是一个合适的模型来抓住物体表面微小的波动来产生自然的声音。我们同样展现了这样的方法是如何产生对于物体表面波动的解析解的。

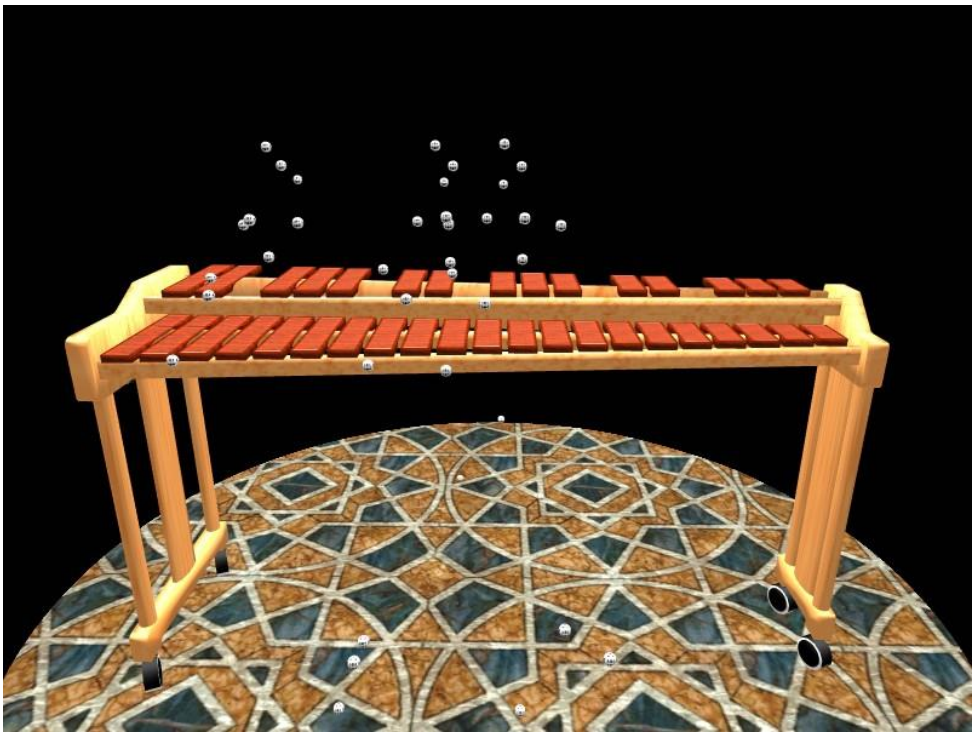

图1：许多骰子连续掉在24音阶的木琴来播放歌曲”取悦者”。对于这个复杂场景，我们的算法能够产生在以最高500FPS下的声音且只占用了3.4GHz Pentium-4 笔记本电脑CPU周期的百分之十

然而，一个比较简易的对上述方法的实现仅仅能够解决小于十个物体的实时模拟。

我们同样提出了一些加速方法。这种加速的计算高效性通过利用听觉感知来实现。它确保了对听觉认知层面的声音质量降低尽量小。同时，声音的质量和对应的每个物体的计算资源消耗是被动态地利用优先级进行了分配，这保证了总共的声音产生在合理的实时声音限制下，且同时保证了合理的听觉体验。我们的方法主要有如下的几个重要特征：

1.该方法基于离散的物理表示，它提供了比较简化的公式和实现。

2.该方法对输入物体网格拓扑结构没有更多要求，物体表面的网格可以直接用于物理建模并直接用于声音生成。

3.该方法能够生产比较自然的碰撞和滚动声音，且不需要特殊的处理。

4.该方法可以生成很多有多种发生物体存在的模拟环境且没有太大的声音质量区别。

我们也使用了OpenAL and EAX™来提供基于Creative Sound Blaster Audigy 2™声卡硬件件加速的建模，使得可以比较轻易的生产空间和环境音效：如距离的衰减和房间环境声效。基于我们现有知识，即使有一些方法使用了同样的物理生成声音的方法，之前也没有工作被用于来对实时的复杂场景进行声音生成(见图1，图5)。

文章架构:本文剩下的部分按照如下结构进行叙述。我们首先回顾了在第二节回顾了相关工作。然后在第三节提出了相应数学公式来对为声音产生的刚体表面波动进行建模，同时描述了各种加速技术来使得实时声音的生成能够适配由成百上千的发声物体组成的大型环境。在第五节，我们讨论了实现遇到的问题和对在复杂环境中发生系统的结果展示。最后我们总结了我们提出的方法和一些对未来研究方向的展望。

2. 相关背景与前述工作

使用离散物理建模的方法对物体表面震动实时建模的概念最开始被Flo- rens and Cadoz [1991]提出。他们使用弹簧质点阻尼模型来对3D形状进行建模且开发出了对应的基于物理的声音生成系统: CORDIS-ANIMA。近年来，更多有限元和数值积分的方法被提出作为一种更准确的对物体表面建模的方法[Chaigne and Doutaut 1997; O’Brien et al. 2001]。这些方法有如下优势:模拟参数与物理参数直接相关且建模

结果更加准确。但主要缺陷是公式和实现的复杂性较高，且模拟的速度较慢。

为了补救上述方法的性能问题。van den Doel and Pai[1996; 1998]建议使用能够被解析计算出的物体震动模式而不是使用数值积分，从而引起了明显的对声音合成的加速使得实时声音合成成了可能。但是，因为表达任意形状震动的偏微分方程是非常复杂的，他们所提出的方法所建立的系统只能处理一些简单的物体清醒，比如盘子等那些解析解已经清楚知道的物体。为了处理一些复杂的，他们的解析解不知道的物体，两种新的方法被提了出来。第一种：[van den Doel et al. 2001]，利用了在物体上物理测量的方法来决定振动模式和物体对点碰撞的反应。然后，这些振动模式被实时应用混合在一起来生成实时的合成声音。但是，任意的3D模型的各种物理参数必须被取得，从而获取产生声音的相关性质。[2002], O’Brien et al他们提出了另外一种解决这个问题的方法来处理任意形状的物体：通过把物体离散化为四面体的组成部分，他们能够在进行近似化处理后，利用对应的有限元分析方法找到解析解。最终他们能够对任意物体进行声音建模且对一些物体能够进行实时的声音合成。

本工作分享了一些和[O’Brien et al. 2002]同样的内容。然而，我们提出了一种相似的质点阻尼系统来对物体表面震动进行建模，且同时提出了一种和上述工作相似的解析解形式。但本工作提供了更简化的公式和实现。更进一步而言，在[O’Brien et al. 2002]中场景的复杂度是比较低的，只包含了少于10个发声物体，而且交互的抓取也主要是通过碰撞进行的，只有碰撞声音。本工作在第五节进行了阐释，本方法能够实时处理成百上千的发生物体而且除了碰撞声音也能够生成实时的滚动声音。

通常而言，沉浸式环境在视觉和听觉的模拟上都比较复杂。这个针对多个物体的声音合成最开始被[Fouad et al. 1997]解决。他们利用了[Chung et al. 1987]的不准确版本的计算模型，然后提出了一个所有被模拟的物体都被迭代式的分配了取决于可用资源和优先级的时间配额。正如第四节所说，我们的方法是也是基于优先级和时间配额来使用了具体对于声音生成技术的资源调度方式且实现了更好的结果。近来的研究如van den Doel et al. [2004]提出了对于实时性场景中刚体和粒子而言合成声音的技术：通过频率掩盖的形式实现。在实时模拟中，他们发现被发出的频率是会被临近的声音大小更大的频率所掩盖的且不会与被掩盖的频率混合。我们利用了[Sek

and Moore 1995]中与上述不同的感知观察方法得出了人对临近频率很难区分的结论。正如我们在第四节所说的，这可以被用于从预先得到的物体的频率光谱中剔出一些没用的频率且可以作为预处理步骤。我们的技术在实时模拟中形成了一个更好的性能且更小的内存占用，且保证最小化的削减了声音的质量。

3. 方法导论

声音是在外力条件下通过弹性物体表面震动发出的。这些震动扰乱了周围的介质从而引起了从物体本身发出的压力波。如果这种压力波的频率是在20到22000赫兹之间，它就能够被人的耳朵听见，从而给我们声音的感知体验。最准确的建模这些表面震动的方法是直接应用经典的力学模型，它认为物体是一个连续介质模型。这样的模型所建立的偏微分方程的解析解对于任意形状物体是未知的。因此，唯一的可适用的方法就是使用合理的离散近似方法来使得整个问题从偏微分方程变成常微分方程，这样回非常利于分析。在这一节，本工作讲具体阐释一个对于物体物理性质的弹簧质点系统是如何与物体表面形变建模的，以及是如何提取物体振动模式的。简而言之，我们认为都是同质对象(每点的物理性质相同)，不同质对象可以利用这里的方法进行延展。进一步而言，我们认为输入的物体是一个薄壳内部是空心的。这样的假设是被现实问题所激发得出的，因为如今大多数的几何结构都是为渲染而建模，只具有物体表面特征而不具有表面连结性。如果一个全体积模型可以被用于上述建模，只需要一点修改，本文的方法也可以被使用来合成声音。图二阐述了本方法的大致流程。

3.1 输入处理

在给予一个由顶点和边组成的网格输入时，我们能够建立起一个对应的弹簧质点系统：用质点代替网格顶点，边代替阻尼弹簧。我们现在讨论如何基于物体材料性质赋值弹簧的参数和质量是的离散化的方法可以近似这样的物理物体。弹簧参数k和质点质量m由如下方程给出:

$$\begin{aligned} k &= Yt \\ m_i &= \rho t a_i \end{aligned} \tag{1}$$

其中Y是弹性材质的杨氏模量，t是物体表面的厚度，是材料密度，是一个质点覆盖的

面积，计算方法是：每个网格面将所在顶点的面积平分，然后考虑对某个顶点所有的有贡献的面涉及的质量加起来。需要注意的是我们这里没有讨论弹簧阻尼的参数，我们将在后节讨论。

3.2 形变建模

一旦上述质点系统已经按照如上方法建立，我们需要解决这个运动方程来产生相应的声音。但是，由于在质点之间的碰撞力不是线性的，造成目前系统的方程让然是数学层面上非常复杂的。然而，我们可以做一些有理由的假设：形变是非常小的而且我们可以根据针对平衡位置进行线性化。这个问题可以被投影成一系列常微分方程的线性系统：

$$M\frac{d^2r}{dt^2}+(\gamma M+\eta K)\frac{dr}{dt}+Kr=f \tag{2}$$

M 是质量矩阵，K是弹性力矩阵，γ和η是对应的材料衰减系数，使用的是Rayleigh衰减模型。这里的M是对角矩阵，每一个值都对应的是质点的质量m,弹性力矩阵K是对称矩阵，对应的两　　个值如果存在一定是他们被一个弹簧相连。变量r是粒子偏离向量(相对于平衡位置)，f是外力向量。从感觉上说，这些上述公式中的属于是与惯性，衰减，弹性，外界力所对应的。上述衰减模型为M和K的线性组合，是著名的Raleigh衰减模型且在事件中运行非常有效。对于一个有N个质点(粒子)的三维系统，上述矩阵的空间为3N*3N。公式(2)的问题非常著名，而且在[O’Brien et al. 2002]的工作中也有被讨论过。其中　　主要的不同在于本方法的力和惯性矩阵都是从弹簧质点模型中获取的，使得上述公式简单了许多。公式(2)的解析解获得思路可以从如下得到，首先将矩阵K对角化：

$$K=GDG^{-1} \tag{3}$$

其中G是由K矩阵的特征向量组成的实矩阵，D是由特征值组成的对角矩阵。在这里我们将G称为增益矩阵，将公式(3)的结果带回到公式(2)，然后再左边乘以一个G^{-1}:

$$G^{-1}M\frac{d^2r}{dt^2}+\left(\gamma G^{-1}M+\eta DG^{-1}\right)\frac{dr}{dt}+DG^{-1}r=f \tag{4}$$

我们观察到因为M是对角矩阵且归一化过的，所以G^{-1}M=M G^{-1}，我们定义z= G^{-1}r，公式(4)

可以被表示为:

$$M\frac{d^2z}{dt^2}+(\gamma M+\eta D)\frac{dz}{dt}+Dz=G^{-1}f \tag{5}$$

因为上述M矩阵和D矩阵都是对角矩阵，公式(2)因此已经被分离成一系列不相关的不同的以变量z定义的常微分方程，而且与物体的振动模式相对应。每一个模式的方程可以是标准的衰减振动方程，而且由如下的解析解:

$$\begin{aligned} z_i(t) &= c_i e^{\omega_i^+ t}+\overline{c_i}e^{\omega_i^- t} \\ \omega_i^{\pm} &= \frac{-(\gamma\lambda_i+\eta)\pm\sqrt{(\gamma\lambda_i+\eta)^2-4\lambda_i}}{2} \end{aligned} \tag{6}$$

其中常量c_i，被称作是模式的增益，由碰撞力决定，我们将在下一节进行讨论。我们用$\overline{c_i}$来表示c_i的复共轭，常数λ_i是对角矩阵的第i个特征值。$w_i^{\pm}$ 的实数部分指出了模式的衰减系数，同时虚数部分如果有的话就指出了模式的角频率。

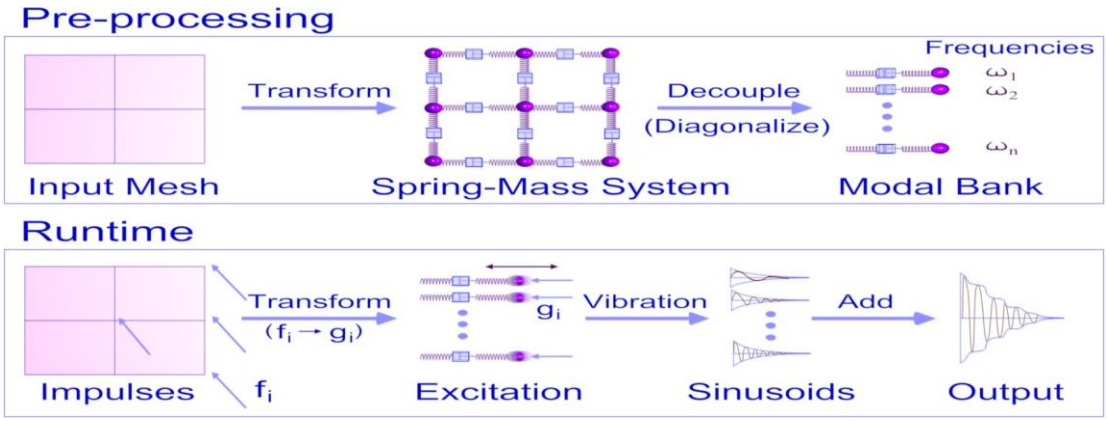

图 2: 该图描述了本方法的大纲。在前处理环节，每一个输入的表面网格结构通过将网格顶点和边用质点和阻尼弹簧代替从而被转换成弹簧质点系统。然后将得到的矩阵对角化来产生特征化的频率和衰减系数。在实时处理中，，刚体的模拟器在碰撞事件中报告碰撞力，然后这些碰撞力被转换为声音模式的增益（即每一个正弦波声音大小)。上述所有生成了一连串合适的衰减正弦波，他们通过傅里叶叠加在一起生成最终的声音。

3.3 碰撞力处理

一旦输入的网格已经被上述处理过而且对应的振动模式已经通过公式(2)-公式(6)提取

出来，我们拥有了所有被需要的物体有关发声性质的信息。物体声音的产生最终还需要取决于发声的量级和碰撞力在表面的位置。我们建模了利用德拉克Δ函数模拟了短时间的冲击性接触。假设一个碰撞向量f包含了应用于顶点的碰撞，我们可以计算出被改变的冲击，g= G^{-1}f来衡量公式(5)右边部分。一旦被建立，第i个模式可以被如下给出:

$$m_i \frac{d^2 z_i}{dt^2} + (\gamma m_i + \eta \lambda_i) \frac{dz_i}{dt} + \lambda_i z_i = g_i \delta(t - t_0) \tag{7}$$

其中t_0是碰撞事件，δ()是狄拉克Δ函数，将上述方程从t_0时间前到t_0事件后一点点进行积分，我们同时知道：，所以得到如下方程

$$m_i \Delta \left(\frac{dz_i}{dt} \right) + (\gamma m_i + \eta \lambda_i) \Delta z_i + z_i \Delta t = g_i \tag{8}$$

我们在这里认为Δt是非常小的，而且利用了形变相对于速度变化是非常小的事实，所以我们忽略了公式(8)中的左边部分最后两个部分来得到速度变化的情况：

$$\Delta \left(\frac{dz_i}{dt} \right) = \frac{g_i}{m_i} \tag{9}$$

上述方程给予了很简单的方法将振动模式时间导数的变化情况于冲击力联系在一起。考虑公式(6)我们且要求z_i在碰撞变化前后是一致的同时根据公式(9)，其导数$\frac{dz_i}{dt}$ 应该是增加的，那么振动模式的模式增益c应该是如此表示:

$$c_i \leftarrow c_i + \frac{g_i}{m_i \left(\omega_i^+ - \omega_i^- \right) e^{\omega_i^+ t_0}} \ . \tag{10}$$

4 实时声音合成

在这一章节中，我们描述了如何将上述的数学公式有效的利用起来生成实时声音。首先，我们描述了一个比较简单的实现方法，然后讨论了提升性能的更多技术。认为现在我们已经有一个刚体模拟器了来处理复杂场景。在碰撞事件中，声音系统首先被告知物体正在经历碰撞而且指导了碰撞力的力度和位置。这样的碰撞在3.3节已经被彻底描述过，我们知道这会导致物体不同振动模式的增益，即c_i的改变。这个在碰撞中产生的方程是由公式(6)产生的。再任何时候振动模式声音的大小是与速度有关的，而不是位置。这是因为一个质点的压力是由速度组成，同时振动模式的速度是与质点物理速

度线性相关的。振动模式的速度可以对公式(6)相对于时间做微分得到:

$$v_i = \frac{dz_i}{dt} = c_i\omega_i^+ e^{\omega_i^+ t} + \overline{c_i}\omega_i^- e^{\omega_i^- t} \tag{11}$$

为了生成声音的样本，我们需要通过公式(11)衡量物体所有振动模式，这是非常不高效的。正如 [O’ Brien et al. 2002]等人提到的方法，通过$e^{iw(t+\delta t)} = e^{iwt}e^{iw\delta t}$可以发现，声音音量增益的改变可以仅仅只需要一个复数乘以以前的数值。然而，这样的性能仍然不足够来实时处理大量物体的声音合成。我们估计整个声音合成系统运行时间如下：一个简单的弹簧致电系统有N个质点，对应3N个模式(自由度)，然后上述的操作需要对每一个振动模式对每一个声音采样。我们认为采样频率是44110Hz,那么生成t秒声音样本的浮点数操作层次的计算将会是：

$$T = 3N \times 4 \times 44100t\ FLOPS\ . \tag{12}$$

考虑如下典型的N取值：5000或者更高，产生长度疫苗的声音需要2646 MFLOPS.因为如今最快的处理器也只能实时处理几千个MFLOPS，所以上述处理过程会花费差不多一秒的时间 (即无法实时)。假设上述估计的时间仅仅是为一个物体的模拟，然而一个典型场景中可能有多个物体，这样的方法肯定是足够快来满足交互性的应用。同时，对于大多实时环境的应用比如游戏和虚拟环境，只有一小快的实际时间能够被分配给声音生成(大多数是其他视觉效果的生成)。所以在这一节的剩下部分，我们将讨论基于弹簧质点系统增快效率的技术，使得可以在交互的频率上来处理复杂场景中的多个物体。

从公式(12)，我们也可以很清楚的看出运行时间是跟振动模式的数量和物体数量成比例的。解析来，我们将提出一系列加速的方法来减少每一个物体中需要合成声

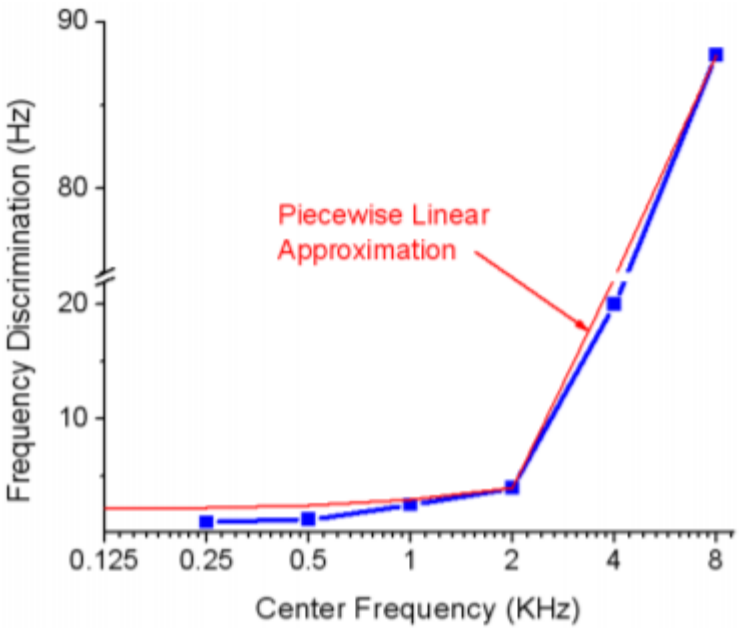

图3：该图展示了人体相对于中心频率而言对其邻近频率的区分能力。从途中可以看出，人类能够分清临近频率的能力在人能听到的范围：2-22KHz中，下降的非常快。我们利用了这样的数据实时来保证没有超过1000个正弦波模式被混合在一起合成一个物体的声音。从而使得几何信息的复杂的对产生正弦波的数量失去影响。红线表示了我们利用线性近似的方式来近似人体对于邻近频率的分辨曲线。

音所需要的振动模式：模式压缩和模式截断。然后利用对各个所需要声音合成的物体动态分级的方法来分配资源，从而加速且尽可能的减少声音质量的损失。

4.1 模式压缩

人类对于频率的感知是有限的，范围是20到22000Hz。它可以立刻被用于去除掉那些在感知范围以外的频率。然而，这里有一个更重要的发现可以利用于更大的衰减在模式混合。[Sek and Moore 1995]等人做了一个感知实验，表明人类对于区分临近频率的能力是有限的，这与[Zwicker and Fastl 1990]提出的有所不同：如果两个相近频率的声音同时输出，一个声音会掩盖另外一个。其实是即使这两个相近频率的声音在连续的时间里输出，听众也很难分清楚他们是两个不同频率的声音。这些作者称在中心频率附近一定频率范围内的频率对于人来说都是一样的，也被称为“Difference

Limens to Change(FLC)”.图3显示了DLC相对于中心频率(.25 to 8KHz)。有趣的是，DLC表明了相对于听者频率范围的大范围变化：在中心频率超过2Khz的时候是非常大的，即听者在超过2Khz的时候对周围频率大范围无法分清。甚至在2KHZ,DLC也是大于1Hz的,这意味着人本身无法从2000Hz中分辨1999Hz。

我们使用了上述的实时来大量减少需要合成一个物体的振动模式的数量。我们利用线性的规则来近似图三中的DLC曲线(分段直线用红线表示)，上述近似分为两个部分:一个是从20Hz到2KHz，另一个是2KHz到22KHz。正如在图三中规划的那样，我们的近似估计稍微过高。但是这增加了效率且我们已经在测试中观察到了很少的声音质量损失。这个压缩方法的主要想法是讲所有模式用很难分辨的频率成组分配到一起。这表示上述的线性近似方法可以被使用，而且可以把多个频率成组地聚集在中心频率上，使得整体上需要被处理地振动模式小于1000。这是非常重要的，因为1000的情况也只出现在从20到22000Hz每一个频率(21,22…)都出现，这是非常稀少的。对于大多数物体而言，频率总体的范围是离散的，因此，在使用了模式压缩方法后的振动模式数量往往小于1000个，基本上在几百个的数量级。

我们现在描述这个方法的具体细节。利用公式(3)中的G变量，这个增益矩阵有非常简单的物理描述:这个矩阵的行于物体顶点相对应，列与不同振动模式相对应(包括对应的频率)。G矩阵的每一列细数了物体不同振动模式的增益，当一个单位冲击被加在对应的定点上，利用上述讨论，所有在G这一行对应的有相近频率的需要被打包在一起。这过利用一个增益入口来代替其他现今频率增益的和来实现。因为一个模式对应一列，这样将操作减少为将所有列元素按列元素对应方式加在一起，整体的过程如下：

- 首先对G中的行按照振动模式频率进行排序。
- 以频率升序的方式遍历每个振动模式。来估计对应的DLC，利用图三的线性曲线进行近似。如果当前的频率和下一个频率在这两个频率无法分辨的范围内，就用他们两个列元素对应和来进行代替。接下来我们讲罗列使用这种方法的主要优势：

1.运行时间将会是一个常数而不是与物体顶点的数量线性相关。比如，如果输入的网格是由5000个顶点组成的，那么需要混合而产生声音的模式是小于1000的而不是3N=150000。

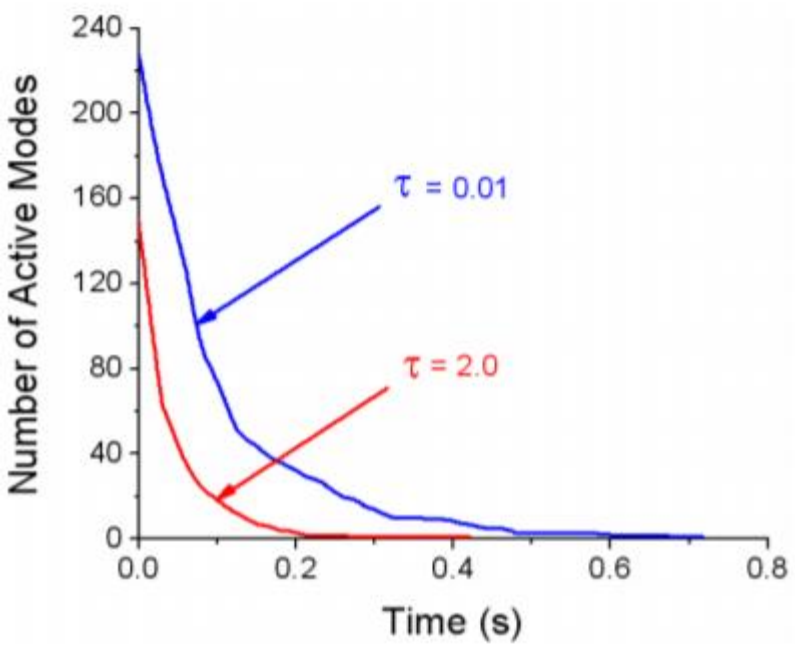

图4：该图表明了相对于时间混合在一起的模式(正弦波)数量。对于一个刚被敲击过的木琴而言，τ是模式截断的阈值，更大τ的取值可以导致在低声音量级更强烈的模式截断，使得可以存储更多的振动模式，比如$\tau = 2.0$可以导致截断更多的模式，将会有30%的效率提升，相对于$\tau = 0.01$几乎没有截断模式而言。而且上述两种情况所导致最后生成的声音质量几乎没有区别。

2.因为这个方法仅仅要求不同振动模式的频率，整个处理过程可以放在预处理步骤而不是实时中。

3.对于上述方法，很明显的是G矩阵列的数量是和振动模式数量相等的，现在也被削减到1000及一下而不是之前的3N。因为这个矩阵需要存入内存从而在实时的时候将冲击转换为模式的增益，本来它的内存消耗将会是一个大问题。用这个技术的话，对于一个有五千个顶点的物体而言，对内存的要求从225MB减少到小于15MB。

4.大多数物体都有一个离散的频率范围同时有很多杂音频率。这是由于对弹力矩阵K进行对角化的不准确和利用弹簧质点系统的近似方法造成的。他们很容易被错认为是能够分清的频

率，但我们的方法可以在绝大多数情况把他们自然的加在一起从而抵消掉大部分影响。

4.2 模式截断

物体的声音由被撞击产生的混合着高频率的短暂冲击组成，同时也会伴随一些低频率。这个短暂的碰撞时对声音质量非常关键的，因为从听觉认知层面上说这决定了声音的音质。所以模式截断方法的本质是当某些振动模式对整个声音的贡献掉到某一提前设定的阈值 τ 以下时，立即停止混入这个振动模式。因为当阈值设定合理的时候，模式阶段的方法能够保存住物体短暂碰撞，所以对整体声音质量的降低时很小的。图4就显示了仍然活跃的被混合的模式相对于时间的情况(木琴被敲进的场景,两个不同的阈值 τ： 0.01和2).这些值都是相对于采样值(65536 -16位声音)进行了标准化处理。所以阈值取0.01时几乎是没有截断任何模式的，仅仅只有一些声音大小接近于零的被阶段了。同时，阈值取2时，需要被混合的振动模式被减少了超过30%，且活跃的振动模式变为被截断的变化也更提前了。我们从图中也可以看出，阈值取2时对于感知上声音质量的减少是很小的。

整个技术的细节如下：我们认为一个物体正在经历一次碰撞，然后导致出的振动模式的增益c_i，是可以通过公式(10)计算得出，增益随时间变化直到下一次碰撞发生。公式(11)给出了振动模式对物体声音的贡献的近似表示。这可以被用来预测什么时候振动模式的贡献衰减到阈值 τ 以下，所以得出的截断的时间t_i^c,即当所有合成时间$t > t_i^c$时，进行截断。

$$c_i\omega_i^+ e^{\omega_i^{'} t} + \overline{c_i}\omega_i^- e^{\omega_i\ t} < \tau \tag{13}$$

利用一个常见的复数公式:对于任意复数x和y而言，$|x + y| \leq |x| + |y|$,来得到下列公式

$$t_i^c \leq \frac{1}{-Re(\omega_i^+)} \ln\left(\frac{2\,|c_i|\,|\omega_i^+|}{\tau}\right) \tag{14}$$

因此，利用上述的不等式，一个碰撞发生后，所有振动模式的截断的时间可以被计算出来。当产生一个模式的声音样本时，仅仅一个浮点数运算需要被完成来确认是不是当前合成时间已经超过了截断时间。如果超过了，即该模式的贡献已经衰减到阈值以下，后续将不再被考虑。

4.3 质量分级

上述两种技术主要是用于提升单个物体声音合成的效率。然而，当场景中发生物体的数量增长到几十上百个时，该方法将不会足够有效进行实时合成，而且也不可能给所有物体的声音都以最高质量进行输出。对于实时系统而言，声音系统有一个比较合理的方法来对合成声音的质量和时间进行灵活限制是非常关键的。我们通过给声音质量分级的方法进行实现。一个物体的声音质量时通过控制混合振动模式的数量来进行改变的。对于大多数场景而言，对于多个发生物体，使用者的注意力往往只集中在视野前方，所以对于发生物体而言，在正前方的发生物体对场景中整体声音贡献最多。因此，如果在正前方物体的声音是以最高质量进行合成的，在后方的物体是以较底质量合成的，这样导致的在听觉层次上的质量下降将会很少，但同时合成速度得到了很大提升。

我们使用了简单的流程来确保了正前方物体的高质量声音。在视频每一帧的末尾，我们都存储了每一个物体所以振动模式音量大小的和，在下一帧，所有的物体根据他们的优先级按照降序排列，同时对于合成声音的时间配合按照线性下降，S。然后，所有的物体按照这个顺序进行处理。对于每一个物体，它的声音质量是首先被分级了的所以他能够在给定的时间配额里进行声音合成，即振动模式在这一段时间内进行混合。下降曲线，S决定了正前方声音质量对于候方声音质量下降的程度(快慢)。所以如果S=0,表面光没有优先级的分配。所有的物体都是被分配了相同的时间配额进行声音合成。如果S=∞,那么对应了给最高优先级的物体近乎全部的时间配额，当这个最高优先级物体合成完毕，剩下的时间若有的话才分配给下一个物体。

4.4 整体加速流程

为了讲述上述描述的所有的技术是如何集成在一起的，我们描述了如下方法的流程。

预处理部分:

- 建立对应于输入的物体网格的弹簧质点系统 (3.1节)。
- 处理弹簧质点系同来提取出增益矩阵G，和物体振动模式的角频率(3.2节，公式(3)到公式(6))。

• **模式压缩**：如4.1节所说，将矩阵G的列组合到一起，根据对应振动模式的频率。

• 在模式压缩后，存储最终的增益矩阵和对应的模式频率，需要注意的是当w_i^+没有虚部的时候，w_i^-不需要被保存。

实时处理部分:

• 读取每一个物体的增益矩阵和对应的振动模式数据。

• 开始模拟的循环流程:

1. 对于每一个物体O：

- **碰撞处理**：如果刚体的模拟器得到信息这个物体O正在经历碰撞事件，那么就根据公式(10)更新相应的参数，利用模拟器得到的碰撞冲击大小和位置(3.3节)。

- **模式截断**：对每一个振动模式根据截断的阈值计算截断时间t_i^c(4.2节，公式(14))。

2. **质量分级**：按照声音大小的贡献对物体进行排序，分配合理的时间配额，然后计算每一个物体需要被混合的振动模式的数量(4.3节)。

3. **声音合成**：对于每一个在时间t的时间片和每一个物体O:

- 考虑在当前质量设定允许条件下的所有的模式$t < t_i^c$，即没有被截断的部分，对所有振动模式进行采样和叠加就是物体合成的总的声音。然后将最终的合成声音进行输出。

5 实现细节和结果

在这一节我们将介绍模拟结果来展现本方法的性能和效果。

5.1 刚体模拟器

我们是利用C++和OpenGL实现了上述算法和加速技术。我们的刚体模拟器将Guendelman et al. [2003]和DEEP [Kim et al. 2002]等人的技术组合在一起，从而实现了更快速更准且的碰撞估计。同时也使用了Mirtich and Canny [Mirtich and Canny 1995]提出的更复杂的摩擦模型，使得在碰撞的计算上更加准确。

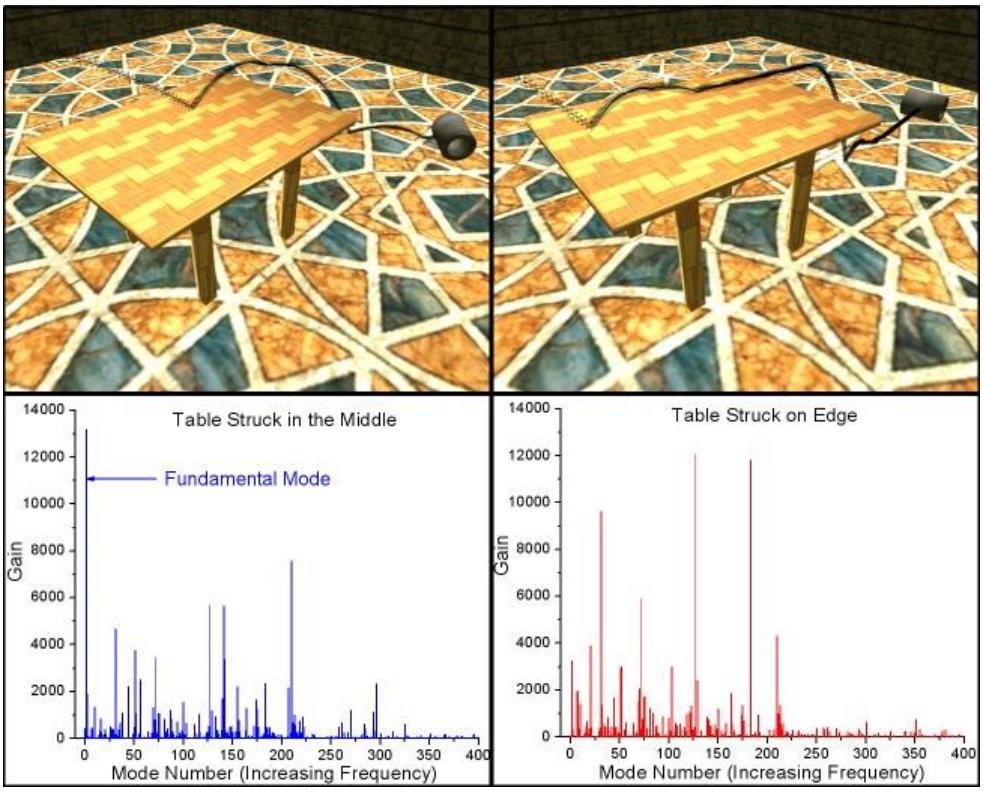

图5：一个金属的圆柱掉落在木制的桌椅上：掉在桌子中间(左图)，掉在桌子边缘(右图),然后滚落。底部图部分显示出对应的频率光谱，能够看出，对于左图而言，大多数碰撞冲击被转换为了低频的振动模式，而右图大多被转换了高频频率。

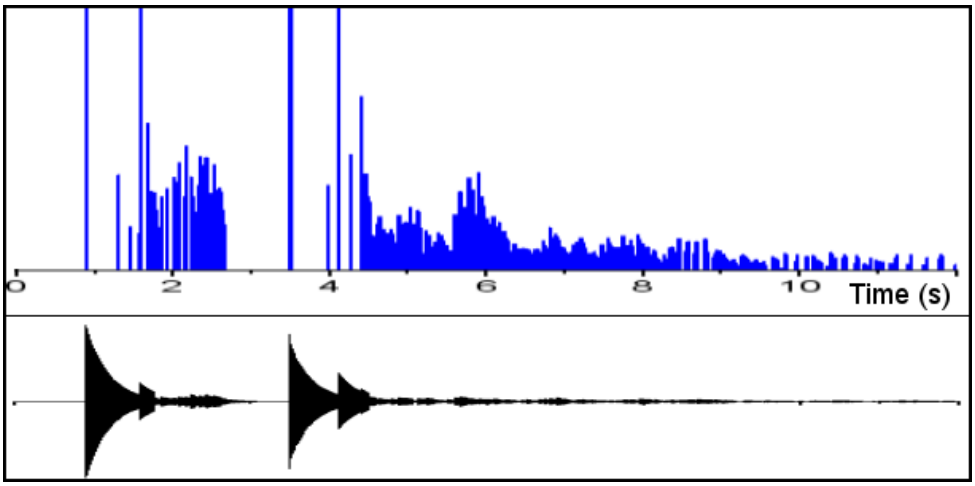

图6：图5场景中，在圆柱上的冲击随时间变化情况其中峰值表示撞击发声，小的连续的较低的值表示滚动力作用。

5.1 刚体模拟器

我们是利用C++和OpenGL实现了上述算法和加速技术。我们的刚体模拟器将Guendelman et al. [2003]和DEEP [Kim et al. 2002]等人的技术组合在一起，从而实现了更快速更准且的碰撞估计。同时也使用了Mirtich and Canny [Mirtich and Canny 1995]提出的更复杂的摩擦模型，使得在碰撞的计算上更加准确。

5.2 取决于碰撞位置的声音

正如上述所讨论的，使用基于物理的声音比利用记录的声音主要的优势在于它对物体之间碰撞的影响效果的抓取，以及对于击打物体不同位置声音不同的性质的模拟。图5展现了一个金属圆柱被扔到木制桌椅上的场景。桌椅和圆柱都在发声。图5展现了两种不同的情况：第一种见图5左边，描绘了圆柱掉落在桌子中间然后滚落，第二种见图5右边，圆柱掉落在圆柱边缘然后滚落。我们在下一节讨论了滚动声音，在这里将继续详细讨论碰撞声音。因为我们将桌子处理为盘子的形式，所以如果敲击发生在桌子的边缘，将会被理解为一个更大的冲击导致更高的频率，而敲击发生在盘子中间就被理解为一般的冲击导致较低的频率，从而导致了有深度的声音。为了验证这个效果，在圆柱发生第一次碰撞，我们实时给上述两种情况画出了频率的光谱图。从图中可以看出，左下图中的峰值出现在基本频率附近而右下图的峰值不是。这个不同清晰地展现了声音在上述两种情况下是不同的，正如通过分析性质一样。另外一个要点是这个技术并不要求网格必须是高度棋盘化分布(即网格密度很高)来获取上述性质。这个桌子仅仅由600个顶点，圆柱由128顶点组成。

5.3 滚动声音

除了能够处理碰撞声音，本方法也能够模拟真实的滚动声音且不需要其他特殊的处理手段。这是由于现有的刚体模拟其已经能够将碰撞力处理的比较好。图6展现了对于圆柱的冲击和对应的声音产生情况。整个场景如图5所示，圆柱在桌子上滚动然后掉在地上，一开始的在桌子上的滚动声音质量非常好。随着圆柱的滚动，桌子的声音传递出对圆柱声音的反馈。圆柱的网格尽管是被棋盘化了，也非常粗糙，仅仅由32个圆周分区组成。图6展现了随着时间变化，在圆柱上的作用力变化情况。力峰值与碰撞相对应：

圆柱掉落在桌子上和掉落在地面上，同时声音的峰值也相对应。6秒-8秒是由一系列的圆柱在地面上发出的滚动声音组成，也能看出对应的力的冲击时非常小，且连续的，且与圆柱的网格相对应(圆形滚动)。为了测试这些冲击的周期性对滚动声音真实性的影响，我们计算了图六数据中冲击的平均时间的平均值和标准差。平均的冲击持续时间是17秒，同时标准差为10秒。标准差比平均值的一般都要大说明冲击几乎不是周期性的。这表明周期性的碰撞并不是对滚动声音真实性感知的关键因素。

5.4 性能分析

我们能够利用本方法给复杂场景进行实时声音模拟。图7显示了一个有100个金属环碰撞木地板发声的场景。所有的圆环和桌面都在发出声音。每一个圆环都被当做单独的物体有自己的声音性质来对待。这个圆环由200个顶点组成。图8展现了一秒的模拟中声音的FPS(在这期间所有的碰撞都发生了)。这个场景的帧率是100FPS。这表明这不是原始数据而是一个移动的平均值所以小范围的波动时可以接受的。在这图的底部是在额米有任何第四节所描述加速技术使用的基础的FPS。声音在这种情况下是非常不连贯的，因为声音的合成不能跟上刚体模拟的交互速度。在有模式压缩和模式阶段的情况下，性能有了很明显的提升。然而在一开始是200FPS以后，当最大数量的碰撞发生时，帧率迅速的掉落了。当再加入了质量分级技术，帧率终于可以保持在200FPS周围，即使是上述最坏的情况发声。这是因为质量分级给予了这些圆环在发生碰撞时进行声音合成的优先级，同时降低了那些碰撞发生在之前的圆环的声音质量，从而减少了总体的模拟时间。这能够充分展现出质量分级技术对有较多物体场景进行声音模拟的重要性。这表明即使现在的例子能够充分表明对声音合成系统稳定帧率的保持，但对于实际的应用也是很难得，因为将会有一百个碰撞在一秒以内发声。这也是为什么图8的场景中，CPU利用率那么高(50%),而图1的场景下CPU利用率较低，仅仅百分之十。

为了说明本方法实时声音合成可以被实现。我们设计了图1所示的木琴场景来进行验证。从图中可以看出许多骰子掉落在母亲的键位上来发出对应的音乐声音。声音的模拟在这个场景中可以跑出500-700的FPS，取决于碰撞的频率。骰子已经被编码成能够按照准确的运动到对应的木琴键位上来发出一系列音符。由于声音合成的效率很高，所以这个场景能够很容易的保持在100FPS的稳定帧率。同时，也有一些许多骰子互相

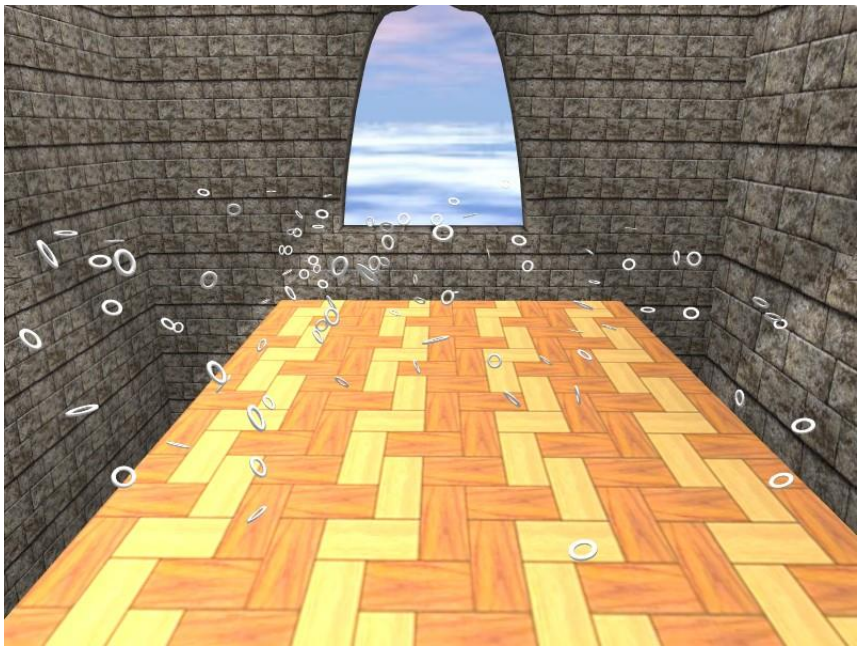

图 7：超过100个金属圆环掉落在木制桌子上，所有的环和地板都在发声。这个声音模拟能够在200FPS的速度上运行，这个程序的帧率是100FPS.所以说质量分级的方法保证了被听到的声音的质量不会降级，同时保证了持续的帧率(见图8)。

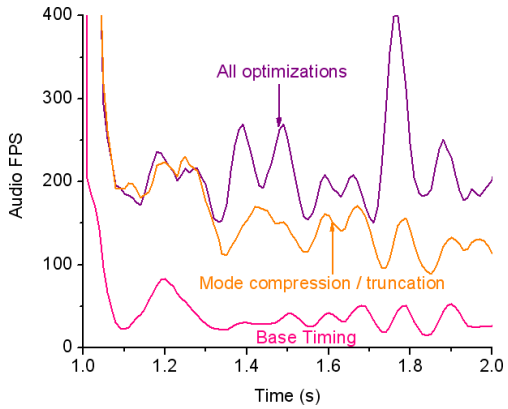

图 8：这张图显示了在图7中的场景声音模拟的FPS(从1秒到两秒),这期间几乎所有的碰撞都发生了。最下方的曲线显示了在没有使用任何加速技术的情况。最上方的曲线显示了使用了模式压缩，模式截断和质量分级技术的情况。请注意FPS是如何保持在近200且其他两条曲线由于太多的碰撞声音在1.5-2秒产生时而下降了。

掉落在了不同木琴键位的情况，但是对整体声音影响不大。尽管我们还没有对模拟的木琴调音到与真实魔琴的频率光谱一模一样，生成的合成声音已经很真实而且抓住了这个乐器的音质。木琴的材质信息是从[Chaigne and Doutaut 1997]的工作中抓取的。

6 结论

本工作提出了一种基于物理的声音合成算法以及一些加速技术来渲染出实时的拥有成百上千的发生物体的大型场景。该方法不对物体网格特征有要求，非常利于实现，而且利用了已存在的硬件加速技巧。我们计划延申这个工作到讲其他的如滑动声音，爆破声音，破碎声音等融合到实时场景中去。

参考文献

CHAIGNE, A., AND DOUTAUT, V. 1997. Numerical simulations of xylophones. i. time domain modeling of the vibrating bars. *J. Acoust. Soc. Am. 101*, 1, 539–557.

CHUNG, J. Y., LIU, J., AND LIN, K. J. 1987. Scheduling real-time, periodic jobs using imprecise results. In *Proc. IEEE RTS*.

DONGARRA, J. J. 2005. Performance of various computers using standard linear equations software (linpack benchmark report). Tech. rep., Knoxville, TN, USA.

FLORENS, J. L., AND CADOZ, C. 1991. The physical model: modeling and simu- lating the instrumental universe. In *Represenations of Musical Signals*, G. D. Poli,

A. Piccialli, and C. Roads, Eds. MIT Press, Cambridge, MA, USA, 227–268.

FOUAD, H., BALLAS, J., AND HAHN, J. 1997. Perceptually based scheduling algo- rithms for real-time synthesis of complex sonic environments. In *Proc. Int. Conf. Auditory Display*.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex rigid bodies with stacking. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH) 22*, 871–878.

KIM, Y. J., LIN, M. C., AND MANOCHA, D. 2002. DEEP: an incremental algo-

rithm for penetration depth computation between convex polytopes. *Proc. of IEEE Conference on Robotics*

and Automation, 921–926.

MIRTICH, B., AND CANNY, J. 1995. Impulse-based simulation of rigid bodies. In *1995 Symposium on Interactive 3D Graphics*, P. Hanrahan and J. Winget, Eds., ACM SIGGRAPH, 181–188. ISBN 0-89791-736-7.

O'BRIEN, J. F., COOK, P. R., AND ESSL, G. 2001. Synthesizing sounds from physically based motion. In *SIGGRAPH '01: Proceedings of the 28th annual con- ference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 529–536.

O'BRIEN, J. F., SHEN, C., AND GATCHALIAN, C. M. 2002. Synthesizing sounds from rigid-body simulations. In *The ACM SIGGRAPH 2002 Symposium on Com- puter Animation*, ACM Press, 175–181.

SEK, A., AND MOORE, B. C. 1995. Frequency discrimination as a function of frequency, measured in several ways. *J. Acoust. Soc. Am. 97*, 4 (April), 2479–2486.

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 Displays*.

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. Foleyautomatic: physically- based sound effects for interactive simulation and animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 537–544.

VAN DEN DOEL, K., KNOTT, D., AND PAI, D. K. 2004. Interactive simulation of complex audiovisual scenes. *Presence: Teleoper. Virtual Environ. 13*, 1, 99–111.

ZWICKER, E., AND FASTL, H. 1990. In *Psychoacoustics*. Springer-Verlag, Berlin.