**大型场景的交互声音合成**

## Nikunj Raghuvanshi Ming C. Lin

计算机学院

北卡罗纳教堂山分校

**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

# 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。

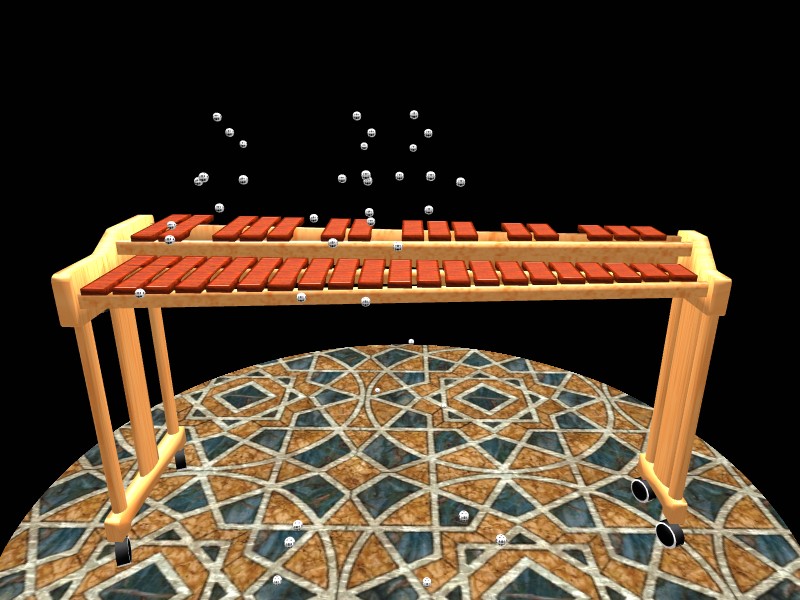


图1: 许多骰子连续掉在24音阶的木琴来播放歌曲”取悦者”。对于这个复杂场景，我们的算法能够产生在以最高500FPS下的声音且只占用了3.4GHz Pentium-4 笔记本电脑CPU周期的百分之十

**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。

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：

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。

*•*

该方法对输入的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然后直接用于声音生成。

*•*

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。

*•*

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 EAXTM来提供基于Creative Sound Blaster Audigy 2TM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。然后在第三届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。

# 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，更多有关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。但主要的缺陷是公式和实现的复杂性较高，且模拟的速度比较慢。

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因此实现了更好的结果。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，且同时保证最小化的削减了声音的质量。

# 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。

### 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:

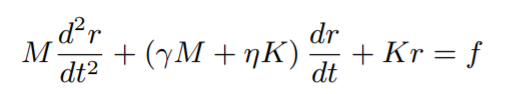
*k* = *Y t*

*mi* = *ρtai* (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，我们将在后节进行讨论。

### 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：

 (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

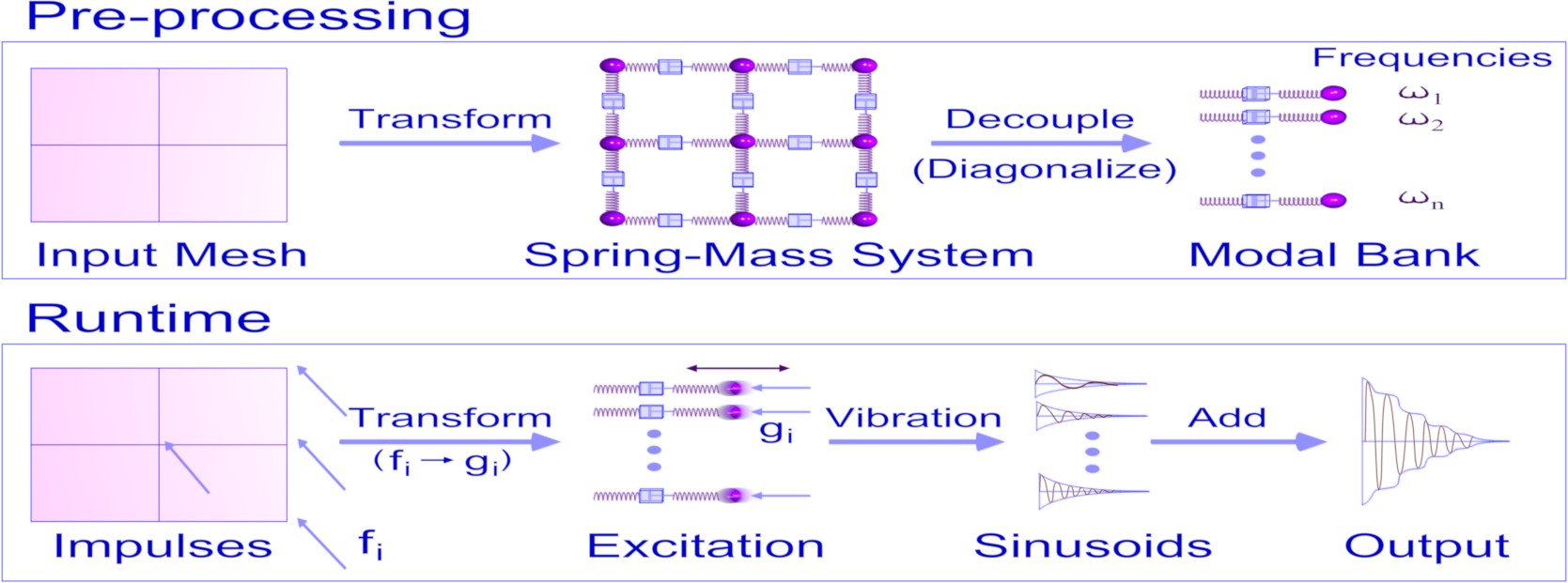


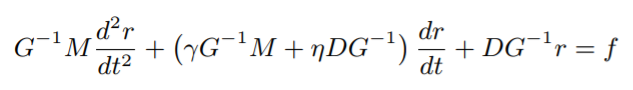
图 2: 该图描述了本方法的大纲。在前处理环节，每一个输入的表面网格结构通过将网格顶点和边用质点和阻尼弹簧代替从而被转换成弹簧质点系统。然后将得到的矩阵对角化来产生特征化的频率和衰减系数。在实时处理中，，刚体的模拟器在碰撞事件中报告碰撞力，然后这些碰撞力被转换为声音模式的增益 (即每一个正弦波声音大小)。上述所有生成了一连串合适的衰减正弦波，他们通过傅里叶叠加在一起生成最终的声音。

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。

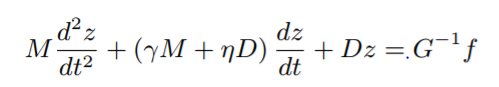
公式(2)的问题是非常著名的，而且在[O’Brien et al. 2002]的工作中也有被讨论过。其中主要的不同在于本方法的力和惯性矩阵都是从弹簧质点模型中获取的，使得上述公式简单了许多。公式(2)的解析解获得思路可以从如下得到，首先将矩阵K对角化：

 (3)

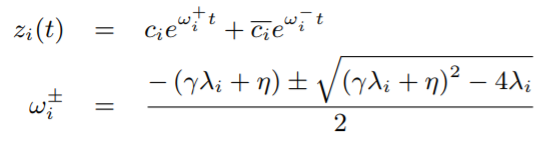
其中G是由K矩阵的特征向量组成的实矩阵，D是由特征值组成的对角矩阵。在这里我们将G称为增益矩阵，将公式(3)的结果带回到公式(2)，然后再左边乘以一个G-1:

 (4)

我们观察到因为M是对角矩阵且归一化过的，所以G-1M=M G-1，我们定义z= G-1r，公式(4)可以被表示为:

 (5)

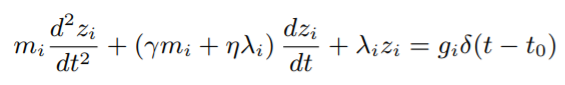
因为上述M矩阵和D矩阵都是对角矩阵，公式(2)因此已经被分离成一系列不相关的不同的以变量z定义的常微分方程，而且与物体的振动模式相对应。每一个模式的方程可以是标准的衰减振动方程，而且由如下的解析解:

 (6)

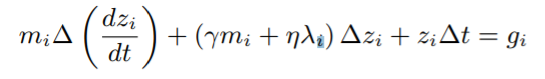
其中常量，被称作是模式的增益，由碰撞力决定，我们将在下一节进行讨论。我们用来表示的复共轭，常数是对角矩阵的第i个特征值。的实数部分指出了模式的衰减系数，同时虚数部分如果有的话就指出了模式的角频率。

### 3.3 碰撞力处理

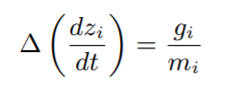
一旦输入的网格已经被上述处理过而且对应的振动模式已经通过公式(2)-公式(6)提取出来，我们拥有了所有被需要的物体有关发声性质的信息。物体声音的产生最终还需要取决于发声的量级和碰撞力在表面的位置。我们建模了利用德拉克Δ函数模拟了短时间的冲击性接触。假设一个碰撞向量f包含了应用于顶点的碰撞，我们可以计算出被改变的冲击，g= G-1f来衡量公式(5)右边部分。一旦被建立，第i个模式可以被如下给出:

 (7)

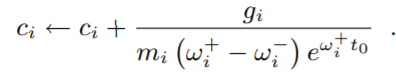
其中是碰撞事件，是狄拉克Δ函数，将上述方程从时间前到事件后一点点进行积分，我们同时知道：，所以得到如下方程

 (8)

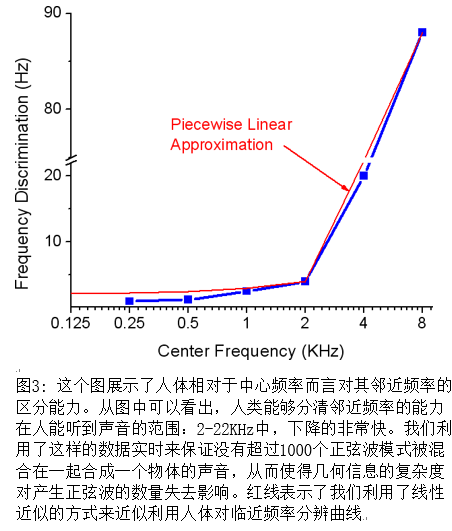
我们在这里认为Δt是非常小的，而且利用了形变相对于速度变化是非常小的事实，所以我们忽略了公式(8)中的左边部分最后两个部分来得到速度变化的情况：



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)我们且要求在碰撞变化前后是一致的同时根据公式(9)，其导数 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:

 (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，然哦胡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:

 (11)

为了生成声音的样本，我们需要衡量

# 4.1 模式压缩

# Real-time Sound Synthesis

In this section, we describe how the mathematical formulation pre- sented 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 han- dle all the dynamics. During a collision event, the sound system is informed of the object that undergoes the impact and the magni- tude and location of impact. This impact is processed as described in Section 3.3 to result in the gains for the different modes of vibra- tion of the object, where the gain for the *i*’th mode being *ci*. 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 proportional1 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* = *d* *zi* = *c ω*+*eω*+*t* + *c ω*−*eω−t* (11)

ple spring-mass system with *N* particles has 3*N* modes, and the above operation needs to be repeated for each mode for each au- dio 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* = 3*N ×* 4 *×* 44100*t FLOPS .* (12)

Considering that the typical value of *N* is about 5000 or higher, pro- ducing sound worth 1 second would take 2646 MFLOPS. Since to- day’s fastest processors operate at a few thousand MFLOPS [Don- garra 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 propor-

*i dt*

*i i i*

*i i i*

tional to the number of modes being mixed and the number of ob-

For generating each audio sample, we need to evaluate the above

equation for all vibration modes of the object, which is quite inef- ficient. As mentioned in [O’Brien et al. 2002], the simple observa- tion that *eiω*(*t*+∆*t*) = *eiωteiω*∆*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

1The 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.

jects. Next, we present acceleration techniques for sound simula- tion by reducing the number of modes per object: “Mode Com- pression” and “Mode Truncation”, and by scaling the audio quality of different objects dynamically with little degradation in perceived sound quality.

### Mode Compression

Humans have a limited range of frequency perception, ranging from 20 to 22000 Hz. It immediately follows that modes with frequen- cies 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 percep- tual study described in [Sek and Moore 1995] shows that humans have a limited capacity to discriminate between nearby frequen-

cies. Note that this is different from frequency masking [Zwicker and Fastl 1990] in which one of two *simultaneously* played frequen- cies 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 audi- ble 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

1. 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 fur- ther 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 lin- ear approximation to the DLC curve is used and indistinguishable modes clustered at the corresponding frequency centers, the maxi- mum 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), *tt*. 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 vi- bration (with their corresponding frequencies). Each row of *tt* 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 *tt* which correspond to modes with close frequencies need to be clus- tered 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 con- stituent 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 *tt* with the corresponding mode frequen- cies as the key. 2

*•*

Traverse the modes in increasing order of frequency. Estimate the DLC, ∆ at the current frequency using the piecewise lin- ear curve shown in Figure 3. If the current frequency and next frequency are within ∆ of each other the two mode frequen- cies 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 number of 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 3*N* = 15*,* 000 which is a substantial performance gain.

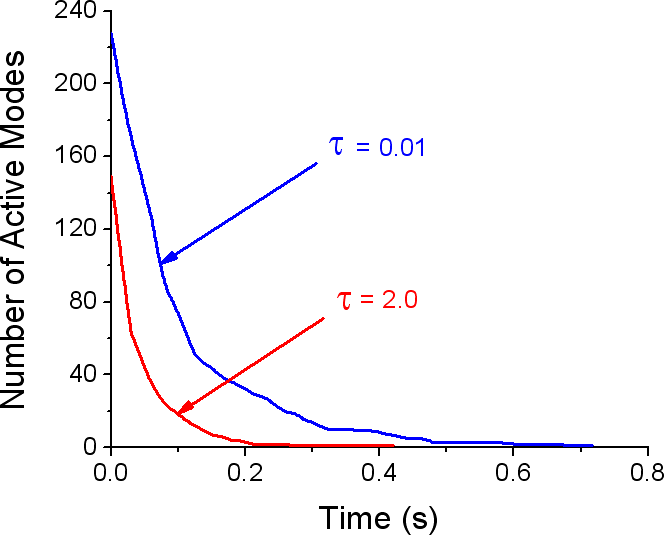


图 4:这个图表明了相对于时间混合在一起的模式数量，对于一个刚被敲击过的木琴而言，*τ*是模式截断的阈值。更大*τ*的取值可以导致在低声音大小量级更强烈饿的模式截断，使得可以存储更多的振动模式。比如*τ* = 2*.*0可以导致截断更多的模式，将会有30%的效率提升，相对于*τ* = 0*.*01几乎没有截断模式而言。而且上述两种情况所导致最后生成的声音质量几乎没有区别。

* 1. Since this scheme requires just the frequencies of the different modes, the whole processing can be done as a pre-process without requiring any extra runtime CPU cycles.
  2. From the above mentioned procedure, it is clear that the num- ber of columns in the matrix *tt*, which is the same as the num- ber of modes, is now bounded by 1000 instead of the earlier value of 3*N* . Since this matrix needs to be present in memory at runtime for transforming impulses to mode gains, its mem- ory 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.
  3. Most objects have a discrete frequency spectrum with pos- sibly many degenerate frequencies. Due to numerical inac- curacies 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.

### 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 charac- teristic “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 pre- serves 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 xy- lophone 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 ob- ject has just undergone a collision and the resulting mode gains *ci* have been calculated as given by Equation (10). From this time on- wards 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”, *tc* is such that for all times *t > tc*:

### 声音合成整体流程

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

### 预处理

Construct a spring-mass system corresponding to the input mesh. (Section 3.1)

*•*

Process the spring-mass system to extract the gain matrix, *tt* and the (complex) angular frequencies of the object’s modes of vibration: *ω*+ and *ω*−. (Section 3.2, Eqns. (3) and (6))

*•*

*i i*

### 模式压缩:

*•*

Aggregate columns of *tt* based on frequencies of the corre- sponding modes, as described in Section 4.1.

*i i*

+ *−*

*ciω*+*eωi t* + *ciω*−*eωi t < τ* (13)

*i i*

* + - Store the resulting gain matrix along with the (complex) con-

stants *ω*+ and *ω*− for modes correspond to the columns of

*i i*

Using the fact that for any two complex numbers *x* and *y*, *|x* + *y| ≤* −

*|x|* + *|y|* it can be shown that,

1 2 *|ci|* ˛*ω*+˛ !

+

*tc ≤*

ln

*i*

*i*

*−Re*(*ωi* )

*τ*

*tt* after compression. Note that *ωi* need not be stored in case *ω*+ has a non-zero imaginary part since in that case

*ω*− = *ω*+.

*i*

*i*

*i*

**Runtime Processing**

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.

(14)

### 声音质量分级

The two techniques discussed above are aimed at increasing the ef- ficiency 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 qual- ity 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 am- plitude. 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 fore- ground 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 ob- jects 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 con- sumed greedily by the next higher priority object. The slope, *S* of the ramp decides the degree to which the foreground sound qual- ity 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 con- verse case with *S* = 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.

*∞*

* 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* under- goes 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 *tc* for each mode based on the mode truncation threshold, *τ* . (Section 4.2, Equation (14))

*j*

### 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)

### Sound Synthesis:

For each timestep at time *t* and for each object, *O*:

* + - * Consider all modes permitted by the current qual- ity setting which satisfy *tc > t*. Sample and sum- mate all these modes as described at the begin- ning of this section. This is *O*’s contribution to the sound.

*j*

* + - * Output the sum of all objects’ sample contribution as the sound sample for time *t*.

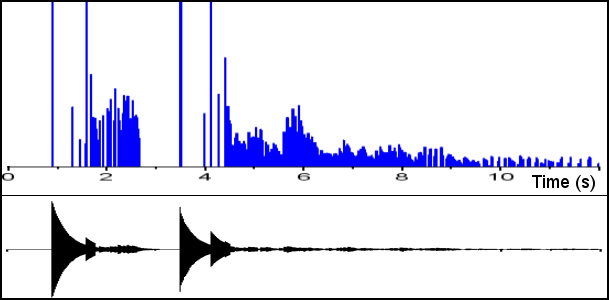
End simulation loop

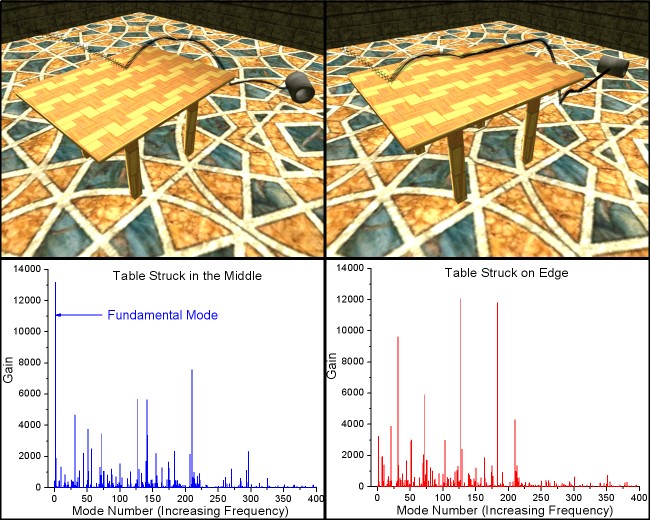
# Implementation and Results

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

### 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 es- timation 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 mid- dle (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.

### 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 impor- tantly, 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 fig- ure 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.

### 滚动声音

In addition to handling impact sounds, we are able to simulate re- alistic 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 sim- ulators. Figure 6 shows the impulses on the cylinder and the cor- responding audio for the case shown in the right side of Figure 5.

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 low- amplitude 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 im- pact. The cylinder, although uniformly tessellated, is very coarse, with only 32 circumferential subdivisions. Figure 6 shows the im- pulses 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 tes- sellation. To test how important the periodicity of these impulses was for the realism of rolling sounds, we found the mean and stan- dard 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 stan- dard deviation is more than 50% of the mean demonstrates that the impulses show very little periodicity. This suggests that the period- icity of collisions is not critical for the perceived realism of rolling sounds.

### 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。

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 elas- tic 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 FPS3 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 without using any of the acceler- ation 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 trun- cation 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 com- pression 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

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

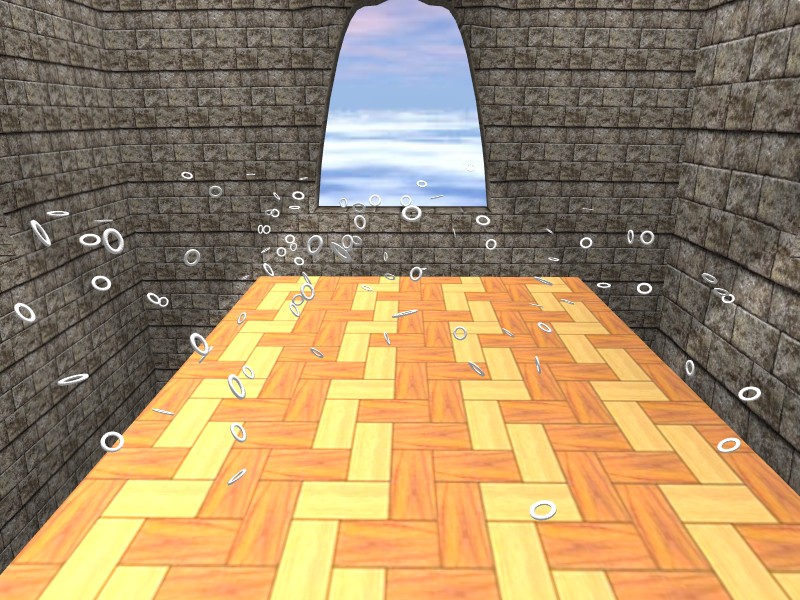
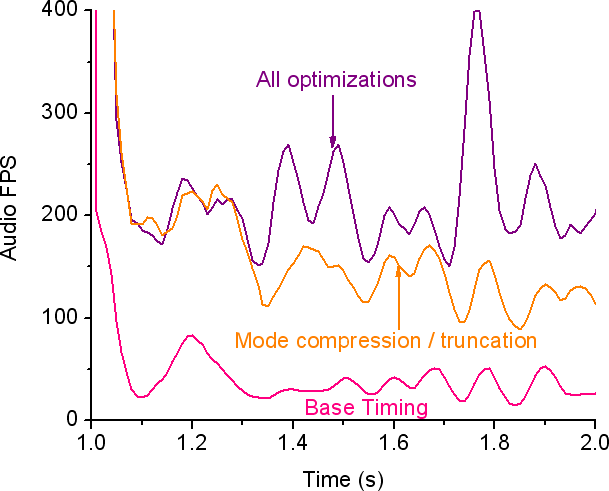
 

图 7: More than 100 metallic rings fall onto a wooden table. All the rings and the table are sounding. The audio simulation runs at more than 200 FPS, the application frame rate being 100 FPS. Quality Scaling ensures that the perceived sound quality does not degrade, while ensuring steady frame rates (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 sec- ond. 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%).

To illustrate the realistic sounds achievable with our approach, we designed a three-octave xylophone shown in Figure 1. The im- age shows many dice falling onto the keys of the xylophone to pro- duce the corresponding musical notes. The audio simulation for this scene runs in the range of 500-700 FPS, depending on the fre- quency 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 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 pa- rameters for the xylophone were taken from [Chaigne and Doutaut 1997].

# Conclusion

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.

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 top- most curve shows the FPS with mode compression, mode trunca- tion 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 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.