Eurographics Symposium on Point-Based Graphics (2004)
M. Alexa, S. Rusinkiewicz, (Editors)

Multi-Resolution Sound Rendering

M. Wand and W. Straßer

WSI/GRIS, University of Tübingen

Abstract

Point-based multi-resolution representations have been used successfully for rendering highly complex three dimensional scenes in real-time. In this paper, we apply this paradigm to sound rendering: A hierarchy of stochastic sample sound sources is used to approximate complex sound environments (containing a large number of sound sources, such as a football stadium), allowing for interactive real-time walkthroughs. Additionally, the proposed technique can be used for observer-dependent auralizations of simple approximations of global sound propagation. Typical applications of the technique are in virtual reality and computer games, especially to complement established output-sensitive algorithms for rendering visual content.

Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Methodology and Techniques – Graphics data structures and data types; G.3 [Probability and Statistics]: Probabilistic algorithms

1 Introduction

One of the major goals of computer graphics is the faithful simulation of artificial environments. Sound is an important aspect of most natural environments and should be included in realistic simulations. It provides important spatial cues and has an especially strong effect on the emotional perception of a scene.

In this paper, we consider the problem of real-time sound rendering for scenes containing a large amount of sound emitters. Typical applications that have to deal with this problem could be a computer game showing complex crowd simulations (such as a football stadium in a sports game) or a VR-walkthrough of a virtual prototype of a production facility (with the sound of many machines and virtual workers). The sound rendering literature provides sophisticated techniques for physically based simulation of sound sources [DKP01, BSG02, BCE01], for simulating the propagation of sound in complex environments [AB79, FCE*98, Sav99, TFN*01], and for auditory display [FJT02], i.e. faithful reproduction through loudspeaker or headphone setups. Additionally, different systems have been presented that integrate sound rendering with virtual reality [NSG02, Sav99]. However, settings with a large number of sound emitters have received only little attention yet.

Reconstructing the acoustic impression from a large number of sound emitters is very similar to the visual rendering problem for large scenes. For visual rendering, we are interested in reconstructing the effect of a large set of electromagnetic waves on the viewer, rather than acoustic waves. A recent, very general approach for effective visual rendering is point-based multi-resolution rendering [PZvB*00, RL00, WFP*01]: A set of surface sample points is chosen with a sampling density declining with the distance to the viewer. After that, the light emission and occlusion of this sample set is considered to reconstruct an image of the scene. In this paper, we propose a similar strategy for rendering sound: Our rendering algorithm approximates the effect of a potentially large set of sound sources by taking a small sample set of representative sources. A dynamic importance sampling strategy is combined with a spatial data structure to extract sample sets in output-sensitive time, independent of the number of sound emitters. The reconstruction is performed by weighting and mixing the sample contributions.

Our algorithm can also be used as "back end" for the auralization of global sound propagation: First, an offline algorithm is used to simulate the global propagation of sound. It places a dense set of secondary sound sources in the scene, along with phase, volume, and directional emission information that approximate the global interreflection. The sampling algorithm is then used to perform a real-time observer dependent "final gathering", as known from the image synthesis literature [Gla95, Kel97, SP94].

We have implemented a prototype system that allows walkthroughs of complex scenes using the proposed algorithm for sound rendering and point-based multi-resolution rendering for rendering the visual presentation. Experiments show that convincing approximations of complex scenes can be handled in real-time using a contemporary PC system.

2 Related Work

Much work has been published for simulating global sound propagation. The simulation can be done using different basic approaches. An elementary approach is *wave based* simulation: Finite elements, finite differencing, or similar techniques are used to solve the Helmholtz equation that describes the wave propagation [Sav99]. The drawback of this approach is that the simulation costs are high and the expenses grow strongly with the maximum frequency to be handled. A second basic approach is a *geometric simulation* of sound propagation using raytracing techniques [KSS68]. A classic method is the image source algorithm [AB79, Bor84]: Specular reflection paths are enumerated by considering the source and the receiver and each reflecting polygon in the scene. Then, the paths are replaced by "image sources". The technique permits capturing all ideal specular reflection paths. However, the combinatorial complexity of this process renders a straightforward adoption of the algorithm to complex environments infeasible. Savioja et al. [SHL*97] use geometric data structures and clustering to accelerate the computation and rendering of the image source. Another possibility is the usage of path tracing techniques: Variants of the photon tracing algorithm [Arv86] can be used to determine the sound propagation by Monte Carlo raytracing. To establish paths of specular reflections, the metropolis algorithm can be used which searches the space of all possible paths using a Markov process of path mutations [VG97]. However, the Monte Carlo techniques can miss important paths due to undersampling. This problem can be circumvented using beam tracing techniques [FCE*98, FMC99, MF00, TFN*01]: Beams covering all specular sound paths are propagated into the scene by recursively clipping to portals and reflecting from walls. A data structure is maintained from which the relevant sound paths can be reconstructed efficiently for any observer position.

The beam tracing technique as well as Savioja's image source technique perform a dynamic selection of sound sources/paths according to the observer position, similar to our technique. However, these methods aim at an efficient extraction of (higher order) specular sound transport paths while our method aims at rendering scenes with many sound sources (not necessarily created by global sound propagation). Our method is not very efficient for rendering highly specular effects because the precomputed data structures account for the spatial location of the sound sources only while directional effects have to be resolved dynamically. The strength of our proposal is the ability to deal with scenes containing a vast amount of sound emitters, such as a football stadium.

Recently, Tsingos et al. [TGD03a, TGD03b] proposed a multi-resolution sound rendering technique related to our technique: A dynamic clustering algorithm according to a perceptual metric is used to sample a set of sound sources. The proposed perceptual metric is more effective than simple stochastic sampling. However, the sampling algorithm cannot handle complex scenes: The largest interactive example scene consists of 225 sound sources.

The last basic approach for simulating sound propagation is an ad-hoc approach, often used in computer games: *Parameterizable digital filters* are used to model room characteristics by manually choosing matching parameters for different parts of the scene and blending between them during walkthrough [Mic02].

A lot of software systems have been devised to integrate sound simulation with virtual reality. One of the first publications on sound rendering in the graphics community is the paper by Takala and Hahn [TH92]. Two recent systems are the DIVA system [Sav99] and the blue-c system [NSG02]. The DIVA system combines wave-based and image source methods to render scenes like a virtual orchestra performance in real-time. The blue-c system is a telepresence system based on a VR cave setup. It uses parameterizable reverberation processors for spatialized sound rendering.

Beyond these topics, there has also been a lot of work in the area of the simulation of the sound sources themselves (e.g. using solid dynamics simulations) and in the area of auditory display (e.g. head related transfer functions, multichannel audio, wave field reconstruction). For a complete review, we refer the reader to a survey article such as [FJT02].

As already stated in the introduction, our approach is also related to multi-resolution point sample rendering [PZvB*00, RL00, WFP*01, WS02]. Our algorithm uses a similar data structure to collect suitable sample sound sources. Then, the overlay of sound waves at the observer is reconstructed from these sample sources, reminiscent of image reconstruction in visual point-based rendering. Our rendering algorithm is also similar to final gathering as described in the global illumination literature [Rei92, Jen96, Kel97, PPD98].

3 The Sampling Algorithm

3.1 Overview

In this section, we describe our sampling algorithm for sound rendering. We start the description with a brief overview of the main components of our sound rendering subsystem (Figure 1): At a rough scale, the system consists of two threads, a sampling thread and a mixing thread. The sampling thread chooses a set of representative sound sources for a given observer position. These sources are handed over to the mixer thread which blends together the

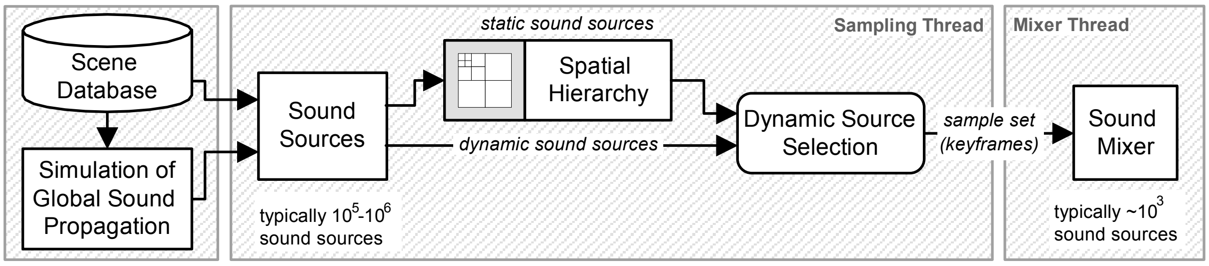


Figure 1: *Overview of the sound rendering system. The set of sound sources consist of sources specified in the scene data base and secondary sources due to global sound propagation. Dynamic importance sampling is used to extract suitable sample sets which are handed over to a mixer thread. A spatial hierarchy is used for static sources to accelerate the selection process. The sample sets are played back by a mixer thread.*

waveforms in real-time. The sampling thread provides keyframes, i.e. lists of the active sound sources along with volume and phase information (for Doppler effect and stereo reproduction) for each source. This mixer architecture is a standard architecture in the sound rendering literature [SHL*97, Sav99, TH92].

The sampling thread consists of a dynamic importance sampling algorithm that chooses sound sources with probability proportional to their effective volume at the given receiver position. It adapts the sampling distribution when the sources or the receiver change their position (or other characteristics). For static sound sources, an additional spatial hierarchy is used that assures that the sampling costs are independent of the number of sound sources. In the forthcoming subsections, we will focus on this sampling algorithm as it is responsible for rendering quality and efficiency.

3.2 Settings

The input to our algorithm is a set of point sound sources s_i, $i = 1\ldots n$. Each source s_i is assumed to emit a periodic sound signal $f_i(t)$. Additionally, each sound source is assigned a directional emission characteristic $e_i(d)$ which assigns different emission intensities to different outgoing directions d. This accounts for directional emission of primary sources as well as for directional reflection effects (glossy reflections) of secondary sources. The observer is also assigned a directional receiver characteristic $e_r(d)$. Overall, we obtain an attenuation coefficients a_i that describes how much intensity of a sound source s_i is received by an observer located in a distance of r_i:

$$a_i := \frac{e_i(d)e_r(d)v_i(d,r)}{r_i^2} \tag{1}$$

The v_i term accounts for the visibility of the source from the receiver. The received signal f_r is given by:

$$f_r(t) := \sum_{i=1}^{n} a_i f_i(t - r_i / v_s) \tag{2}$$

with r_i being the distance to source i and v_s being the speed of sound. To model frequency dependent effects, such as different specularity of reflection/emission depending on the signal frequency, we prefilter the sound sources and split them into frequency bands. Each frequency band is treated as a separate sound source with a potentially different characteristic: The terms e_i, e_r, v_i can be modified for each source according to the represented frequency band (see e.g. [TG97] for a treatment of frequency dependent occlusion).

3.3 Stochastic Sampling

The task of the sound renderer is the evaluation of Equation (2). This must be done at high update rates, typically 44,100 times per second, and usually for at least two receivers (stereo). For a large numbers of sound sources n, this is not possible in real-time. According to our experiments, it is possible to mix about 2000 sound sources in real-time on a contemporary PC in software (16Bit 44,1Khz, mono output, Direct Sound 9.0 [Mic02]), which can be easily exceeded in certain situations. Thus, we need a more efficient sampling strategy.

It is a well known fact from statistics that large sums like Equation (2) can be approximated efficiently using smaller sample sets. Therefore, it is not necessary to sum up all sound sources to obtain a realistic result [Gla95]. Instead, we choose k sample sources from the n sources with probability p_i for source s_i. Let π: $\{1..k\}\to\{1..n\}$ be the outcome of the random selection. The received signal f_r is then approximated by

$$\tilde{f}_r(t) := \frac{n}{k}\sum_{i=1}^{k} \frac{a_{\pi(i)} f_{\pi(i)}(t)}{p_{\pi(i)}} \tag{3}$$

Obviously, the expected value of this estimator is f_r for any sampling distribution p (with nonsingular p_i for all i=1..n). How large is the error introduced by this approximation? The central limit theorem states that the distribution of the estimator converges to a normal distribution with expected value f_r and standard deviation $O(\sigma_p/\sqrt{n})$. σ_p is the standard deviation of $a_{\pi(i)}f_{\pi(i)}(t)/p_{\pi(i)}$ and depends on the sampling distribution p.

Our goal is to keep σ_p as small as possible. This is a typical application for an importance sampling strategy [Gla95]: An optimal choice for the sampling distribution would be probabilities p_i proportional to $a_i f_i(t)$. However, calculating these weights is as expensive as computing the solution to Equation 2. The elements to be sampled are products of two factors: a_i and $f_i(t)$. Thus, we use a_i as sampling weight, neglecting $f_i(t)$:

$$p_i := \frac{a_i}{\sum_{i=1}^{n} a_i} \qquad (4)$$

The attenuation coefficients a_i depend only on the position of the observer (and probably on that of the sound sources, in dynamic scenes) and thus change slowly. Therefore, we use them as weights for the importance sampling[1]. Usually, it is sufficient to update the a_i values at a low frequency whereas the sample values $f_i(t)$ must be updated at several ten thousand Hz. Thus, the costs for estimating the sampling distribution can be reduced drastically while maintaining a considerable variance reduction. It is especially effective if all signals f_i are normalized prior to sampling (i.e. scaled to have the same average volume). The rescaling factors then have to be included in the a_i terms for compensation.

3.4 Dynamic Importance Sampling

We propose the following sampling algorithm to implement the importance sampling strategy: It takes a list of sound sources as input, consisting of sound buffers f_i (e.g. created from wave files) and weights a_i. The sampling algorithm examines the weights a_i of the sound sources periodically (typically 10-20Hz) and passes the mixing parameters (a list of sound sources with phase and volume) to the mixer thread. Between these *keyframes*, volume and phase (time shift) are interpolated by the mixer. The sampling algorithm starts with random importance sampling to generate the first keyframe: k sources are selected with probability p_i proportional to a_i. In contrast to the purely independent sampling approach, we additionally make sure that any sound source may be chosen only once. For large n, this does not make a significant difference and thus, the asymptotic convergence rate is retained. However, if n is closer to k, we can avoid to chose the same source multiple times and thus obtain a better sample set for smaller scenes. In our implementation, we use a simple divide and conquer algorithm that recursively divides the list of all sources in half and assigns samples to the left and right half proportional to the summed a_i-values in the corresponding parts. When only one source is left, random importance sampling is used for selection. This algorithm uses $O(n)$ time. When the observer moves through the scene or if sound sources are moving or changing their directional emission or their current average volume, the weights a_i change and a resampling must be performed to retain a good sample set with low sampling error. There are different opportunities to adapt the sampling distribution dynamically:

[1] Note that a similar approach is also used in visual point-based rendering where the probability of choosing sample points is usually proportional to the projected area, i.e. declines with $1/z^2$.

Full resampling: The easiest way to keep the distribution up to date is to choose a completely new sample set periodically. However, this strategy has some drawbacks: Due to its stochastic nature, the estimate is not exact. If we switch between random sample sets periodically, the static approximation error will turn into perceptible noise artifacts. Even if we blend between different approximations at a lower frequency (say every few seconds), the differences might still be audible, especially for small sample sizes k.

Stochastic diffusion: In order to avoid blending artifacts, a better strategy is to exchange only a part of the sample set at a time, say at most $q = 20\%$ per second. To do this, we chose a small number of *q/update_frequency* sound sources to be faded out at every key frame. After the fadeout time (typically 0.05-0.2 sec), the sound source is replaced by a new one that is chosen among the unused sources by importance sampling. The new source is then faded in during a similar time period. This strategy comprises a trade-off: If we use very short fading times and allow a large fraction of sound sources to be in "fading"-state, we obtain a fast adaptation to the current distribution. On the other hand, this also leads to more artifacts due to the changing sample set.

Adaptive diffusion: To improve on this, we would like to adapt the rate of change to the rate of changes in the weights a_i. We use the selected sample sources to detect the changes: We store the a_i value at the time of sampling for each sample source. When the value is changing, we check if the value has become smaller or larger by a factor of ε, with an ε typically in the range of 1.1 - 1.5. When the a_i value has become too small, the source is faded out and replaced by a new one obtained by importance sampling. If the value is too large (i.e. the source has become even more important), a random source out of the other sample sources is faded out and replaced by a new one. A random source is faded out in order to avoid a systematic bias. As all representative sources have been determined using importance sampling, they can be considered to be of similar importance so that we can fade out a random one.

3.5 Hierarchical Sampling

The dynamic sampling strategy proposed in the last section has still an important drawback: It requires $\Theta(n)$ time to do the resampling at every keyframe. Thus, it is not applicable to large scenes with potentially millions of sound sources. In this section, we propose a data structure to enhance the resampling speed for static scenes. Static means that all sources must have a fixed position and we can only account for a fixed average volume (the dynamic strategy can also take into account the change of the average volume of the sound sources over time). This does not mean that the weights a_i are constant as they still depend on the observer position, the visibility, and on the direction vector to the observer (for directional emission).

The dependency on the position and overall volume of a sound source as well as the receiver characteristic e_i can be compensated by the use of a spatial hierarchy: In a preprocessing step, we build an octree on the sound sources by dividing a bounding cube of the scene recursively until every octree node contains only one source. After that, a representative source is chosen for each inner node: Starting at the ancestors of the child nodes, a representative source is obtained by choosing one of the representative sound sources of the (up to) 8 children with a probability proportional to their average volume. The volume of the representative source is set to the sum of the volumes of all sources it represents.

If all the signals played by all sound sources in the octree were distinct and had the same length, we could also store the sum of all signals in the inner nodes rather than only a random sample. In such a case the memory demands for the inner nodes would be in the same range as that of the original sources. However, the sound sources are usually only instances of only a few (but space consuming) different base signals with slight variations in phase, volume, and playback frequency. In this case, the blow up in terms of memory demands for premixing all possible combinations is prohibitively large. Thus, we only store a reference to a single representative sample source.

After the precomputation of the spatial hierarchy, the relevant sound sources can be extracted efficiently using a priority based tree traversal: We start by adding the root node to a priority queue. The priority is given by the volume of the sound source multiplied by the distance attenuation factor and by the directional receiver characteristic (the latter corresponds to view frustum culling in visual rendering). The algorithm does the following iteration until k sources are found: The most important (loudest) node is removed from the queue and its children are added again to the queue[2]. When the queue contains k elements, the recursion is stopped and the sources in the queue are output as sample set. This implements an importance sampling strategy with preselected representatives in the octree. Additionally, the regular structure of the octree also leads to a spatial stratification of the sample set [Gla95]. For image generation, stratification of point samples is very important to reduce the costs due to oversampling [WS02]. For sound rendering, the benefits are not as obvious. It is possibly useful for stereo and multi-channel reproduction where single sound sources can be located in space by phase/volume differences.

The time complexity of the octree based sampling algorithm is fully output-sensitive. Assuming non degenerated octrees, i.e. every node has at least two children, the running time is $O(k \log k)$ for a sample set of k sound sources. Thus, it can be applied to highly complex scenes with millions of sound sources. Using hierarchical instantiation of the data structure (as applied in point rendering [WFP*01]), even billions of sound sources can be handled with reasonable time and memory demands. The drawback of the technique, besides being limited to static scenes, is that it can only consider the volume, the distance attenuation and the directional sensitivity of the *receiver* for choosing the sample set. Visibility and the directional *emission* characteristic of the sound sources are neglected. Thus, we end up with a bad sample set (i.e. a high variance of the estimate) for scenes with much occlusion or a strong directionality of emission. This is especially a problem for secondary sound sources that have been reflected specularly.

In order to improve on this, we combine the octree strategy with dynamic importance sampling: As choosing a sound source with one of the proposed sampling algorithms is much less expensive than actually mixing a source, it is possible to choose a much larger set of candidate sources from the octree. Theses candidates are then fed into the dynamic sampling algorithm that is able to consider occlusion and the emission characteristics, too. If we want to mix e.g. 500 sound sources, it is no problem to choose from 5,000 or 50,000 candidates. Even if only every tenth or hundredth contributes significantly to our sample, we still obtain a low variance sample set. However, this means that the amount of samples that must be extracted from the octree grows with the percentage of occluded sound sources (similarly to point-based image generation) as well as with the directionality of emission of the sound. The second restriction is especially relevant in sound rendering: If the algorithm is used to auralize the result of a global acoustics simulation, it usually has to deal with directional sources stemming from highly specular reflection, which is typical for the reflection of higher frequency sound waves at larger objects. If a precise simulation of such effects is needed, algorithms like [FCE*98] are better suited and could be possibly combined with our approach to complement it.

4 Implementation

We have implemented a prototype sound rendering system to evaluate the proposed algorithm empirically. We have integrated the sound rendering technique into the framework of a walkthrough system that uses a point-based multi-resolution renderer to visualize complex scenes, taken from [WS02]. In this section, we discuss some details of our implementation.

[2] Please note that the representative sound sources have been chosen before using random sampling. Thus, the choosing the "loudest" sources is not biased. However, the sampling decision is fixed after preprocessing.

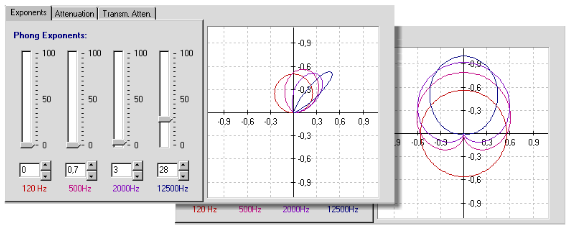


Figure 2: *Phong reflection (left) and emission (right).*

Emission and reflection properties: To model emission and reflection characteristics, we use the heuristic Phong-like model proposed in [TH92]: Primary sources are modeled using a $(1+\cos\alpha)^p$ law for an angle of α between the main direction of the source and the direction of emission. Secondary sources use a specular Phong reflection model with a directional intensity of $\cos^p\alpha\cdot\cos\beta$. Here, α is the angle between the reflection vector and the direction of emission and β is the angle between the surface normal and the direction of emission. The exponents as well as a linear scale factor can be specified for different spectral bands using a graphical user interface (see Figure 2). Linear interpolation is used to obtain values between the specified frequencies.

Sound Mixing: We have implemented two sound mixer variants. The first is a C++ implementation of the mixer task. It serves as reference implementation and thus it is used in all our examples. It has not been optimized for speed. We have also implemented a second variant that uses multiple DirectX “secondary sound buffers” [Mic02] to do the mixing. It also runs in software but, as part of the operating system, the DirectX sound mixing system has been optimized more thoroughly. It is usually about ten times faster than our reference mixer. However, the DirectX interface is not well suited for our application as the phase can only be controlled by specifying varying playback frequencies instead of exact explicit phase shifts, easily leading to phase drifts. Therefore, we did not use it as reference for our experiments. As this is only an interface problem (the effort for mixing should be the same), we believe that our mixing tasks could be performed at a similar speed using an optimized implementation.

Global sound propagation: We have implemented a simple simulator for global sound propagation. It uses a photon tracer [Arv86, Gla95, KSS68, SP94] that shoots random rays into the scene from the sound sources. The sources as well as the emission and reflection characteristics are sampled using importance sampling. Russian roulette is used to account for absorption. The Phong-reflection characteristics of the surface are treated as specular emission characteristics of secondary sources. Diffraction effects are not modeled. This simplistic simulation is of course far from being physically accurate. However, it can generate plausible reverberation effects that might be acceptable for applications such as computer games.

Current Limitations: The current implementation still lacks some features: Time dependent average volume calculation for use in the dynamic sampling algorithm has not yet been implemented; instead, a constant volume is assumed for the complete sound signal. For our test cases, this is of minor importance as the sound signals are quite short and of relatively uniform intensity. Occlusion has also not yet been implemented (the example scenes contain only little occlusion). Adding frequency dependent occlusion effects should be straightforward following the direction described in [TG97].

5 Results

We have tested our implementation on a dual Xeon 2.2Ghz workstation with a GeForce 4 Quadro graphics board. The dual processor setup has the advantage of providing better latencies for multi-threaded applications and allows us to run the graphics output and sound output module in parallel. This is especially advantageous for our prototypical C++ reference mixer. For the more efficient DirectSound implementation, a dual processor setup is not necessary. The results are demonstrated in the accompanying video. Using our reference C++-sound mixer, we were able to mix about 300 sound sources in real-time using one processor (150 sources with stereo reproduction, i.e. two receivers, 44.1 kHz, 16 bit). The DirectX sound mixer was able to mix about 2000 sound sources in real-time using one processor (1000 in stereo, 44.1 kHz, 16 bit, with resampling to varying playback frequencies enabled).

5.1 Example Scene 1: Many Sound Sources

Our first test case is a football stadium scene, taken from [WS02]. It shows 16,416 football fans, yelling, singing, and shouting (see Figure 3). Each football fan is assigned one of 13 prototype sound signals with a random phase shift of 300ms. All sources are in 16 bit, 44.1 kHz, mono format; the mixer outputs the same format in stereo, simulating two different receivers. Additionally, 20,000 secondary sources were generated by “photon”-tracing, using diffuse reflections only. The sampling was done using the octree strategy, without dynamic importance sampling. We tested sound rendering with sample sets of varying size. Using our C++-mixer, we were able to render the scene (including sound & graphics) in real-time with up to 150 sound sources. This approximation already yields plausible renderings but can still easily be identified as approximation. Using 2,000 sound sources, which is the maximum theoretical software mixing capacity of the reference platform (twice the mixing capacity of the DirectX mixer for two CPUs), permits a good approximation quality. In order to examine the speed-quality trade-off more thoroughly, we have made two tests: First, we have measured the l_2-error of the approximations (relative to the original version, i.e. using all sound sources). The result is shown in Figure 4: The absolute l_2-error decreases with $O(k^{-1/2})$ for k sample sources, as predicted. However, this consideration is problematic: As the modeling of our example scene is rather simplistic (just randomly phase shifted copies of some model samples), the solution for an infinite number of football fans converges towards a constant function (the sum of the integral over the model samples). Therefore, the reduction of the l_2-error corresponds mostly to a reduction of the amplitude of the mixing result (which is mathematically correct but not wanted in our application). The relative error (i.e. comparing renormalized results) does not converge according to the central limit theorem, thus yielding worse results (Figure 4, red curve). However, the relative l_2-error does also not describe well the perceptual match of our approximation. In order to determine at which number of sound sources a sufficient perceptual quality is reached (for normalized volume), we have conducted a small user study. People from our institute were provided with different approximations. No indication of the approximation settings were provided to the subjects (blind review via email, random file order). They were asked to quantify the deviation of the different approximations by numbers (1 = no audible difference, 3 = audible differences, 5 = bad approximation) using conventional PC-speaker systems (as representative for a PC gaming scenario, which would be a typical application). The results are shown in Figure 5: Approximations with up to 128 samples were clearly distinguished as bad approximations. The effect may still sound somehow believable; however, the subjects were asked to determine the differentiability to the original sound, which is obvious for small sample sizes. 2048 samples already provided a satisfactory approximation quality for most subjects. The result is not indistinguishable from the original but might be sufficient for gaming or virtual reality applications. For 8192 and more samples, the approximation quality was very close to the original. Please note that only 8 subjects have participated in the experiment. Thus, the results are not statistically significant in a strict sense but just indicate a rough impression of the perceived approximation quality. The sound files used in our evaluation are included in the supplementary material accompanying this paper.

5.2 Example Scene 2: Global Sound Propagation

The second test scene is a violin performance in a presentation room (Figure 3c). The scene contains only one primary sound source. 50,000 additional secondary sound sources are generated by our global sound propagation simulator. The accompanying video shows four different variants: Ideal diffuse reflection (with two different speeds of sound), specular reflection (Phong exponent 20) and a combination of both: Phong exponents ranging from diffuse to specular for increasing frequency. For the last example, the sound signal of the violinist has been prefiltered into 8 octave bands. The simulator and the sampling algorithm treat all 8 bands as different input sound sources. Again, we observe a satisfactory approximation quality using 2000 sound sources. For smaller sample sizes, the quality is worse, but may be acceptable for simple gaming applications. A large sample size is especially required for the last case of sampling multiple frequency bands. The random selection of frequency bands leads to a larger deviation (especially for the benchmark scene in which only some frequency bands contain significant signals). Further experiments are necessary to quantify this effect.

To examine the approximation error, we have measured the impulse response of the approximation (i.e. the result of sampling a single Dirac pulse). Figure 6 shows the result for 150, 2000, and 20000 sample sources for diffuse and specular reflection (Phong exponent 20). Differences between specular and diffuse reflection can be observed: As expected, the specular version tends to show a few distinguished echoes while the diffuse version shows a more continuous impulse response. The stochastic convergence can also be seen: The 150-sources diffuse version approximates the final function only roughly, still showing temporal undersampling. This is consistent with the observations in the example scenes: The time spacing is too small to distinguish individual sound sources. However, the resulting reverberations do not sound like a natural reverberation. The 2000 sample sources version comes already quite close to the 20000 sources version and should be sufficient for interactive applications. This is also consistent with our perceptual experience. In the specular case, the 150 source impulse response is merely random. The 2000 source version approximates the 20000 source version up to some noise (especially in between the main echoes), leading to an acceptable approximation.

6 Conclusions and Future Work

We have presented a new sound rendering algorithm for scenes with a large number of sound sources. It is based on a stochastic sampling approach, as known from visual point-based multi-resolution rendering. It uses a combination of hierarchical sampling and dynamic importance sampling to select suitable sample sets according the observer position. The running time does not depend on the number of sound sources but only on the ratio of visible and invisible sources and on the directionality of the sound emission. The algorithm can also be used to auralize simple simulations of global sound propagation. Here, the main restriction is that the algorithm can only deal with diffuse reflections and glossy reflections of moderate specularity. For highly specular reflections, the sampling costs become too large. As the algorithm is quite easy to implement as a complement to a visual multi-resolution rendering algorithm and already delivers plausible results at a small sample size, we believe that it is useful for sound rendering in computer games and VR-applications. When the complete processing power of a contemporary PC-system is used for sound rendering, we already obtain results that come close to the exact solution of mixing all sound sources.

The quality of the output depends on the number of sound sources that can be mixed simultaneously. Thus, we would like to improve the mixer performance in future work. A promising direction for mixing a very large amount of sound sources is the utilization of the GPU as a sound mixer [TGD03b]. Considering the raw pixel fill rate of a contemporary GPU, it should be possible to mix more than 10,000 instantiated sound sources in real-time using pixel shaders and additive blending. The perceptual convergence could probably enhanced substantially by using a perceptually based importance metric during sampling, as proposed by Tsingos et al. [TGD03b] instead of the simple "average amplitude" heuristic. It would also be interesting to examine whether culling of sound sources based on perceptual masking [TGD03b] could be included.

Acknowledgments

The authors wish to thank Alexander Ehlert, Stefan Guthe, Michael Hauth and Timo Schairer for valuable discussion. The 3d scenes have been created by Amalinda Oertel and Peter Klingen using the Poser animation package. The sound samples have been provided by Martin Rüther and Andreas Galezka.

References

[AB79] Allen, J.B., Berkley, D.A.: Image method for efficiently simulating small-room acoustics. In: *Journal of the Acoustical Society of America*, 65(4), 943-950, 1979.

[Arv86] Arvo, J.: Backward Ray Tracing. In: *Developments in Ray Tracing, SIGGRAPH `86 Course Notes*.

[Bor84] Borish, J.: Extension of the Image Model to Arbitrary Polyhedra. In: *Journal of the Acoustical Society of America, 75(6)*, 1827-1836, 1984.

[DKP01] van den Doel, K., Kry, P.G., Pai, D.K.: FOLEYAUTOMATIC: Physically-based Sound Effects for Interactive Simulation and Animation. In: *SIGGRAPH 2001 Proceedings*.

[FCE*98] Funkhouser, T., Carlbom, I., Elko, G., Pingali, G., Sondhi, M., West, J.: A Beam Tracing Approach to Acoustic Modeling for Interactive Virtual Environments. In: *SIGGRAPH 1998 Proceedings*.

[FJT02] Funkhouser, T., Jot, J., Tsingos, N.: "Sounds Good to Me!", Computational Sound for Graphics, Virtual Reality, and Interactive Systems. In: *SIGGRAPH 2002 Course Notes*.

[FMC99] Funkhouser, T., Min, P., Carlbom, I.: Real-Time Acoustic Modeling for Distributed Virtual Environments. In: *SIGGRAPH 99 Proceedings*.

[Gla95] Glassner, A. S.: *Principles of Digital Image Synthesis*. Morgen Kaufmann Publishers, 1995.

[Jen96] Jensen, H.W.: Global Illumination using Photon Maps. In: *Rendering Techniques '96*, Springer, 1996.

[Kel97] Keller, A.: Instant Radiosity. In: *SIGGRAPH 97 Conference Proceedings*.

[KSS68] Krockstadt, A., Strom, S., Sorsdal, S.: Calculating the acoustical room response by the use of a ray tracing technique. In: *J. Sound and Vibrations, 8(1)*, 1968.

[Mic02] Microsoft Direct X9 SDK, 2002. *http://msdn.microsoft.com/directx*

[MF00] Min, P., Funkhouser, T.: Priority-Driven Acoustic Modeling for Virtual Environments. In: *Eurographics 2000 Proceedings*.

[NSG02] Naef, M., Staadt, O., Gross, M.: Spatialized Audio Rendering for Immersive Virtual Environments. In: *Proceedings of the ACM symposium on Virtual reality software and technology*, 2002.

[BSG02] O'Brien, J.F., Shen, C., Gatchalian, C.M.: Synthesizing Sounds from Rigid-Body Simulations. In: *SIGGRAPH 2002 Proceedings*.

[BCE01] O'Brien, J.F., Cook, P.R., Essl, G.: Synthesizing Sounds from Physically Based Motion. In: *SIGGRAPH 2001 Proceedings*.

[PPD98] Paquette, E., Poulin, P., Drettakis, G.: A Light Hierarchy for Fast Rendering of Scenes with Many Lights. In: *EUROGRAPHICS 98 Proceedings*.

[PZvB*00] Pfister, H., Zwicker, M., van Baar, J., Gross, M.: Surfels: Surface Elements as Rendering Primitives. In: *SIGGRAPH 2000 Proceedings*.

[Rei92] Reichert, M.C.: *A Two-Pass Radiosity Method to Transmitting and Specularly Reflecting Surfaces*. M.Sc. thesis, Cornell University, 1992.

[RL00] Rusinkiewicz, S., Levoy, M.: Qsplat: A Multiresolution Point Rendering System for Large Meshes. In: *SIGGRAPH 2000 Proceedings*.

[SHL*97] Savioja, L., Huopaniemi, J., Lokki, T., Väänänen, R.: Virtual Environment Simulation - Advances in the DIVA Project. In: *Proc. Int. Conf. Auditory Display (ICAD)*, 1997.

[Sav99] Savioja, L.: *Modeling Techniques for Virtual Acoustics*, Doctoral thesis, Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, Report TML-A3, 1999.

[SP94] Sillion, F., Puech, C.: *Radiosity and Global Illumination*, Morgan Kaufman, 1994.

[TH92] Takala, T., Hahn, J.: Sound Rendering. In: *SIGGRAPH 92 Proceedings*.

[TFN*01] Tsingos, N., Funkhouser, T., Ngan, A., Carlbom, I.: Modeling Acoustics in Virtual Environments Using the Uniform Theory of Diffraction. In: *Siggraph 2001 Proceedings*.

[TGD03a] Tsingos, N., Gallo, E., Drettakis, G.: Perceptual Audio Rendering of Complex Virtual Environments. INRIA technical report #4734, 2003.

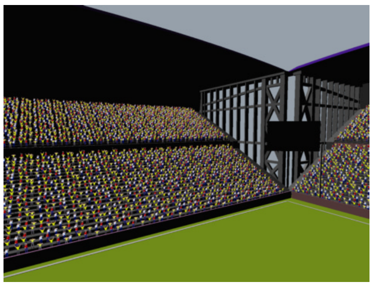
[TGD03b] Tsingos, N., Gallo, E., Drettakis, G.: Breaking the 64 spatialized sources barrier. In: *Gamasutra Audio Resource Guide* (www.gamasutra.com), May 2003.

[TG97] Tsingos, N., Gascuel, J.: Soundtracks for Computer Animation: Sound Rendering in Dynamic Environments with Occlusion. In: *Graphics Interface '97 Proceedings*.

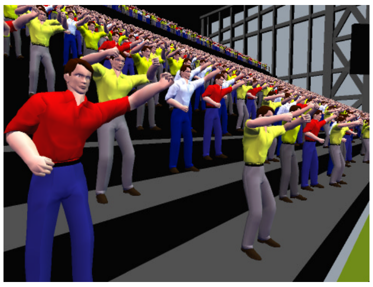
[VG97] Veach, E., Guibas, L.J.: Metropolis light transport. In: *SIGGRAPH 97 Proceedings*.

[WFP*01] Wand, M., Fischer, M. Peter, I., Meyer auf der Heide, F., Straßer, W.: The Randomized z-Buffer Algorithm: Interactive Rendering of Highly Complex Scenes. In: *SIGGRAPH 2001 Proceedings*.

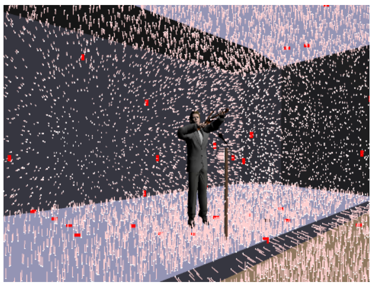
[WS02] Wand, M., Straßer, W.: Multi-Resolution Rendering of Complex Animated Scenes. In: *Eurographics 2002 Proceedings*.



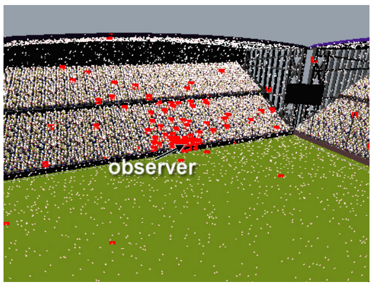
(a) football stadium scene, overview



(b) football stadium scene, close-up



(c) violin performance scene,
with secondary sources (light red) and sample set (red)



(d) sound source distribution for image (b)
sample set of 150 sources shown in red

Figure 3: *Screenshots from the example scenes. See the accompanying video for sound and animation.*

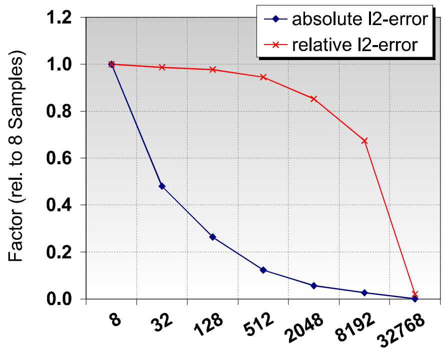


Figure 4: Measured L_2*-error for the stadium scene.*

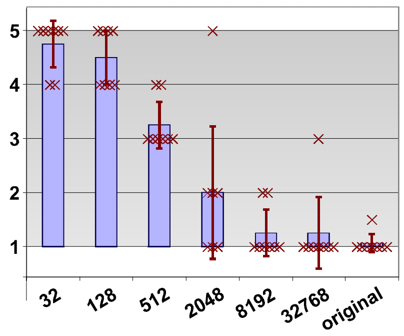
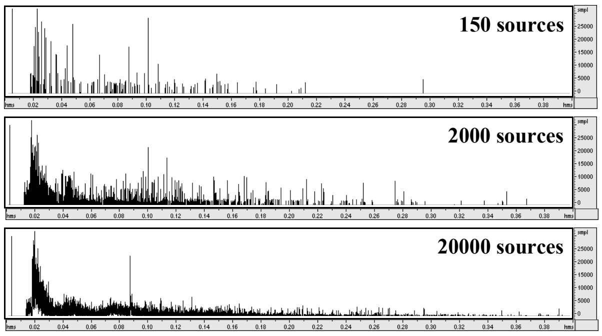
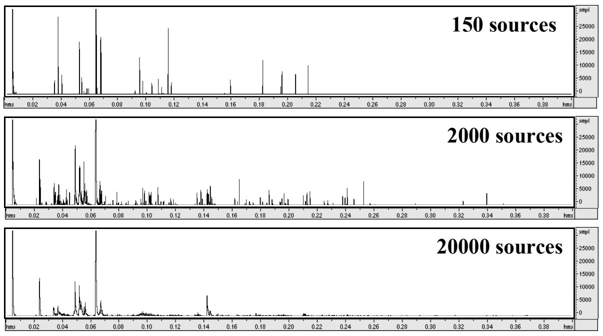


Figure 5: Perceptual difference (relative error).



(a) diffuse reflection



(b) specular reflection

Figure 6: *Impulse responses for the "violin performance" scene (Figure 3c) for two different material settings. The x-axis is the time axis (overall time 400msec), the y-axis shows the received response to a single impulse (normalized). Figure (b) shows specular reflection with a Phong exponent of 20, leading to more distinguished echoes.*