A Real-Time Sound Rendering Algorithm for Complex Scenes

Michael Wand, Wolfgang Straßer

WSI-2003-5
ISSN 0946-3852
July 2003

Technical Report

WSI/GRIS,
University of Tübingen
Sand 14
D-72076 Tübingen, Germany

Email: {wand, strasser}@gris.uni-tuebingen.de
WWW: http://www.gris.uni-tuebingen.de

A Real-Time Sound Rendering Algorithm for Complex Scenes

Michael Wand Wolfgang Straßer

WSI/GRIS, University of Tübingen
Sand 14, 72076 Tübingen, Germany
{wand, strasser}@gris.uni-tuebingen.de

Abstract

We present a novel output-sensitive algorithm for sound rendering of complex scenes, i.e. scenes that contain a large amount of sound sources. It allows walkthroughs of complex sound environments (such as a football stadium) in real-time and can be used for observer-dependent auralizations of global sound propagation.

The algorithm is based on a stochastic sampling strategy similar to point based techniques used for rendering images. Typical applications of the technique are in virtual reality and computer games, especially to complement established output-sensitive algorithms for rendering visual content.

1 Introduction

One of the important long term goals of interactive computer graphics is virtual reality. Many applications in entertainment and computer games as well as in simulation and computer based training demand for a faithful simulation of artificial environments. Sound is an important aspect of most natural environments and should be included in a realistic simulation. It provides important spatial cues and has an especially strong effect on the emotional perception of a scene. Thus, a lot of research has been done on sound rendering in three dimensional scenes. Special attention was paid to the physically based simulation of sound sources [4,16,17] and the propagation of sound in complex environments [1,5,22,25]. A lot of work was also done on auditory display [6] and on software systems that integrate sound rendering with virtual reality applications [15,22].

In this paper, we consider the problem of real-time sound rendering for scenes containing a large amount of sound emitters. This is an important problem in many applications: Consider for example computer games showing complex crowd simulations, such as a football stadium in a sports game or an army in a strategic simulation. Another example could be 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 the simulation of the interaction of a few sound sources with complex environments. However, settings with a large number of sound emitters have not yet been considered specifically.

Our algorithm is based on point sampling: The 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 used to obtain estimates with low variance. For static scenes, an additional spatial hierarchy is precomputed that allows the extraction of sample sets in output-

sensitive time, independent of the number of sound emitters. Statistical arguments guarantee the stochastic convergence to the exact solution.

The 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. Examining the solution, it places a dense set of secondary sound sources in the scene, along with phase, volume and directional emission information. The sampling algorithm is then used to perform a real-time observer dependent "final gathering", as known from the image synthesis literature [8,11,23].

We have implemented a prototype virtual reality system that allows walkthroughs of complex scenes using the proposed algorithm for sound rendering and point-based multi-resolution rendering [29] for rendering the visual presentation. We examine two example scenes: A football stadium with 16,000 fans, singing and yelling and a room with a violinist. Both scenes contain secondary sound sources obtained from a simple raytracing-based reverberation simulator.

2 Related Work

In this section, we review work related to our approach, especially concerning the simulation of sound propagation. 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, multi-channel audio, wave field reconstruction). For a complete review, we refer the reader to a survey article such as [6].

The propagation of sound can be simulated 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 [22]. 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 [12]. A classic method is the image source algorithm [1,3]: Specular reflection paths are constructed by considering the source and the receiver and each reflecting polygon in the scene. For each polygon, an image source, located behind the polygon, is constructed that serves as substitute for the indirect sound path. This scheme is carried on recursively for the image sources obtained to calculate higher order reflections. This technique can enumerate all ideal specular reflection paths. However, the combinatorial complexity of this process renders a straightforward adoption of the algorithm to compute higher order reflections in complex environments infeasible. Savioja et al. [21] use geometric data structures to accelerate the computation of the image source. Additionally, only one image is generated for sources close to each other. During walkthrough, relevant image sources are selected automatically. Another possibility for global sound propagation is the usage of path tracing techniques: Variants of the photon tracing algorithm [2] can be used to calculate 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 [27]. A problem of monte-carlo techniques is aliasing: The space of all energy exchange paths is only sampled and important paths can be missed. This problem is circumvented by beam tracing techniques [5, 7, 14, 25]: An adjacency data structure of cells and portals is constructed. Then, 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 technique can also be generalized to deal with diffraction at edges [25].

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 stemming from global sound propagation). Our method is not very efficient for rendering highly

specular effects, especially not for higher order specular reflection. The strength of our proposal is the ability to deal with scenes containing a vast amount of sound emitters, such as a football stadium.

The last basic approach 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 (e.g. rooms) of the scene and blending between them during walkthrough [13].

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 [24]. They describe a sound rendering pipeline based on keyframes and discuss sound synthesis and propagation. Two recent systems are the DIVA system [22] and the blue-c system [15]. 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 reverb processors for spatialized sound rendering.

Our approach is also related to point sample rendering [18, 20, 28, 29]. Point based rendering algorithms for image generation try to reconstruct an image of a complex geometry from a small sample set of surface points. Our algorithm pursues the same goal for sound waves instead of images. The idea of final gathering is taken from the global illumination literature [19]: Many techniques such as photon mapping [10] or instant radiosity [11] use a coarse solution to the global illumination problem and then display the shadows that occur if the solution is treated as an extended light source. This leads to a drastic improvement of image quality at moderate costs. The same technique is applied by our sampling algorithm that serves as "final gathering" step for a global sound propagation simulation.

3 The Sampling Algorithm

3.1 Overview

In this section, we describe our sampling algorithm for sound rendering. First, we would like to give a brief overview of the main components (Figure 1): The rendering system consists of two threads, a sampling thread and a mixing thread. It expects a list of sound sources as input which can be specified explicitly in the scene description or generated by a simulation of global sound propagation. 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 waveforms in real-time. At the interface, keyframes are exchanged. They consist of lists of the active sound sources along with volume and phase (time shift) information for each source (for Doppler effect and stereo reproduction). This mixer architecture is a standard architecture in the sound rendering literature [21,22,24].

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 describe the algorithm more in detail.

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 re-

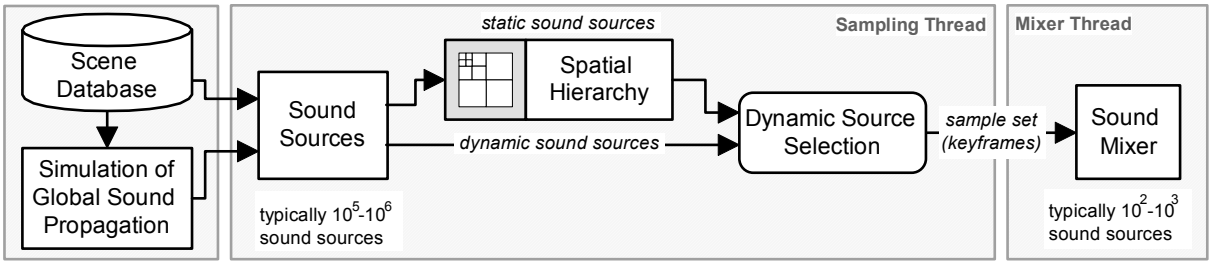


Figure 1: Overview of the sound rendering system. The set of sound sources consist of sources specified in the scene data base and secondary sources due to global sound propagation. Dynamic importance sampling is used to extract suitable sample sets which are handed over to a mixer thread. A spatial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

flection 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:

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

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

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

To model frequency dependent effects, such as different specularity of reflection or 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. [26] 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 [13]). However, this is not sufficient for two reasons: First, this leads to 100% CPU utilization and does not leave any room for other activities, such as graphics display. Second, the number of 2,000 sources can be exceeded easily in many situations, for example if a lot of secondary sources are used to simulate reverberations. 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 [8]. Instead, we choose k sample sources from the n sources with probability p_i for source s_i. Let π: $\{1..k\} \rightarrow \{1..n\}$ 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 ex-

pected value f_r and standard deviation $O(\sigma_p/\sqrt{n})$. σ_p is the standard deviation of $a_{\pi(i)} f_{\pi(i)}(t)/p_i$ and depends on the sampling distribution p.

Equation 3 is a typical case for an importance sampling strategy [8]: 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 $f_i(t)$:

$$p_i := \frac{a_i}{\sum_{i=1}^{n} a_i} \tag{4}$$

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 very slowly. Therefore, it is usually sufficient to update these 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 drastically reduced 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. A mixer thread is used to mix a sample of k sources in real-time. 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 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 the analysis, we assumed to choose all sample sources independently of each other to fulfill the requirements of the central limit theorem. However, in practice it is more convenient to 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 analysis is still applicable. 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. This can be performed in $O(n)$ 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:

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. Because of the $O(\sqrt{n})$ convergence order, it is virtually impossible to obtain approximations with an error below the level of perception. 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 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. The strategy can probably be refined by defining a replacement probability for all sources according to their change of weight and then continuously performing random replacements according to this probability. However, this has not yet been implemented but will be subject of future work.

3.5 Hierarchical Sampling

The dynamic sampling strategy proposed in the last section has still an important drawback: It still 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 and more. In this section, we propose a data structure to enhance the resampling speed for the special case of static scenes. A scene is called static if only the observer walks through the scene. All sources must have a fixed position and a fixed average volume[1]. 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 directed emission).

The dependency on the position and overall volume of a sound source 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 1:8 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), a better strategy would be to 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, playback frequency, and so on. In this case, the blow up in terms of memory demands for premixing all possible combinations is prohibitively large. Thus, it is usually only possible to store a single representative sample.

After the precomputation of a 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. The algorithm does the following iteration until k sources are found: The most important (loudest) node is removed from the queue and its children added again to the queue. 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 struc-

[1] Note that the dynamic strategy can also take into account the change of the average volume of the sound sources over time.

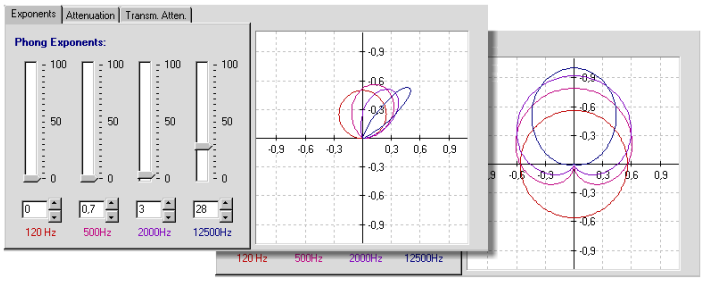


Figure 2: Phong reflection (left) and emission (right).

ture of the octree also leads to a spatial stratification of the sample set [8]. For image generation, stratification of point samples is very important to reduce the costs due to oversampling [29]. 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 $O(k \cdot \log k)$ 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 [28]), 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. 100 sound sources, it is no problem to choose from 1000 or 10,000 candidates. Even if only every tenth or hundredth contributes 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 (similar to point based image generation, as analyzed in [28]) 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 [5] 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 integrated the sound rendering into the framework of a walkthrough system that uses point base multi-resolution renderer to visualize complex scenes [29]. In this section, we discuss some details of our implementation.

Emission and reflection properties: To model emission and reflection characteristics, we use the heuristic Phong-like model proposed in [24]: 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 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 especially optimized for speed. We implemented a second variant that uses multiple DirectX "secondary sound buffers" [13] 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 own reference mixer. Nevertheless, we did not use it as reference for our experiments because the interface does not directly support a fine granular control of phase and a memory efficient instantiation of larger sound buffers.

Global sound propagation: We implemented a simple simulator for global sound propagation. It uses a photon tracer [2,8,12,23] 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.

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 [26]. The simple "photon" tracer used for simulating the global sound propagation currently does not consider any diffraction effects. Diffraction could possibly be handled by placing secondary sources at edges, analogous to [25].

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 (i.e. 150 sources with stereo reproduction, 44 kHz, 16 bit) using one processor. The DirectX sound mixer was able to mix about 2000 sound sources (i.e. 1000 in stereo, 44 kHz, 16 bit) in real-time using one processor.

5.1 Example Scene 1: Many Sound Sources

Our first test case is a football stadium scene, taken from [29]. 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, 44.100 kHz, mono format, the mixer outputs the same format in stereo, simulating two different receivers. Additionally, 20000 secondary sources where 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 k = 20, 50, 150 and 2000 sound sources. Using our C++-mixer, we were able to render the scene (including sound & graphics) in real-time with up to 150 sound sources. The 2000-sources version could only be recorded offline. We included it in our examples as this is about the limit that could probably be handled in real-time with an optimized

implementation at full system utilization (twice the mixing capacity of the DirectX mixer for two CPUs).

The rendering quality for 20 and 50 sources is not satisfactory. The sampling nature of our approach is clearly audible. Using 150 sources, we already obtain a quite plausible result that may be sufficient for applications like computer games. The CPU-utilization of the (optimized) Direct Sound mixer was only about 16%, retaining enough resources for graphics and game logic. However, the 2000 sources version provides still a substantial improvement.

If we move slowly into the crowd of football fans, the adaptivity of the sampling algorithm can be examined. As shown in the accompanying video, the sampling set begins to concentrate on nearby football fans and the voices of individual fans become audible. In a close-up, we can clearly distinguish the voices of the fans next to us while still hearing the more distance sound of the complete stadium[2]. This effect is similar to multi-resolution rendering of images: Nearby objects appear in full detail while objects that are farer away are display at a coarser level of detail which is still sufficient for their reproduction.

5.2 Example Scene 2: Global Sound Propagation

The second test scene is a violin performance in a small presentation room (Figure 4c). The scene contains only one primary sound source. 50000 additional secondary sound sources are generated by our global sound propagation simulator. The accompanying video shows three different variants: First, we simulate diffuse reflections only with a different numbers of sample sources and different speed of sound. Again, the 150 sample sources provide a plausible effect while 2000 sound sources provide a much better quality. The second variant shows the same example but with all materials set to specular reflection of sound with Phong exponent 20. The third example combines both variants by setting different Phong exponents ranging from diffuse to specular depending on the frequency. 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. In the case of frequency dependent reflection, a higher sample rate is necessary. For 150 sources only, we encounter a strong variation in the frequency composition when walking around in the room. This is due to the fact that all frequency bands are sampled independently. Using 2000 sample sources a good quality is achieved.

Figure 4 shows measured impulse responses for the example scene. Figure (a-c) show the result of diffuse material settings with 150, 2000, and 20000 sample sources. Figure (d) shows the result of 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 nicely. The 150-sources 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 the perceptual observations.

6 Conclusions and Future Work

We presented a new sound rendering algorithm for scenes with a large number of sound sources. It is based on a stochastic sampling approach. It uses a combination of hierarchical sampling and dynamic importance sampling to select suitable sample sets according the observer position. For static scenes, the running time does not depend on the number of sound sources but

[2] Note that the sound in the video was recorded using dynamics compression in order to avoid large scale changes in volume.

only on the ratio of visible and invisible sources and on the directionality of the sound emission. For dynamic scenes, a strong reduction of the processing costs can still be obtain in comparison to mixing all sound sources in the scene.

The algorithm can also be used to auralize 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 even at a small sample size (and thus reasonably small CPU-utilization), we believe that it is well suited 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, independent of their original quantity.

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 [9]. Considering the raw pixel fill rate of a contemporary GPU, it should be possible to mix more than 10,000 sound sources in real-time using pixel shaders and additive blending. Other directions for future work are an optimization of the dynamic sampling strategy and an improvement of the sound propagation simulation, as noted earlier in the text.

Acknowledgments

The authors wish to thank Alexander Ehlert, Stefan Guthe, Michael Hauth and Timo Schairer for valuable discussion. The 3d models were designed by Amalinda Oertel and Peter Klingen and the sound samples were provided by Martin Rüther and Andreas Galezka. Thanks go also to Tobias Hüttner/EGISYS for providing the Poser animation software.

References

[1] **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.

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

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

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

[5] **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*.

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

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

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

[9] **Stefan Guthe**, *personal communication.*

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

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

[12] **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.

[13] Microsoft Direct X9 SDK.
`http://msdn.microsoft.com/directx`

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

[15] **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.

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

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

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

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

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

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

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

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

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

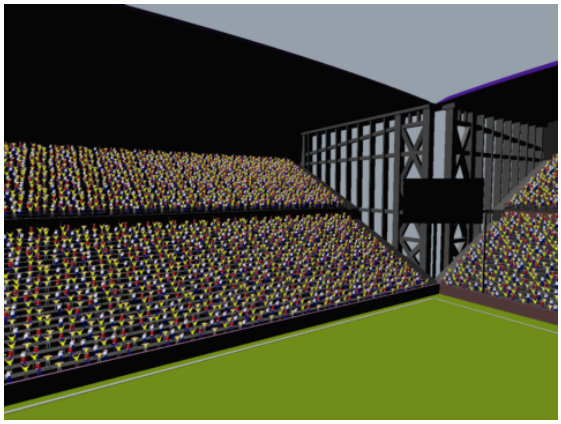
[25] **Tsingos, N., Funkhouser, T., Ngan, A., Carlbom, I.:** Modeling Acoustics in Virtual Environments Using the Uniform Theory of Diffraction. In: *Siggraph 2001 Proceedings*.

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

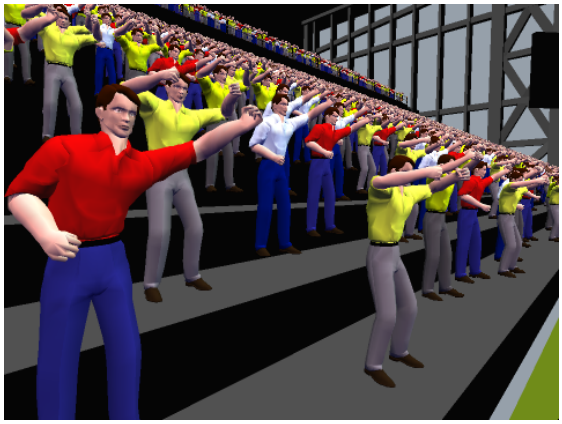
[27] **Veach, E., Guibas, L.J.:** Metropolis light transport. In: *SIGGRAPH 97 Proceedings*.

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

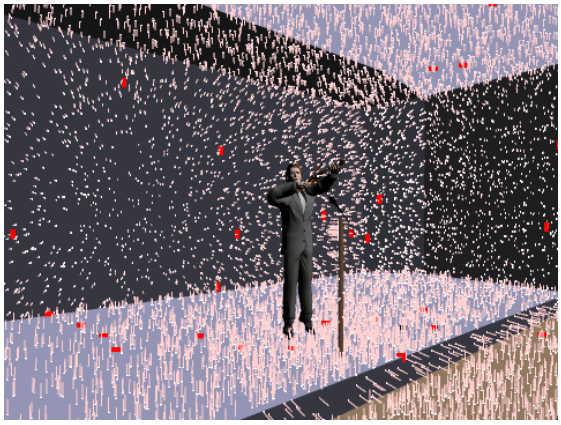
[29] **Wand, M., Straßer, W.:** Multi-Resolution Rendering of Complex Animated Scenes. In: *Eurographics 2002 Proceedings*.



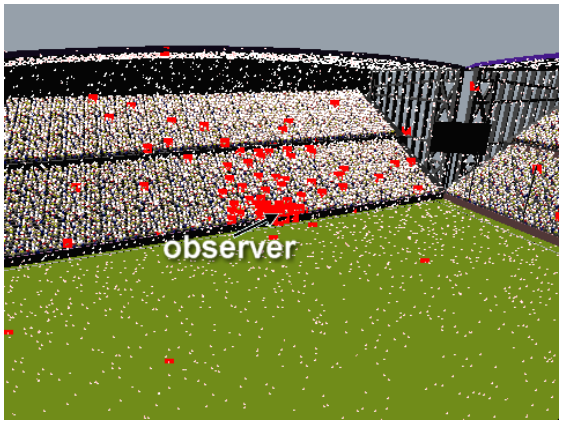
(a) football stadium scene, overview



(b) football stadium scene, closeup

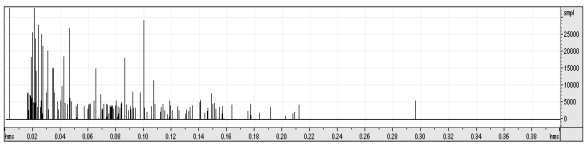


(c) violin performance scene, with secondary sources (light red) and sample set (red)

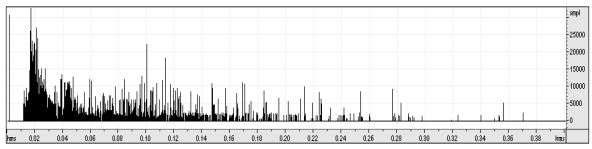


(d) sound source distribution for image (b) sample set of 150 sources shown in red

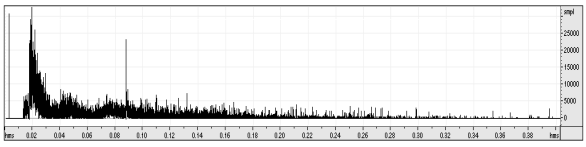
Figure 3: Screenshots from the example scenes. See the accompanying video for sound and animation.



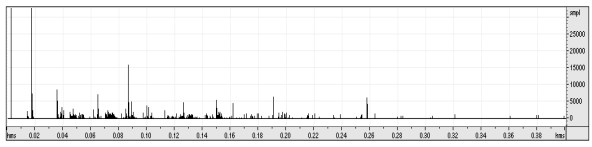
(a) diffuse reflection, 150 samples



(b) diffuse reflection, 2000 samples



(c) diffuse reflection, 20000 samples



(d) specular reflection, 2000 samples

Figure 4: 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. The reverberation part has been normalized. Note the convergence for increasing sample sizes from (a)-(c). Figure (d) uses shows specular reflection with a Phong exponent of 20, leading to more distinguished echoes.