GSOUND: INTERACTIVE SOUND PROPAGATION FOR GAMES

C. SCHISSLER, D. MANOCHA

The University of North Carolina at Chapel Hill
http://gamma.cs.unc.edu/GSOUND/

We present a sound propagation and rendering system for generating realistic environmental acoustic effects in real time for game-like scenes. The system uses ray tracing to sample triangles that are visible to a listener at an arbitrary depth of reflection. Sound reflection and diffraction paths from each sound source to the listener are then validated using ray-based occlusion queries. Frame-to-frame caching of propagation paths is performed to improve the consistency and accuracy of the output. Furthermore, we present a flexible framework, which takes a small fraction of CPU cycles for time-critical scenarios. To the best of our knowledge, this is the first practical approach that can generate realistic sound and auralization for games on current platforms.

1 Introduction

Auditory displays and sound rendering are frequently used to enhance the sense of immersion in computer games and related applications. Aural cues can be combined with visual cues to improve realism and the user's experience. In practice, aural cues are frequently used to create various emotions including fear, apprehension and horror. Games constantly improve the realism of graphics and AI but little work is done to enhance sound within games. Current sound systems use precomputed reverberation filters for specific locations to enhance the player's sense of space. While this approach is often a good enough approximation, these methods may not work well in dynamic scenes. More importantly, the effects of sound occlusion are often critical to how players perceive sound within a game. However, these phenomena are rarely modeled. In practice, realistic sound occlusion and propagation computation is regarded as a difficult problem. This is one of the main goals of sound propagation: to accurately simulate how sound interacts with an environment and is heard by a listener.

While sound propagation solutions exist, they are either too slow or not appropriate for dynamic scenes. This is the largest challenge in developing such a system for games. A sound propagation system for games must be able to produce reasonable output while consuming a minimum of CPU time and memory. In this paper, we present a novel sound propagation system that uses backwards ray tracing and propagation path caching to improve on existing work in this area.

2 Components of Sound Propagation

Sound propagation is usually modeled as a combination of several different phenomena. Any sound received by a listener can be split into 3 components: direct sound, early reflections and late reverberation. Direct sound is transmitted directly from a sound source to a listener.

Early reflections comprise the first echoes of a sound that reach a listener after direct sound arrives. These contributions can be produced by specular reflection off of surfaces in the scene, diffuse reflections, and diffraction about edges. Early reflections give auditory cues to a listener about the size of the environment and any salient features.

Late reverberation is the last component heard by a listener. It consists of many thousands of higher-order reflections that sum to produce a decaying reverb tail. Though we have a solution for reverb estimation we will focus on the early reflection and diffraction portions of our sound propagation system in this paper.

3 Prior Work

Much work has been done in the past on simulating realistic sound propagation in virtual environments. The earliest of these systems were developed as aids to architectural acousticians. These systems used ray tracing to estimate the acoustic qualities of concert halls and other rooms [Krokstadetal.1968; Vorländer 1989]. From this point, two classes of sound propagation methods emerged: numeric and geometric.

Numeric methods are accurate and produce true-to-life results but are generally too slow to be used in real-time systems. FDTD methods are very accurate but require enormous amounts of time and memory to simulate and therefore cannot be used in any real time system [Botteldooren 1995]. ARD makes improvements on FDTD methods by subdividing a complex environment into rectangular subdivisions in order to

reduce the simulation time. However, this method still requires significant time and resources [Raghuvanshi et al. 2009]. Direct-to-Indirect Acoustic Radiance Transfer borrows from radiosity ideas in graphics to perform simulation of diffuse reflection at real-time rates. However, this method still requires a large memory footprint and a long preprocessing step. In addition, scene geometry must be static, causing the approach to loose realism in situations with dynamic geometry [Antani et al. 2010].

Geometric methods generally use algorithms from graphics to model how sound propagates in an environment. These methods are generally more tractable than numeric simulation because they usually allow for moving sources, listeners, and dynamic scene geometry. This makes geometric sound propagation attractive for interactive game applications. The most accurate geometric approach is the image source method where sound sources are recursively reflected over every triangle in a scene to form images of the source positions. Reflection paths from the listener to the source are then validated in reverse via occlusion queries. While accurate, the image source method is extremely slow: the running time increases exponentially with the reflection depth [Allen and Berkley 1979].

Other geometric approaches use scene sampling to find potentially valid propagation paths. In beam tracing, rectangular beams are traced from sound sources through the scene to see if the listener is contained in any beams and therefore hears the sound [Funkhouser et al. 1998; Laine et al. 2009]. Frustum tracing is another method similar to beam tracing that uses rectangular frusta instead of beams to perform sound propagation [Chandak et al. 2008]. While these methods work well, they suffer from performance issues due to the geometric complexity inherent in tracing volumes through an environment. Neither method is fast enough to be practical in an interactive application. Other methods have been proposed such as RESound that uses a hybrid of ray and frustum tracing [Taylor et al. 2009b]. However, these methods still do not meet real-time requirements.

3.1 Ray-Tracing Sound Propagation: iSound

Ray tracing for sound propagation has several advantages over other numeric and geometric simulation techniques: it can easily handle dynamic scenes, requires less preprocessing, and is much more amenable to real-time simulation. Due to the simplicity of ray tracing, it can also be implemented on the GPU, improving simulation times

The most recent prior system, iSound, uses a forward ray-tracing algorithm on the GPU to perform sound propagation at real-time rates [Taylor et al. 2010]. In this approach, a random spherical sampling of rays is cast from each sound source in a scene. These rays are then propagated through the environment via specular reflection and diffraction to an arbitrary user-defined recursion depth. Diffraction is performed whenever a ray touches a triangle which as been previously marked as having a diffracting edge as part of a preprocessing step [Taylor et al. 2009a]. The ray is tested to see if the barycentric coordinates of its intersection point lie closer to the edge than some threshold value. If so, secondary rays are cast throughout the shadow region bounded by the adjacent triangle. These rays are then propagated through the scene as described above.

This initial ray propagation is used to determine sets of geometry visible to sound sources at each recursion depth. As each ray is cast, it is checked for intersection with a sphere of user-defined radius centered at the listener's position. If so, the set of triangles that the ray has interacted with as it was propagated is added to a list of potentially valid propagation paths from the sound source to the listener.

After all visibility rays have been traced from every sound source in the scene, the algorithm validates all potential propagation paths in order to determine a final set of contributions of each sound source detected by the listener. An image-source method is performed to validate each path [Allen and Berkley 1979]. The position of the sound source is recursively reflected over each triangle in the propagation path to produce a series of source image positions. If a path contains diffraction, the image source position at that depth is defined on the diffraction edge by the UTD formation for edge diffraction [Kouyoumjian and Pathak 1974]. Occlusion query rays are then traced from the listener backwards to the most recent source image. If the ray reaches the triangle in question without intersecting any other geometry, an intersection point is calculated and the process is repeated until the source is reached. If there is occlusion, the propagation path in question is marked as invalid and removed from the list of propagation paths.

At this point, all paths left in the list will be valid. Additional data needed for sound rendering is then calculated for each path: the total distance from the source to the listener, the direction from the listener to the first source image, and a frequency-dependent attenuation value caused by interaction with materials in the scene at each depth of recursion. The UTD formulation for edge diffraction is used to determine attenuation coefficients for diffraction interaction along the path.

4 Fast Sound Propagation for Games

Our new algorithm builds upon the work in iSound by modifying the algorithm to avoid issues encountered and to achieve the fastest performance possible for game-like scenes. The primary differences of our approach include backward ray tracing from the listener in the visibility determination step and propagation path caching

for better frame coherence.

4.1 Backward Sound Propagation

In our algorithm, we chose to shoot visibility rays from the listener, rather than from each sound source. This decision was made based upon the following observation: the early reflections and diffractions that are perceptually important tend to come from geometry in the vicinity of the listener. Thus, it is advantageous to cast more rays from the listener's position. When casting rays from each sound source, only a few may reach the listener and these may not be the most perceptually important paths, requiring more rays to be cast in order to get all necessary propagation paths. Our method tries to get as many of the important paths as possible while shooting fewer rays than other techniques. This strategy has the added benefit that the amount of rays no longer scales linearly with the number of sound sources.

The algorithm begins by casting a random spherical sampling of rays from the listener. These rays are propagated through the scene as with iSound. Rays are specularly reflected by triangles they intersect up to a user-defined maximum recursion depth. The algorithm maintains a hash table of visited propagation paths for each depth of reflection. All paths with one reflection are kept in one table, all paths with two reflections are kept in another, and so on. At each depth of reflection, the hash table for the current depth is queried to see if it contains the ordered sequence of triangles previously visited by the current visibility ray. If so, the algorithm continues propagating that ray to the next depth of reflection. If the triangle sequence has not been visited, it is added to the hash table and any valid propagation paths are found. The hash table keeps the algorithm from producing duplicate propagation paths and is crucial for frame-to-frame path caching. It is analogous to the visibility sets kept by iSound.

When a new triangle sequence is visited, the listener's position is reflected over each triangle in the sequence, creating a series of listener image positions (versus iSound's source image positions). Each sound source in the scene is then tested to see if there is a valid reflection path back to the listener using a variation of the image source validation method from iSound. If any edge of the most recent triangle has been previously marked as a diffraction edge, diffraction paths over that edge from source positions to the listener image position are found. The algorithm considers only sound sources that lie in the diffraction shadow region from the listener's perspective. This approach is valid because the geometry of UTD diffraction is symmetrical [Kouyoumjian and Pathak 1974]. Like iSound, we use the UTD diffraction formulation to determine the point on the edge at which diffraction occurs and then perform path validation back to the listener as with reflection paths.

For each valid propagation path, the system calculates the same output as iSound: the total distance along the path, the direction of the path from the listener, and the total frequency-dependent material attenuation along the path. In addition to these values, our algorithm also calculates the relative speed along the propagation path of the source and listener. This allows the sound renderer to perform more accurate delay interpolation by using physically correct doppler shifting to guide the interpolation.

4.2 Propagation Path Caching

Given the random nature of the rays used for visibility determination, visible triangle sequences are often inconsistent from frame to frame. This results in propagation paths that drop in and out, even when neither source nor listener are moving. This problem is solved in our approach by using the visibility hash tables as persistent caches. At the beginning of each frame, all triangle sequences in the hash tables are checked to see if propagation paths exist from the current source positions to the listener. If a previously valid path is invalidated in this step, these triangle sequences are removed from the hash table. Otherwise, the triangle sequences remain in the hash table.

This approach has the effect that once paths are found, they are kept and updated until they become invalid. This avoids the frame coherency issues that plague other ray-sampling algorithms. Perhaps the greatest benefit of the caching is that far fewer visibility rays need to be cast each frame. The visibility sets are iteratively refined over many frames, resulting in a higher overall frame rate and lower latency for real time applications. For example, casting 1000 rays per frame over 10 frames has a similar time cost and output to casting 10000 rays in a single frame. The former is far more preferable for applications like games where lower latency is critical for the user.

5 Analysis

In order to evaluate our system we used several benchmarks that tested a variety of situations relevant to games and real time virtual simulation. The first benchmark is an indoor scene from the Gamebryo game engine with 1556 triangles. This scene is representative of simple indoor environments common in some games. The second scene is an outdoor desert environment from the Gamebryo game engine with 11642 triangles. This scene is similarly representative of common outdoor game environments consisting of a heightfield terrain with scattered buildings. The final scene used was a model of the Sibenik cathedral with 76088 triangles. This scene was chosen because it is of a higher complexity than most game scenes and shows how the system's performance

degrades with high-resolution environments. All benchmarks were performed on a single-core 2.16 GHz consumer laptop.

5.1 Performance

Detailed performance benchmarks were gathered for the indoor scene that show how the time per frame varies based on the number of visibility rays. Figure 1 shows this relationship for three different propagation configurations: 4th-order reflection with 1st-order diffraction, 4th-order reflection, and 2nd-order reflection. We observed the expected linear time relationship between the time per frame and number of visibility rays. There is also a linear correlation with the reflection depth: 4th order reflection is nearly twice as slow as 2nd order reflection. First-order diffraction adds only a small overhead to the system.

We find that the performance of our system is sufficient to meet the demands of real-time applications like games. With 1000 visibility rays, 4 orders of reflection and 1 order of diffraction, our system runs at over 30 frames per second on a single consumer CPU core. We believe that this frame rate is sufficient to provide a low-latency experience in modern fast-paced video games. Our system can also be tuned to run even faster with minimal quality decrease as shown in section 5.3.

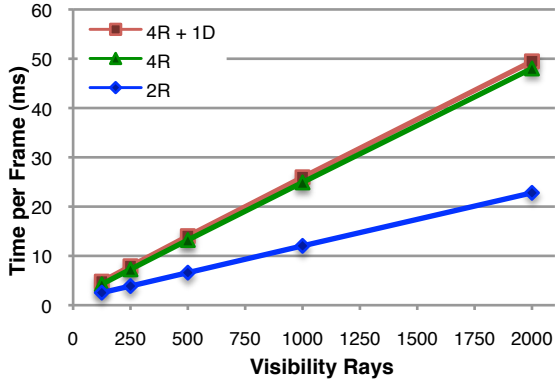

Figure 1: A plot of the time per frame for the indoor scene versus the number of visibility rays for each trial. Diffraction incurs minimal additional cost while the total time is linear with respect to the number of rays and reflection depth.

5.2 Accuracy

The same indoor scene and propagation configurations were used to perform benchmarks that show how the simulation quality changes with the number of visibility rays and depth of reflection. We measure the simulation quality by calculating the average number of propagation paths per frame for the entire benchmark demo. See figure 2 for a graph showing this relationship.

Perhaps the largest advantage of our sound propagation algorithm is that the number of propagation paths that it detects is less sensitive to the number of visibility rays. For all three scenarios in the graph, we observed an almost constant number of paths detected when more than 500 rays were cast. Below 500 rays, the number of paths drops off slowly. In fact, for 4th order reflections, 125 visibility rays were able to detect 84% of the paths detected by 500 rays. Of further interest is how lower-order reflections are even less susceptible to quality degradation with very small numbers of rays. When performing 2nd order reflection, just 125 rays is able to find 97% of the paths found with 500 rays.

The figure also shows how enabling diffraction can result in a significantly higher number of paths. 4th order reflection with diffraction produces 25-30% more paths than 4th order reflection without diffraction.

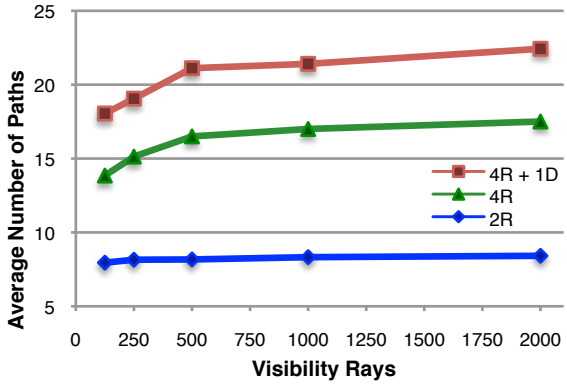

Figure 2: A plot of the average number of propagation paths per frame for the indoor scene versus the number of visibility rays for each trial. Our system maintains accuracy when shooting even a very small number of rays.

5.3 Performance & Accuracy Tuning For Games

Most modern video games have very stringent CPU budgets. This is even worse on game consoles where resources are limited. Some games allocate as little as 5 to 10% of the CPU budget for sound. Thus, it is important for any practical sound propagation system to be highly performant. We believe our system to be highly tunable such that it can fit within these CPU budgets while still providing a significant increase in aural realism over current non-physically-based methods.

The primary ways that our system allows for performance tuning is by changing the number of visibility rays, the depth of reflection and whether or not diffraction is enabled. Given that the time complexity of our system is linear with respect to both the number of visibility rays and the reflection depth, and that the accuracy of the simulation is not very sensitive to the

number of rays except at higher depths of reflection, the system can be easily optimized for any given environment. For the indoor scene discussed in this section, the optimum number of visibility rays is around 500. Above this point there is minimal benefit from shooting more rays and below this point the simulation's accuracy decreases more noticeably. This performance point is scene dependent: the nature of the algorithm causes more rays to be necessary when the scene complexity is higher. Table 1 shows the performance and number of propagation paths for the other benchmark scenes.

Scene	# of Triangles	Time Per Frame (ms)	Average # of Paths
Indoor	1566	26	21.4
Outdoor	11642	27	4.4
Cathedral	76088	108	6.6

Table 1: Benchmarks for various scenes. Data was gathered with 1000 visibility rays, 4 orders of reflection and 1 order of diffraction.

6 Comparisons

To our knowledge, the fastest geometric sound propagation system up to this point is iSound. Since both systems were tested using 3 of the same benchmarks, the results can be directly compared. For the indoor scene, our system achieves similar performance to iSound when using 500 visibility rays. Our system finds fewer propagation paths for the outdoor and cathedral scenes but achieves similar performance for these scenes with 1000 visibility rays [Taylor et al. 2010]. This is due primarily to the increasing complexity of these scenes that necessitates more visibility rays. While iSound performs well in these scenes, it is a GPU-based sound propagation system and this makes it impractical for real-time graphics-heavy applications like video games. In these situations, a fast CPU-based sound propagation system is necessary.

When compared to other CPU-based geometric propagation algorithms such as RESound, ADFrustum, and the CPU version of iSound, our algorithm is by far the fastest. For the Sibenik cathedral scene, our system is at least 30 times faster than RESound for 3 orders of reflection and 1 order of diffraction [Chandak et al. 2008]. ADFrustum performs similarly to RESound for the same cathedral scene [Taylor et al. 2009b]. In addition, both of the benchmarks for these systems were performed using 7 threads on a multicore CPU, while our system was tested using a single core. While CPU timings for iSound are not publicly available, we estimate our system to be at least 10 times faster for similar scenes because iSound must shoot many more rays per frame to find the same propagation paths.

We are able to achieve these performance numbers primarily because our system uses path caching to reduce the number of rays needed for each frame. Since ray casting is the slowest part of ray tracing propagation algorithms, it is advantageous to design an algorithm that requires less rays to meet a give standard of accuracy. We believe our algorithm meets this goal and is perhaps the first geometric sound propagation algorithm that is practical enough to be used in games.

7 Conclusions & Limitations

We have presented a novel geometric sound propagation algorithm that maintains the accuracy of prior methods but is also able to meet the stringent time requirements of games and other interactive systems. We have integrated our sound propagation and rendering system into the Gamebryo game engine in order to demonstrate that it is very acceptable for these applications. Our system executes on a single CPU core and has a small memory footprint, making it practical for resource-intensive games. We are able to achieve these results using a ray-tracing algorithm due to our use of propagation path caching and backwards ray tracing. These contributions allow our system to maintain accuracy with very small numbers of visibility rays versus other existing algorithms.

While our algorithm performs well, there are still many improvements that could be made. In the future, we would like to focus on both algorithmic improvements and optimization. Our algorithm doesn't support diffuse reflection or higher order diffraction. Solving these problems would produce even more accurate results. However, initial investigations show that higher order diffraction seems to be very difficult to do at interactive rates. In addition, the ray tracer that is used by our system is simple and doesn't make use of the SIMD instructions present on modern processors. Implementing this improvement could provide a sizable speed increase. Finally, we would like to do more work with integrating our sound propagation system into game engines in order to evaluate the algorithm's performance on real-world environments.

8 Acknowledgments

This work was supported in part by Army Research Office, RDECOM and National Science Foundation.

References

TAYLOR, M., CHANDAK, A., MO, Q., LAUTERBACH, C., SCHISSLER, C., MANOCHA, D., 2010. iSound: Interactive GPU-base Sound Auralization in Dynamic Scenes, Tech Report TR10-006, Department of Computer Science, UNC Chapel Hill

ALLEN, J. B., AND BERKLEY, D. A. 1979. Image method for efficiently simulating small-room acoustics. The Journal of the Acoustical Society of America 65, 4 (April), 943–950.

CHANDAK, A., LAUTERBACH, C., TAYLOR, M., REN, Z., AND MANOCHA, D. 2008. AD-Frustum: Adaptive Frustum Tracing for Interactive Sound Propagation. IEEE Transactions on Visualization and Computer Graphics 14, 6 (Nov.-Dec.), 1707–1722.

DALENBA¨CK, B.-I. L. 1996. Room acoustic prediction based on a unified treatment of diffuse and specular reflection. The Journal of the Acoustical Society of America 100, 2, 899–909.

FUNKHOUSER, T., CARLBOM, I., ELKO, G., PINGALI, G., SONDHI, M., AND WEST, J. 1998. A beam tracing approach to acoustic modeling for interactive virtual environments. In Proc. of ACM SIGGRAPH, 21–32.

FUNKHOUSER, T., TSINGOS, N., CARLBOM, I., ELKO, G., SONDHI, M., WEST, J., PINGALI, G., MIN, P., AND NGAN, A. 2004. A beam tracing method for interactive architectural acoustics. Journal of the Acoustical Society of America 115, 2 (February), 739–756.

KOUYOUMJIAN, R. G., AND PATHAK, P. H. 1974. A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface. Proc. of IEEE 62 (Nov.), 1448–1461.

KROKSTAD, A., STROM, S., AND SORSDAL, S. 1968. Calculating the acoustical room response by the use of a ray tracing technique. Journal of Sound and Vibration 8, 1 (July), 118–125.

LAINE, S., SILTANEN, S., LOKKI, T., AND SAVIOJA, L. 2009. Accelerated beam tracing algorithm. Applied Acoustic 70, 1, 172–181.

LENTZ, T., SCHRODER, D., VORLA¨NDER, M., AND ASSENMACHER, I. 2007. Virtual reality system with integrated sound field simulation and reproduction. EURASIP Journal on Advances in Signal Processing. Article ID 70540, 19 pages.

PIERCE, A. D. 1974. Diffraction of sound around corners and over wide barriers. The Journal of the Acoustical Society of America 55, 5, 941–955.

SCHRO¨DER, D., AND LENTZ, T. 2006. Real-Time Processing of Image Sources Using Binary Space Partitioning. Journal of the Audio Engineering Society (JAES) 54, 7/8 (July), 604–619.

TAYLOR, M., CHANDAK, A., ANTANI, L., AND MANOCHA, D. 2009. Resound: interactive sound rendering for dynamic virtual environments. In MM '09: Proceedings of the seventeen ACM international conference on Multimedia, ACM, New York, NY, USA, 271–280.

TSINGOS, N., FUNKHOUSER, T., NGAN, A., AND CARLBOM, I. 2001. Modeling acoustics in virtual environments using the uniform theory of diffraction. In SIGGRAPH 2001, Computer Graphics Proceedings, 545–552.

TSINGOS, N. 2009. Pre-computing geometry-based reverberation effects for games. 35th AES Conference on Audio for Games.

VORLA¨NDER, M. 1989. Simulation of the transient and steady-state sound propagation in rooms using a new combined ray-tracing/image-source algorithm. The Journal of the Acoustical Society of America 86, 1, 172–178.

BOTTELDOOREN, D. 1995. Finite-difference time-domain simulation of low-frequency room acoustic problems. Acoustical Society of America Journal 98 (December), 3302–3308.

RAGHUVANSHI, N., NARAIN, R., AND LIN, M. C. 2009. Efficient and accurate sound propagation using adaptive rectangular decomposition. IEEE Transactions on Visualization and Computer Graphics 15, 5, 789–801.

ANTANI, L., CHANDAK, A., TAYLOR M. AND MANOCHA, D. 2010. Direct-to-Indirect Acoustic Radiance Transfer, Tech Report TR10-002, Department of Computer Science, UNC Chapel Hill