REAL-TIME SIMULATION OF SOUND SOURCE OCCLUSION

Chris Share
Sonic Arts Research Centre
Belfast, U.K.

Graham McAllister
Sonic Arts Research Centre
Belfast, U.K.

Keywords: Audio, multimedia, occlusion, virtual auditory environments.

Abstract: Sound source occlusion occurs when the direct path from a sound source to a listener is blocked by an intervening object. Currently, a variety of methods exist for modeling sound source occlusion. These include finite element and boundary element methods, as well as methods based on time-domain models of edge diffraction. At present, the high computational requirements of these methods precludes their use in real-time environments. In the case of real-time geometric room acoustic methods (e.g. the image method, ray tracing), the model of sound propagation employed makes it difficult to incorporate wave-related effects such as occlusion. As a result, these methods generally do not incorporate sound source occlusion. The lack of a suitable sound source occlusion method means that developers of real-time virtual environments (such as computer games) have generally either ignored this phenomenon or used rudimentary and perceptually implausible approximations. A potential solution to this problem is the use of shadow algorithms from computer graphics. These algorithms can provide a way to efficiently simulate sound source occlusion in real-time and in a physically plausible manner. Two simulation prototypes are presented, one for fixed-position sound sources and another for moving sound sources.

1 INTRODUCTION

1.1 Sound Source Occlusion

Sound source occlusion occurs when the direct path from a sound source to a listener is blocked by an intervening object. An example of an occluded sound source is shown in Fig. 1. In situations such as that of Fig. 1., sound emanating from the source often still reaches the listener, however its characteristics are altered due to the size and orientation of the objects it encounters during its propagation.

This is a common occurrence. In everyday life the effects of sound source occlusion surround us, shaping our perception of our physical environment and providing important information about the size and orientation of objects within this environment. If we are to create immersive and believable acoustic simulations of real environments it is important to include this physical phenomenon.

Despite the existence of several different methods for simulating sound source occlusion, in general this phenomenon has not been included in computer simulations of acoustic environments. This is especially true in the case of real-time simulations such as computer games. In this case, developers have generally either ignored the phenomenon of sound source occlusion or used rudimentary and perceptually implausible approximations. This is detrimental to the simulation and detracts from its realism. Thus, as Martens (Martens, 2000) points out, the simulation of sound source occlusion remains one of the unsolved problems in the rendering of real-time virtual acoustic environments.

In this paper we present a novel approach to simulating sound source occlusion in real-time virtual environments. This approach is based on the use of graphical shadow algorithms. It is shown that this approach has several advantages over existing methods and is capable of producing plausible results in a timely and efficient manner. Two simulation prototypes are presented, one for fixed-position sound sources and another for moving sound sources.

In Section 2 of this paper we review current approaches to simulating sound source occlusion in virtual environments. In Section 3 of this paper we present a novel approach to the simulation of sound

Share C. and McAllister G. (2006).
REAL-TIME SIMULATION OF SOUND SOURCE OCCLUSION.
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications*, pages 193-199
DOI: 10.5220/0001571501930199

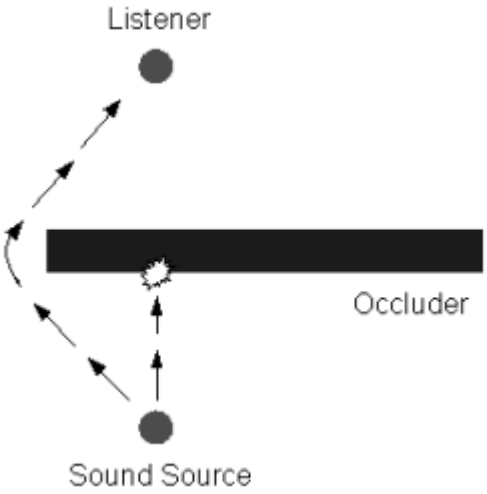

Figure 1: Occluded sound source.

source occlusion based on the use of shadow algorithms from computer graphics. Two prototype implementations using this approach are described. Finally, we discuss several ways in which the existing two prototypes can be extended.

1.2 Real-time Rendering

This paper focuses on real-time applications so it is important to be clear about the meaning of real-time in this context. Akenine-Moeller *et al.* (Akenine-Moeller and Haines, 2002) describe real-time graphics rendering as a process of interaction between a user and a screen image in which the user reacts to the image, thereby producing appropriate changes in the image. It is generally accepted that a real-time application should be responsive to user input and that the response time should be relatively short, preferably in the millisecond range. For graphics applications Akenine-Moeller *et al.* recommend a framerate of between 15 and 72 frames per second in order for the images to be perceived as continuous. This gives a minimum rendering time of approximately 66 milliseconds for each frame. It is important to note however, that in real-time graphics libraries such as OpenGL, the framerate is determined by the complexity of the scene that the application renders. The more computationally intensive the scene, the slower the framerate will become. This is because the application must wait for the current call to the render loop to complete before the next call to the render loop begins. Although framerates below 15 frames per second produce stilted, choppy animations and are considered less than optimal, they may still be acceptable to users for short time periods.

The case of real-time audio rendering is somewhat different to that of real-time graphics rendering. Audio applications typically use a buffered, interrupt-driven approach to audio generation. In this case, a time-based operating system callback is periodically called to fill an audio buffer. This callback runs regardless of whether the audio rendering has been completed or not. This means that developers must ensure that the time taken to generate each audio buffer is equal to, or less than, the time taken to play the buffer (see (Bencina, 2003) for further discussion of this issue). Otherwise, the output audio stream will be compromised. The time period of the audio callback is normally user-defined and can be as little as 16 samples (approximately 0.3628 ms at a sample rate of 44100 Hz) in low-latency applications. For this reason, the audio component of virtual environments must be as efficient as possible. The constraints of real-time operation mean that computational complexity must be limited. Note also that in the case of environments such as computer games, a CPU allowance is usually allocated to the audio component of the application. This is usually in the vicinity of 5% to 10% of the application's total CPU usage. This further limits the allowable computational complexity of the audio algorithms that are used.

1.3 Simulation vs. Modeling

The goal of the approach presented in this paper is to produce a plausible output rather than one which is necessarily physically accurate. This approach mirrors that commonly taken in real-time computer graphics simulations where the goal is to produce a perceptually plausible output.

2 BACKGROUND

In comparison with other areas of acoustics research, the effects of an occluding object on the wavefront produced by a sound source has received relatively little attention. In one of the few systematic studies into this subject, Farag (Farag, 2004) has shown that the general effect consists of two parts: a lowpass filtering effect due to the diffraction of the wavefront around the occluding edge, and a comb filtering effect due to the combination of the various different paths that the wavefront takes on its way to the user. Experiments by Martens *et al.* (Martens et al., 1999) confirm the presence of these two effects.

These changes in the wavefront produce significant acoustic effects that can be perceived by the listener. For example, Farag (Farag, 2004) has shown that when a sound source is occluded by an object,

the listener perceives a phantom source located at the edge of the object from which the signal arrives first. Another finding in this study was that sound source occlusion produces a perception of increased distance to the sound source. Studies by Russell (1997) (Russell, 1997) suggest that listeners may also be able to perceive information such as passability or the size of openings from the acoustic signal. That sound source occlusion produces significant physical effects, and that these effects are perceived by listeners, highlights the need for the inclusion of this phenomenon in virtual environments. By so doing, the believability and immersiveness of the acoustic simulation will be enhanced.

At a basic level, the problem of simulating the effects of sound source occlusion involves solving the wave equation at the position of interest (usually the listener position) in terms of the wave equation at surrounding positions. The two most commonly used approaches to this problem are the finite and boundary element methods (see for example, Kopuz and Lalor (Kopuz and Lalor, 1995)). These approaches solve the wave equation by expressing it as a set of linear equations derived from the discretization of the environment in space and time. To do this accurately is a difficult computational challenge and is generally too computationally intensive for use in real-time applications.

A related approach using waveguide meshes has been proposed by Murphy and Beeson (Murphy and Beeson, 2003) and Savioja and Lokki (Savioja and Lokki, 2001). This approach extends the one-dimensional digital waveguide technique pioneered by Smith (Smith III, 1987) to meshes of two and three dimensions. In terms of occlusion, an advantage of this technique is that the wave propagation effects are an inherent part of the implementation. However, there are several difficulties with this approach. Firstly, despite the simplifications, the method is still computationally intensive. Also, there are unsolved problems concerning dispersion and boundary conditions that impact upon the accuracy of the simulation.

An alternative approach is the use of time-domain models of edge diffraction (see for example, Lokki *et al.* (Lokki et al., 2002)). In this approach, the impulse response of the occluding edge is directly calculated. The difficulty with this approach is that the calculated impulse response is often quite long and therefore, too computationally intensive for real-time use. The authors suggest that the calculated response can be simulated by using a warped IIR filter with a matching frequency response. Unfortunately, difficulties involved in this matching process precludes the use of this approach in real-time environments.

A variety of other approaches based on geometric methods have been used to generate room acoustic models (e.g. (Allen and Berkley, 1979), (Krockstadt et al., 1968)) however these rarely incorporate diffraction or occlusion effects. In these cases, the model employed assumes that sound propagates in the same manner as light, that is, that the wavelength of the propagating wave is much shorter than the size of the objects in the environment. This means that low frequency effects such as diffraction and occlusion are not included.

Simpler approaches to simulating sound source occlusion include the use of Fresnel zones (Tsingos and Gascuel, 1997) which generate a frequency-dependent visibility factor that can then be used to set the parameters of a filter that is then applied to the source signal. More simple again is the approach taken by Takala and Hahn (Takala and Hahn, 1992) who chose to reduce the effects of sound source occlusion to that of changes in source gain using a value proportional to the degree of occlusion. A similar approach is generally taken in computer games where sound source occlusion is either completely ignored or simulated using a simplified model that attenuates the source signal, and in some cases, also filters it using a lowpass filter. A typical example of this latter approach is that of Creative's EAX software (Environmental Audio Extensions 5.0, 2006).

From the above discussion it can be seen that at present there is no method capable of producing plausible sound source occlusion effects in a real-time environment such as a computer game. Lokki *et al.* (Lokki et al., 2002) suggest that in order to simulate this phenomenon, a mapping directly from a geometry (positions of source, receiver and edge) to diffraction filter coefficients is needed. A potential solution to this problem is the use of shadow algorithms from computer graphics. This approach can be used to efficiently and plausibly simulate sound source occlusion. An implementation of this approach is described in the following section.

3 IMPLEMENTATION

3.1 Soundshadows

One possible way of mapping virtual scene geometry to filter coefficients is through the use of real-time graphical shadow algorithms. The effect of this approach is to produce what we have termed soundshadows. These soundshadows can be used to store the parameters of a filter system that simulates the aural effect that a listener perceives when an object intervenes between their position in the environment and that of the sound source.

The generation of shadows has been a focus of computer graphics research since the 1970s. Since then, a wide variety of algorithms for generating

graphical shadows have been developed. In general, these methods can be grouped into five distinct categories:

1. Lightmaps (Fake shadows)
2. Projected Planar Shadows (Blinn, 1988)
3. Shadow Volumes (Crow, 1977)
4. Shadow Maps (Williams, 1978)
5. Soft Shadows (e.g. (Assarsson, 2003))

In this paper we describe implementations of soundshadows based on the first two methods listed above: lightmaps and projected planar shadows. These algorithms were implemented in C using the OpenGL graphics library (OpenGL, 2006) and GLUT windowing toolkit (Robbins and Kilgard, 2006). A visual scene comprising a sound source, an occluding object and a listener was developed. The user observes the scene from a first-person perspective and can move about within the scene using keyboard commands. In order to simulate changes in scene geometry, and hence changes in occlusion effects, a door which can be opened or closed by the user is included in the scene.

As this project has a focus on real-time applications, and in particular, computer games, it was decided to use the OpenAL audio library (OpenAL, 2006) for audio output. OpenAL is a free, open source, cross-platform 3D audio library. The library models source-listener distance and azimuthal source position. The library has both hardware rendering and software rendering options available. In the case of the latter, source distance and position effects are simulated using changes in output signal amplitude. In the case of the prototype software, the listener position was simulated using a mono output signal, and the effects of the occluding object were simulated in two dimensions only. OpenAL does not incorporate any audio filtering capability so in order to simulate the effects of an occluding object it was necessary to add this to the library. Hence, a second-order IIR filter was added to the library. The update rate of the occlusion calculation is determined using a combination of the OpenGL frame-rate and an operating system-based timer. It was found that an update period of 0.05 seconds provided suitable results for changes in the occlusion value.

The general algorithm used for implementing graphical shadows as soundshadows is shown in Figure (2).

The algorithm is implemented in the following manner. First, a test is performed to determine whether the listener is in an area designated as the occluded sound field. If the result of this test is negative, then no further processing of the soundshadow algorithm takes place. If the result of the test is true, then the algorithm returns an occlusion value between 0.0 and 1.0. This value is then used to set the cutoff frequency of a lowpass filter, thus simulating the effect of the occluding object. A key component of the algorithm is the manner in which the occlusion value is calculated. This is described further in the following two sections.

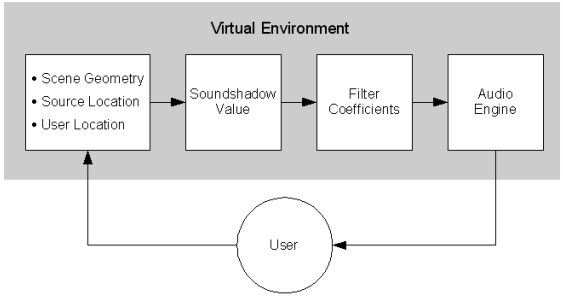

Figure 2: General soundshadow algorithm.

3.2 Lightmaps and Soundmaps

Lightmaps (also know as shadow maps) are a common part of virtual environments, especially computer games. They provide a fast, efficient way to simulate lighting effects for fixed-position light sources. Lightmaps are essentially graphical textures, usually greyscale bitmaps, that are layered over an existing texture and surround a light source. The lightmap is blended with the underlying texture so that a light-to-dark fade effect, corresponding to distance from the light source, is produced. This simulates the effect of the light source on the surrounding object. It is important to note that lightmaps suffer from several inherent limitations, in particular, that they cannot interact with other objects in the environment. For example, an object moving past a lightmap will not receive any light from it. Another limitation is that lightmaps can only be applied to fixed position light sources, as it is too computationally expensive to recalculate the lightmap as the light source moves.

A first soundshadow prototype has been developed using the lightmap algorithm as a model. As noted above, this algorithm can only be applied to stationary sound sources (although it does support changing environment geometry). A screen capture from this prototype is shown in Figure 3. Note that in the screencapture the soundmap is visible but that in general use, it will not be seen by the user.

In the case of the soundmap approach, the occlusion value is calculated by finding the value stored in the soundmap at the current user position. The manner in which the soundmap is initially created is flexible and can be as simple or as complex as the developer wishes. For the prototype described above the

Figure 3: Using a soundmap to simulate sound source occlusion.

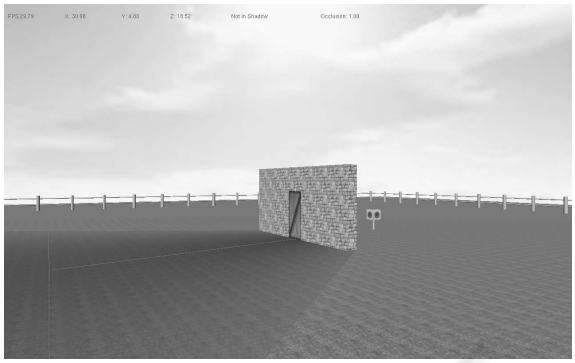

Figure 4: Using a projected planar shadows to simulate sound source occlusion.

soundmap was created by drawing it using a graphics program. However, because the creation of the soundmap is an offline process, more accurate approaches could certainly be used and are currently under development.

3.3 Projected Planar Shadows

Blinn (Blinn, 1988) first introduced the concept of projected planar shadows. In this approach a graphical shadow is generated by projecting the light source on to a two-dimensional plane. The algorithm is simple and fast however it does not produce intensity values within the shadow field. The algorithm for generating a projected planar shadow is shown in (1).

$$P = V + \frac{d - N.V}{N.V - N.L} + (V - L) \qquad (1)$$

where P is a point on the plane, V is the vertex position of the object casting the shadow, d is the dot product of P and N, N is a normal to the plane and L is the light position. Unlike the lightmap algorithm described above, the projected planar shadow algorithm can handle moving light sources.

A second soundshadow prototype based on the projected planar shadow algorithm has also been implemented. This prototype supports moving sound sources. A screencapture from this prototype is shown in Figure 4. Again note that although the soundshadow is displayed, it will not usually be seen by the user.

In the case of the projected planar soundshadow approach, the occlusion value O is calculated using the two distance measurements d and c shown in Figure 5. The first measurement used is the distance from the listener position to the occluding object, while the second measurement used is the distance from the listener position to the centreline of the occluded soundfield. These two values are combined to produce an overall occlusion value according to (2).

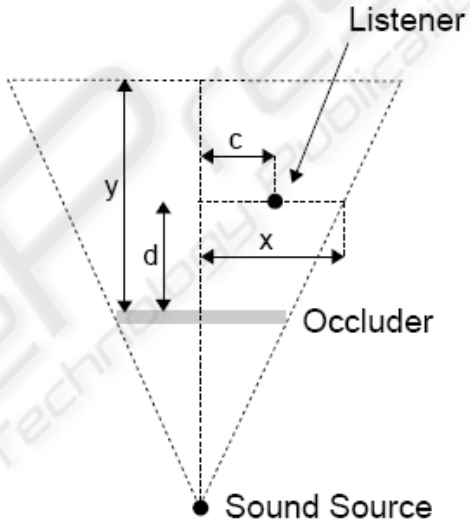

Figure 5: Derivation of the occlusion value in the case of projected planar shadows.

$$O = (1 - \frac{d}{y}) \times (1 - \frac{c}{x}) \qquad (2)$$

Note that by using this method the value at the edge of the occluded soundfield will always be equal to 0.0 which is interpreted as meaning that no occlusion takes place.

4 DISCUSSION

The initial implementations of the soundmap and projected planar shadow prototypes incorporated only a lowpass filter effect. While their simulation of sound source occlusion was reasonably effective, it immediately became apparent that the phantom sound source

arising at the nearest occluding edge needed to be included in the simulation to make it plausible. In cases where the prototype simulated multiple paths from the sound source to the listener around several occluding edges, it was found that the comb filter effect was a crucial part of the simulation. Its absence reduced the plausibility to a large degree. This was particularly true in cases where sound reached the listener via paths of approximately similar length.

The comb filter effect was found to be particularly important in the case of the moving sound source prototype. In this case it was found that the introduction of a type of amplitude panning between the various occluding edges was necessary in order to create a convincing effect as the sound source moved within the environment. By incorporating a pathlength-dependent delay into the sound source's signal it was found that the comb filter effect produced by the multiple phantom sources could be approximated.

Implementing sound source occlusion using graphical shadow algorithms has several key advantages over existing real-time methods. Of primary importance is that the algorithms are designed to run in real-time. A secondary advantage is that the algorithms used may already be part of the virtual environment software. Most computer games now include some form of real-time shadow simulation. This means that implementing soundshadows requires less additional coding effort on the part of the environments developers. Another potential advantage is that as these algorithms become more widely used it is likely that they will be implemented in graphics hardware, thereby allowing further acceleration.

Simulating sound source occlusion as a soundshadow can be justified on the grounds that sound, like light, is a wave phenomena and so can be modeled in a similar manner. However, it is important to note that there are significant differences between sound and light which must be taken into consideration and which limit the accuracy of the simulation. These differences, described by Funkhouser *et al.* (Funkhouser et al., 2003) include wavelength, speed, coherence, dynamic range and latency. These differences impose natural limits on the accuracy of the simulation.

In order to test the accuracy of the soundshadow algorithms an application is currently being developed which compares the simulated frequency response at particular listener positions with the predicted analytical frequency response. Farag (Farag, 2004) showed that Svenssons edge diffraction model (Svensson et al., 1999) agrees well with experimental evidence. As a result, it was decided that Svensons model would be used as the standard by which the accuracy of the various sound shadow algorithms output would be evaluated. Due to the computational complexity of the Svensson model this application cannot run in real-time.

5 CONCLUSIONS AND FUTURE WORK

From the preliminary results produced by the soundshadow approach to simulating sound source occlusion it can be concluded that graphical shadow algorithms can be used to simulate the effects of this physical phenomenon in a plausible manner in real-time. The approach produces better results than those currently achieved in commercial software and has a negligible impact on the graphical framerate.

Further research remains to be done in order to increase the quality of the simulations. In the case of the soundmap prototype, the first step is to develop a method of accurately generating the soundmap. This could be performed using an offline process such as that outlined in Svensson (Svensson et al., 1999). In the case of the projected planar shadow prototype it is possible that a soundmap could also be used, however the constantly changing positions of the source/occluder/listener may produce occlusion effects that are too complex to simulate accurately with this approach. Instead, it may be more useful to investigate more sophisticated filter algorithms that better approximate the predicted frequency response, and then use the approach described above to control the filter parameters.

The virtual environment created in the two prototype applications is very simple, comprising only a single sound source, occluding object and listener. In practice, most virtual environments are far more complex than this, consisting of possibly hundreds or even thousands of sound sources and objects. Hence, an important question is the degree to which the soundshadow method can be scaled to larger virtual environments. Closely related to this question is the issue of performance. In the case of more complex environments it will be necessary to evaluate the efficiency of the soundshadow algorithms using benchmarking and asymptotic analysis so that the efficiency of the algorithms is ensured.

Another important question is the degree of accuracy necessary in the occlusion algorithm in order to produce an acceptable occlusion effect. In general, occlusion has not been considered an important effect to simulate in real-time environments and so has been coarsely simulated. However, as games and virtual environments become more realistic, it is likely that more accurate acoustic simulations will be required. For this reason it is necessary to determine the level of simulation error that is acceptable to the user in the occlusion model. Perceptual testing using the prototypes described above may be able to provide an an-

swer to this question. Initial feedback from informal listening tests that were conducted as the prototype software was being developed suggest that the accuracy of the soundshadow approach provides a satisfactory level of plausibility for most users.

The development of a simple, efficient yet plausible method of simulating sound source occlusion in virtual environments will contribute to making these environments more immersive and believable. It is hoped that this research can be used to produce virtual environments in which audio plays a more meaningful role in the user experience.

ACKNOWLEDGEMENTS

Special thanks to Dr. Graham McAllister and Prof. Michael Alcorn for their supervision and guidance. Thanks to Mr Chris Corrigan for technical assistance. Funding for this project was provided by a SPUR scholarship.

REFERENCES

Akenine-Moeller, T. and Haines, E. (2002). *Real-time Rendering*. A K Peters.

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

Assarsson, U. (2003). *A Real-Time Soft Shadow Volume Algorithm*. PhD thesis, Department of Computer Engineering, Chalmers University of Technology.

Bencina, R. (2003). Portaudio and media synchronisation - it's all in the timing. In *Proceedings of the Australasian Computer Music Conference*, pages 13–20., Perth. Australasian Computer Music Association.

Blinn, J. (1988). Me and my (fake) shadow. *IEEE Comput. Graph. Appl.*, 8(1):82–86.

Crow, F. C. (1977). Shadow algorithms for computer graphics. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 242–248, New York, NY, USA. ACM Press.

Environmental Audio Extensions 5.0 (2006). http://developer.creative.com/.

Farag, H. H. A. (2004). *The Psychoacoustics of Sound-Source Occlusion*. PhD thesis, Faculty of Engineering, Alexandria University.

Funkhouser, T., Tsingos, N., and Jot, J.-M. (2003). Survey of methods for modeling sound propagation in interactive virtual environment systems. *Presence and Teleoperation*.

Kopuz, S. and Lalor, N. (1995). Analysis of interior acoustic fields using the finite element and the boundary element methods. *Applied Acoustics*, 45:193–210.

Krockstadt, A., Strøm, S., and Sørsdal, S. (1968). Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound and Vibrations*, 8(18):118–125.

Lokki, T., Svensson, P., and Savioja, L. (2002). An efficient auralization of edge diffraction. In *Proceedings of the AES 21st International Conference*, pages 166–172. Audio Engineering Society.

Martens, W. L. (2000). Efficient auralization of small cluttered spaces: Simulating sonic obstructions at close range. In *Proceedings of the 7th Western Pacific Regional Acoustics Conference*, pages 317–320, Kumamoto, Japan.

Martens, W. L., Herder, J., and Shiba, Y. (1999). A filtering model for efficient rendering of the spatial image of an occluded virtual sound source. In *Proceedings of the Forum Acusticum*, Berlin, Germany.

Murphy, D. T. and Beeson, M. J. (2003). Modelling spatial sound occlusion and diffraction effects using the digital waveguide mesh. In *Proceedings of the AES 24th International Conference*, pages 207–216. Audio Engineering Society.

OpenAL (2006). http://www.openal.org/.

OpenGL (2006). http://www.opengl.org/.

Robbins, N. and Kilgard, M. (2006). The OpenGL Utility Toolkit (GLUT). http://www.xmission.com/ nate/glut.html.

Russell, M. K. (1997). *Studies in Perception and Action IV*, chapter Acoustic Perception of Aperture Passability, pages 88–91. Lawrence Erlbaum Associates.

Savioja, L. and Lokki, T. (2001). Digital waveguide mesh for room acoustic modeling. In *Acoustic Rendering for Virtual Environments: ACM SIGGRAPH and Eurographics Campfire*, Snowbird, Utah, U.S.A.

Smith III, J. O. (1987). Musical applications of digital waveguides. CCRMA Report No. STAN-M-39, Stanford University.

Svensson, U. P., Fred, R. I., and Vanderkooy, J. (1999). An analytic secondary source model of edge diffraction impulse responses. *Journal of the Acoustical Society of America*, 106:2331–2344.

Takala, T. and Hahn, J. (1992). Sound rendering, proceedings,. In *Computer Graphics (Proceedings of SIGGRAPH '92)*, pages 211–220. ACM Press.

Tsingos, N. and Gascuel, J.-D. (1997). Soundtracks for computer animation : Sound rendering in dynamic environments with occlusions. In *Graphics Interface'97, Kelowna B.C., Canada*, pages 9–16.

Williams, L. (1978). Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274, New York, NY, USA. ACM Press.