The Use of 3-D Audio in a Synthetic Environment:
Building an "Aural Renderer" for a Distributed Virtual Reality System

Stephen Travis Pope and Lennart E. Fahlén
Distributed Systems Laboratory
Swedish Institute for Computer Science (SICS)
Isafjordsgatan 26
Kista S-16428 Sweden
Electronic mail: stp@sics.se, lef@sics.se

Abstract

We have investigated the addition of spatially-localized sound to an existing graphics-oriented simulation-based synthetic environment—the SICS Distributed Interactive Virtual Environment. To build "3-D audio" systems that are robust, listener-independent, real-time, multi-source and able to give stable sound localization is beyond the current state-of-the-art—even using expensive special-purpose hardware. The "auralizer" or "aural renderer" described here is intended as a test-bed for experimenting with the known techniques for generating perceptual cues for sound localization based on the geometrical models available in a synthetic 3-D world. This paper introduces the psychoacoustical background of sound localization in a general way, noting some of the difficulties in synthesizing sound localization cues, and then describes the software design and usage of the DIVE auralizer. We close by evaluating the system's implementation and performance in the context of the current DIVE project.

Introduction

Sound plays a very important role in the way humans communicate with each other and how they perceive their environment. The SICS Distributed Interactive Virtual Environment (DIVE) is a virtual reality framework for building applications and user interfaces based on the natural metaphor of presence and interaction in 3-D-space. It uses graphical rendering—visualizers—to display objects in synthetic worlds. As an extension of this, we developed a package for the generation of multiple audio signals that have spatial localization cues derived from the geometry of virtual worlds—the so-called auralizer. The DIVE is a multi-user virtual reality system that is implemented within a coarse-grained heterogeneous multiprocessing environment (i.e., a network of different UNIX workstations). The system is a tool kit for building distributed interactive applications based on the simulation of virtual worlds. Independent DIVE applications (called "AIs" after W. Gibson), can run on various nodes distributed within a local- or wide-area network, and update a shared (node replicated) object database. This architecture makes possible the distribution of the computational load of object animation and interaction. User input and output processes are loosely coupled with the rest of the system, allowing for user input device handling, and graphical rendering.

The auralizer tool models the sounds of objects and events that take place in the simulated world using a “source/filter/listener” model of sound generation and processing. The module can also be used as a stand-alone system for 3-D rendering and building of complex interactive sound environments. The package allows DIVE AIs to use sound as another output medium, and allows researchers to experiment with the use of 3-D sound environments, allowing, for instance, new kinds of user feedback using different “aural perspectives” on objects as a way to heightening the degree of illusion and usability.

There are several researchers in VR technology who are developing sophisticated models and hardware/software systems for listener-specific or generic real-time 3-D sound spatialization (e.g., [Wenzel 1992] or [Begault 1992]). Also, a number of expensive hardware systems aimed at high end music studio and film post-production work has been announced recently by major manufacturers. Neither of these domains is the main purpose of our work in the DIVE auralizer, rather we are interested in achieving a “pretty good” 3-D spatialization over headphones or stereo loudspeakers using known techniques and widely available hardware, also adding the requirement that the system run in real-time with multiple (on the order of ten or more) active sound sources. We want to provide a framework in which one can experiment with simple or complex models of the perceptual cues involved in sound source localization, and test these in virtual worlds in combination with visual cues and 3D-interaction devices such as gloves etc.

This document discusses the design issues related to the DIVE auralizer’s architecture. We first outline the basis for spatialization of sounds in the hearing system and its implications for system design and performance. We focus on the “cheap” psychoacoustical models it uses—derived from the literature of recording engineering and electroacoustic music—for doing physical-model-based sound spatialization in three dimensions. We will then discuss some of the design issues related to the architecture and software design of the DIVE auralizer as a real-time, distributed digital audio signal processing application. The work described in this report was done as part of the MultiG project, a Swedish national effort at developing very-high-speed wide area networks and distributed software tools and applications.

Synthesis of Sound Localization Cues

There exists a significant literature in psychoacoustics that is devoted to understanding how humans determine the characteristics of spaces using aural cues and how we localize sound sources

in a 3-D space. As in several other areas of perception psychology, there are several simple, well-understood mechanisms at play, and other, quite complex and still poorly-understood, ones that are equally important. It is obvious that it is central to our ability to localize sound sources that we have two ears, that they are rather directional in their reception pattern, are separated by a small (and constant) distance, and face in different directions. Deeper reflection leads to the realization that our ears are asymmetrical in both the horizontal and vertical (cutting through the head) planes, that there are other cues (e.g., Doppler shift, or inter-ear delays), and higher level functions in the brain (see e.g., [Blauert 1983] or [Durlach et al. 1992]) that we use to localize sound sources.

A simple model of sound localization would use direction and distance to characterize the relative geometrical orientation of the sound source and the listener, and provide cues for these, such as relative and absolute amplitude, spectrum envelope and inter-ear delays. Simple reverberation as a cue for the characteristics of the space is also widely used, whereby reverberation time can be keyed to the volume of the space, and the direct-to-reverberated signal ratio is mapped onto the source-listener distance. The more complex relationships between the spectrum of the sound and its location and the characteristics of the space is still a topic of significant research. It is interesting to note the extremely high time-domain resolution that the auditory system is capable of, especially for inter aural delays. Some researchers claim on the order of microseconds! (Nordmark 1976)

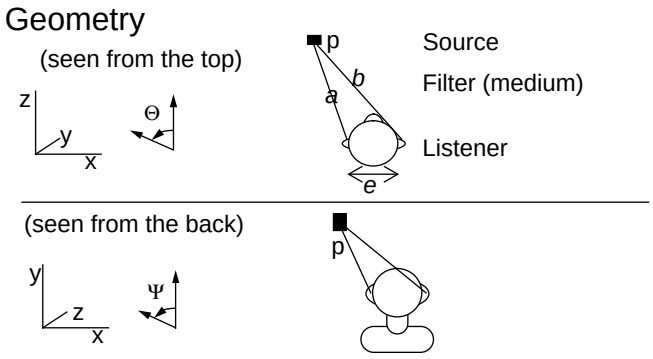
Looking at the basic perceptual features of a sound, we can make a table relating each of them to one or more aspects of a spatial model as shown in Table 1 below.

- Amplitude (loudness) — distance to source
- Position (in 3D space) — direction to source
- Spectrum (many dimensions) — distance, direction, medium, space
- Reverberation (directional/dispersed sound ratio) — distance, space, environment
- Inter-aural delay time — direction to source (the Haas or precedence effect)

Table 1: Some Mappings between Sound Features and Spatial/Locational Cues

There are several approaches to simulating the cues we use to localize sounds. Some of them are based on direct interpretations and implementations of the physics of sound propagation and the anatomy of our ears—so-called “sound ray-tracing” techniques that use the “pinna transform,” also called the “head-related transfer function” (HRTF)—and others are based on simple psychoacoustical insights. We will present an example of the latter kind of model—the type that we are trying to explore and improve upon in the DIVE auralizer—below.

Figure 1 shows the geometrical model and the basic mapping equations used in current psycho-acoustical models of localization. The distances—shown as *a* and *b* in the figure—between the source and the listener's two ears, which are assumed to be separated by the fixed distance *e*, determine the relative amplitude of the signal that the listener hears. The left-right stereo volume balance is related to the ratio of the difference between *a* - *b*, and *e* (shown by the angle Θ in the horizontal [*x/z*] plane). The amplitude scale factor and direct-to-reverberated signal ratio are proportional to the square of the average distance $d = (a + b) / 2$. The absorbtion of high frequencies by the air between the source and the listener can be modeled as a low-pass filter whose slope (*Q*) and cutoff frequency vary proportional to d^2.



Spatial Cues

Distance (d) = (b + a) / 2
Loudness α d^2
L/R ratio α (b - a) / e
Low-pass filter α d^2
Direct/reverb ratio α d^2
Spatial low-pass filter ratio α Θ
Spatial band-reject filter α Ψ
Inter-aural time delay α (a - b) / e
Initial/decay reverb ratio α Ψ
(many more possible)

Figure 1: APE Spatial Model Parameters

There are several ways to model the spectral changes in a signal according to its direction in the horizontal plane. Making sound sources behind the listener "less bright" than those in front of, or to the side of, the listener is perhaps the most trivial model. This can be simulated simply by a low-pass filter whose *Q* and cutoff frequency vary (according to some non-linear look-up table) with Θ.

The height of a sound source, measured as angle Ψ in the *x/y* plane, effects the inter-aural time delay, the sound's spectrum (via the HRTF and possibly the listener's shoulder), and the characteristics of the reverberation. The spectral effects are complex and still poorly-understood, but can be simulated in our "cheap" model by two filters; A fairly broad band-reject or "notch" filter in the upper speech range of frequencies (2-4 kHz) combined with a mild boost in the 7 kHz region.

The change in the reverberation signal can be approximated by the addition of more early reflections to the signal, simulating echoes off of the "floor," "walls," and "ceiling" of the simulated environment (Kendall 1989) using a ray-tracing approach to reverberator construction.

This simple geometrical model has all the necessary information to map geometrical properties onto more sophisticated or higher-level cues, such as the provision of more complex filters for the front-back and height cues, and more location-dependent reverberation techniques.

There are many other details of localization, such as what cues we use to differentiate between sounds that have the same inter-aural time delay—those that lie on the so-called "cones of confusion" that surround each ear and are symmetrical with respect to the medial plane (the *y/z* plane passing through the listener's head between the ears)—and how we determine the difference between the characteristics of the space and of localized sound sources based on the nature of the signal's reverberation. These are, however, beyond the scope of our present model. The interested reader is referred to (Chowning 1971) or (Moore 1989) for more detailed discussions.

Building an Auralizer for DIVE

Our stated task was to build a platform for developing an audio localization system within the existing synthetic environment (VR system) framework. We will now describe that environment (the MultiG DIVE) in enough detail to provide the background for our introduction of the software architecture of the DIVE auralizer. A more in-depth discussion can be found in (Fahlén 1991) and (Carlsson and Hagsand 1992).

The DIVE Architecture

Most VR systems, are mainly concerned with a 3-D rendering of a simulated virtual world, and with user navigation among, and interaction with, objects in this world (Helsel and Roth 1991). In these systems, some "database" of simulation-style or purely graphical objects exists, along with some data about the "user's" position in the world (the position and perspective from which the world will be rendered). This data is used by the rendering component, which generates a visual display. The user interaction, database management, and rendering systems can be relatively independent of each other (as they are in the DIVE), with the interface asynchronously driven by changes in the user's position and the state of the objects in the world each time the renderer generates a new frame of video information for display. The renderer can thus be said to "poll" the world and user state for each frame it displays.

The basic unit of activity in the DIVE is the AI, or application—a UNIX process running on some node in the network that updates the shared world and object databases or broadcasts change mes-

sages via the system's event distribution mechanism (currently based on the ISIS distributed programming tool kit [Birman and Cooper 1990]). An AI might, for example, represent a clock that updates itself every second, distributing messages throughout the network as to its new visual appearance. Visualization AIs that are in the same world as the clock would receive this message, and update their renderings of it in case it is in the user's field of vision.

Nodes that mainly interact with the shared database are called "computation nodes" (CNs) and nodes that interact with the database and the user interaction and some kind of visualization are called "visualization and interaction nodes" (VINs).The DIVE architecture allows multiple "virtual worlds" (which may or may not be connected by "gateways") to be active on the same network, with one or more users in each of them at any one time. It is this distributed, multi-world, multi-user and non-server architecture that is perhaps the main interesting feature of DIVE. Figure 2 shows a schematic overview of the architecture. Visualization and interaction nodes manage the graphical output and gestural input for the user. The visualization nodes can potentially support stereo-optic output systems such as head-mounted stereo displays. Computation nodes can execute AI processes such as ticking clocks (i.e., those that generate animated, possibly with some kind of "semi-intelligent" behavior, virtual objects in the synthetic world), or "supporting" processes such as a collision detector or gravity simulator, which update the state of other objects in the world database without introducing any of their own. The degenerate case is that all of these processes are running on the same workstation, though we use between three and six in common practice.

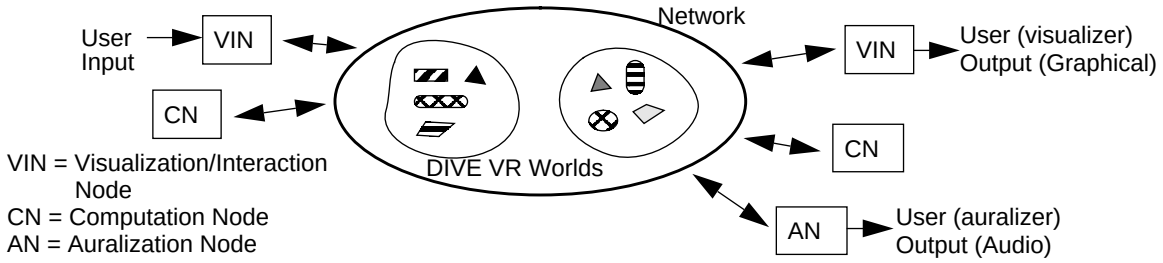


Figure 2: DIVE Architecture; see also (Fahlén 1991) and (Carlsson and Hagsand 1992)

The Auralizer node shown in the lower-right part of Figure 2 is the component that we have added to the system in this project. Like a visualizer, it will "render" the state of the world object data base, but using audio (2 channels at present) instead of video as its output medium.

This architecture is illustrated in a different schematic representation in Figure 3, which shows the flow of control, and the residency of world data between an AI and the processes in a VIN. The

AI also has its own copy of the shared memory (not shown here); more importantly it sends out messages that are distributed among the processes in its world (e.g., other AIs or visualizers). These messages will generally change the state of the shared object data in each participating member. The input server—e.g., one connected to a 3-D input device such as a magnetic tracker—updates the visualizer's notion of the location and orientation of the virtual "user," which defines the perspective from which the graphical model will be rendered. The visualizer is broken up into the components called the visualizer proper, the vehicle (which controls the mapping between the input device and changes in the user's location and orientation), and the renderer (which generates image frames at some specified target rate, typically 10-20 Hz).

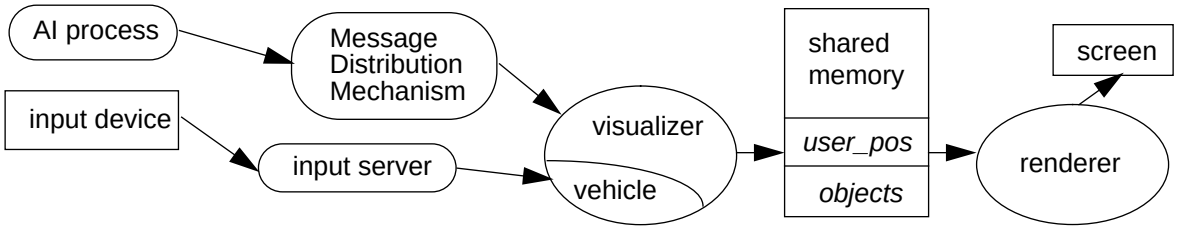


Figure 3: DIVE Interaction and Visualization Control and Data Flow

Visual and Aural Rendering

To incorporate audio output into such a VR system, it is necessary to extend the simulated object model of the virtual world so that "events" can trigger sounds, and so that continuous sounds can be produced by AIs or other components. This means that the auralizer must have a different architecture than a visualizer/renderer, and must be connected to AIs in a different manner. Returning to the clock example we mentioned above, if the clock advances one second, it changes its graphical rendition as it is stored in the shared database (it sends out a message via DIVE's distribution mechanism), and assumes that any renderers that are "watching" will update their renditions of it (perspectives on it), as appropriate. If this clock wants to make a "tick" sound (or to say the word "tick," or to play an e-minor guitar chord), it is likely that it must broadcast that information in a different format than it uses for updating its graphical perspective. This is because of the architecture described above, whereby renderers "poll" the state of object memory when rendering each frame of graphics, whereas sound events can (and will) occur asynchronously and must be scheduled with less latency than graphical ones. The fact that audio is not "frame-based" is an important difference that will naturally effect how the two media are handled differently.

There are a great many issues in the software architecture of an aural renderer for a system such as DIVE; they are described in more detail in (Pope and Fahlén 1993) and primarily involve how one models event distribution vs. polling for handling sound triggering and user position updates, and the software factoring of the digital signal processing needed for implementing the localization model so as to be portable, easily extensible, and distributable among several workstations. We have also avoided the issue of AIs that generate their sounds in real time by requiring them to pre-register sound samples, giving a sound file name and a sound index and perspective (which can be thought of as a sample name and channel and key numbers in MIDI synthesizer terminology). We do this because we do not want to have to address the issues of network bandwidth with multiple real-time sound sample streams, or of abstract timbre description languages, in the present system, which is intended for experimenting with localization models and 3D audio environments.

The DIVE Auralizer

AI programs "register" sounds with the Auralizer by sending out messages that include their own unique identifiers ("client IDs"), a sound file name, and a sound index and perspective. The AI can later "play" this sound by sending out a message that includes the above information (without the sound file name), along with their current positions, and the relative amplitude of the sound (and a few other parameters that are beyond the scope of this introduction—see below).

The Auralizer runs as a separate process, probably on a network node with high-quality stereo audio output hardware, such as a suitably-equipped Sun SPARCstation or SGI Indigo. It maintains a table of the sounds that have been registered by AIs in the current world and responds to sound output messages by playing the chosen sounds, possibly mixing them with other active sounds and spatializing the result to provide for an aural model of the virtual space. To trigger a sound, the Auralizer needs to know which sound is being requested, the position of the sound source (possibly in motion), the position and orientation of the listener, and the characteristics of the virtual space. It will use this information to select and process the stored sample according to the source/filter/listener model described above. There can be several types or levels of auralizers depending on the amount of computation power that can be dedicated to the auralization (i.e., if it is running on the same workstation as the visual renderer, or if special digital signal processing (DSP) hardware is provided), and on which output format is being used (e.g., monophonic CODEC or multichannel

CD-quality output). The first auralizer that was built mixed 16-bit 8 kHz monophonic sound samples into a stereo aural image using left-right sound placement and amplitude as spatial cues. The current generation runs at 16 kHz and incorporates stereo reverberation, a simple filter-based front-back directional model and inter-ear time delay. More sophisticated filters (modeling the HRTF), and dynamic reverberation algorithms to provide up-down directional cues and more stable cues in general are planned.

The method of integration of the Auralizer into the DIVE system was debated for some time, and several approaches were prototyped. The final design was factored into several components: (1) the interface to the network distribution mechanism; (2) sample structure storage and management; (3) the geometry functions for determining the relative locations of sound sources and mapping the geometry onto the parameters of the mixer, filter and reverberator; (4) sound mixer, filter and reverberator; and (5) the real-time output handlers for the DACs. The rough architecture of the APE and its interconnection with the rest of DIVE is illustrated in Figure 4.

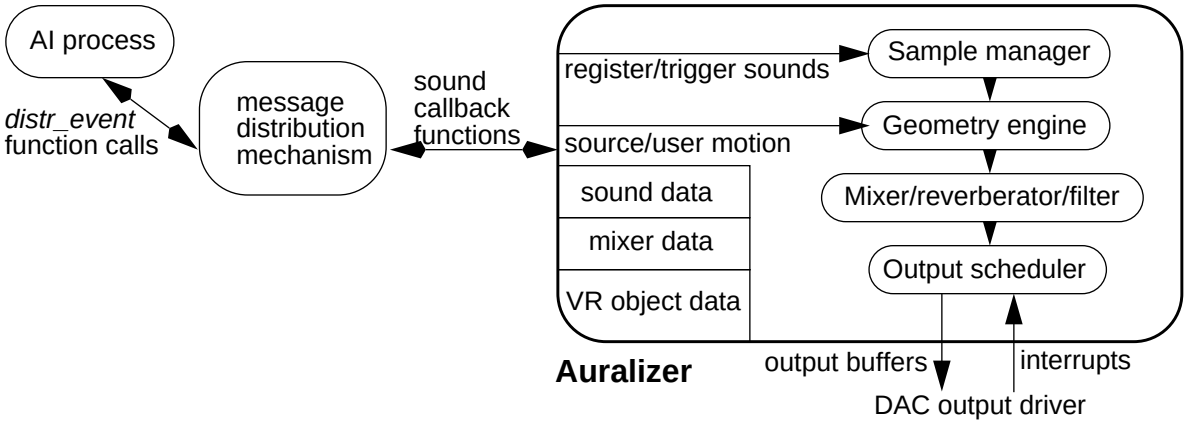


Figure 4: DIVE Auralizer Architecture

The auralizer's sample storage uses an model of a multi-dimensional array mentioned above. The items stored in this array are special structures that include the actual samples of the sound, its duration, sample rate, sample format, and other data. When a client registers a new sound, the auralizer allocates space for the sound's sample array, reads the samples from the disk, and places the sound structure in the array such that it can be located later when the client wants to play it. (The addressing and structure management to do this in real time is non-trivial.) The second component of the Auralizer is its geometry functions. Each time a client plays a sound, it is possible that both the client and the listener may have moved, so it is necessary to get the coordinates and orientation

of the user (defined as the user's "eye" vector in DIVE), and of the client AI in order to determine the relative position and amplitude of the source. In our simple initial system, we used the relative positions to set the stereo position and scale the loudness of the sound samples. The more sophisticated current models also use inter-aural time delay, non-linear front-back filters, and a configurable reverberator. A client can also register an event for continuous updating of its position if, for example, it is moving quickly or is making a sound with a long duration. This should probably be the default case, but the current mechanism for calculating relative locations continuously was thought too computationally expensive for the present system.

The sound mixer functions can manage multiple active clients at different locations, and perform the summation of up to 32 monophonic sources into a stereo output stream. The reverberation model adds a small amount of reverberation to the mixed sound sample array. The ratio of the direct and reverberated signals is dependent on the distance between the source and the listener. The interface to the network distribution mechanism uses the DIVE's platform-independent tools to communicate via the ISIS system with AIs and visualizers. It registers handlers for the collection of callback functions that AIs can call with sound output, source movement, or other auralizer-related messages. Finally, the real-time drivers and interrupt handlers for the sound output via the DACs are implemented so that different DAC hardware can be substituted with relative ease (hopefully). The real-time I/O module must handle the interface to the low-level sound output functions or device driver, and the setup and clean-up for each mix block. It must also know what to do if the system cannot keep up with real-time, i.e., if the next buffer is not ready when the output driver generates an interrupt.

Applications

A number of application areas have been targeted for our testing and further development of the DIVE auralizer. These include extending the available modalities of the user interface in order to increase naturalness and reduction of cognitive load. Sophisticated use of sound can often be a workable substitute for (the very hard to achieve) tactile feedback. The system will also be used in teleconferencing systems to give cues about activities that goes on "off camera" such as participants entering and leaving the conference, networks going down etc. We hope to be able to build more fully featured environments for artistic/esthetic expression, i.e. sound sculptures, dance, music compositions, virtual instruments and interactive experiences such as films and games and so

on. A related project aims to make it possible for sight impaired persons to use and participate in activities in synthetic environments with little or no disadvantages.

Evaluation

The DIVE auralizer was developed through several phases during the Summer and Autumn of 1992. The primary platform was a Sun SPARCstation 2 with Ariel Corp. digital-to-analog convertors. It was tested in a DIVE network running multiple sound-generating AIs (mostly ticking clocks) on various nodes. We implemented the spatial cues of loudness, inter-aural balance, inter-aural time delay, directional filtering in the horizontal plane, and simple reverberation (fixed reverberator parameters with mixing of direct and reverberated signals). The system was found to support sustained real-time output with between five and ten AIs making noises and to provide a "reasonably good" spatial model, with the expected problems related to the "cones of confusion" and missing height cues.

Future Work and Conclusion

As mentioned above, because the Auralizer is being built in the context of a project whose goals are understanding distributed programming on very-high-speed networks, and the development of new techniques and tools for supporting this, it was not our intention to place a great deal of emphasis on the exact quality of the spatial model. The Auralizer was designed for maximal "pluggability" so that modules can be changed, and more sophisticated ones substituted after this initial implementation. Another design criterium was to have loose coupling between the Auralizers components, so that they can be distributed over a network in various fashions as we build more complex models and worlds. Examples of the "pluggability" would be to the planned porting of the real-time output layer to other hardware platforms (e.g. the SGI Indigo), move the sample storage to an object-oriented database management system, and the running the mixing, reverberation and spatial filtering components on a special distributed real-time signal processing framework that will be the topic of a future project.

We hope to build better geometry subsystems, fancier spatial models, and distributed mixing components in the future while retaining the basic APE architecture. One important issue that has not been addressed in the present paper is the handling of multiple worlds and movement between those. We also think it is important to build, as soon as possible, non trivial applications and have

users experiment with them in order for us to gain experience both from the "systems" standpoint and from a "user interface" standpoint.

References

Begault, D. R. 1991. "Challenges to the Successful Implementation of 3-D Sound." *Journal of Audio Engineering Society* 39(2): 864-870.

Birman, K., and R. Cooper. 1990. "The ISIS Project: Real Experience with a Fault-Tolerant Programming System." *Proceedings of the Workshop on Fault-Tolerant Distributed Systems*. New York: ACM Press.

Blauert, J. 1983. *Spatial Hearing*. Cambridge, Massachusetts: MIT Press.

Carlsson, C. and Hagsand, O. 1992. "The Architecture of the MultiG Distributed Interactive Virtual Environment." *Proceedings of the Fifth MultiG Workshop*. Stockholm: KTH.

Chowning, J. 1971. "The Simulation of Moving Sound Sources." *Journal of Audio Engineering Society* 19: 2-6.

Durlach, et al. 1992. "On the Externalization of Auditory Images." *Presence: Telepresence and Virtual Environments* 1(2): 251-257.

Fahlén, L. E. 1991. "The MultiG Telepresence System." *Proceedings of the Third MultiG Workshop*. Stockholm: KTH.

Helsel, S., and J. Roth, eds. 1991. *Virtual Reality Theory, Practice, and Promise*. Menlo Park, California: Addison-Wesley.

Kendall, G. S. et al. 1989. "Spatial Reverberation: Discussion and Demonstration." in M. V. Mathews and J. R. Pierce, eds. *Current Directions in Computer Music Research.* Cambridge, Massachusetts: MIT Press: 89-103.

Moore, F. R. 1989. "Spatialization of Sounds over Loudspeakers." in M. V. Mathews and J. R. Pierce, eds. *Current Directions in Computer Music Research*, Cambridge, Massachusetts: MIT Press: 65-87

Nordmark, J. O. 1976. "Binaural Time Discrimination." *Journal of Acoustical Society of America* Vol 60: 870-880.

Pope, S. T. and Fahlén, L. E. 1993. "The DIVE Auralizer." *SICS Report* Stockholm: SICS (forthcoming).

Wenzel, E. M. 1992. "Localization in Virtual Acoustic Displays." *Presence: Telepresence and Virtual Environments* 1(1): 80-107