

# The Use of 3-D Audio in a Synthetic Environment: An Aural Renderer for a Distributed Virtual Reality System

Stephen Travis Pope and Lennart E. Fahlén

DSLab — Swedish Institute for Computer Science (SICS)

Isafjordsgatan 26, Kista, S-16428 Sweden

Fax: +46/8/751-7230; E-mail: stp@sics.se, lef@sics.se

## Abstract

We have investigated the addition of spatially-localized sound to an existing graphics-oriented synthetic environment (virtual reality system). 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 was built as a test-bed for experimenting with the known techniques for generating sound localization cues based on the geometrical models available in a synthetic 3-D world. This paper introduces the psycho-acoustical background of sound localization, and then describes the design and usage of the DIVE auralizer. We close by evaluating the system’s implementation and performance.

## 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) [1] [2]

is a virtual reality (VR) framework for building applications and user interfaces based on the metaphor of presence and interaction in 3-D-space. It uses graphical rendering—visualizers—to display objects in simulated 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” [3].

The DIVE is a multi-user virtual reality system that is implemented within a loosely-coupled heterogeneous multiprocessing environment (i.e., a network of different UNIX workstations). Independent DIVE applications (called “AIs” after W. Gibson), run on nodes distributed within a local- or wide-area network, and update a shared (node-wise replicated) object database. This architecture enables the distribution of object animation and interaction processes for load-sharing between the user input device handling, and graphical rendering tasks.

The auralizer models sounds that take place in the simulated world using a “source/filter/listener” model of sound processing. It can also be used as a stand-alone system for 3-D rendering 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.

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., [4] or [5]). Also, a number of expensive hardware systems aimed at high-end music studio and film post-production work have 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 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 magnetic trackers.

This document discusses the design of the DIVE auralizer. 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 model-based sound spatialization. We then discuss the auralizer as a real-time, distributed DSP 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 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, and that there are other cues (e.g., Doppler shift, or inter-ear delays), and higher level functions in the brain (see e.g., [6] or [7]) that we use to localize sound sources.

The basic model of sound localization uses direction and distance to characterize the relative geometrical orientation of the sound source and the listener, and provides 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 research.

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

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.

- Amplitude (loudness)  
distance to source
- Position (in 3D space)  
direction to source
- Spectrum (many dimensions)  
distance, direction, space
- Reverberation (ratio/characteristic)  
distance, space, environment
- Inter-aural delay time  
direction to source (Haas effect)

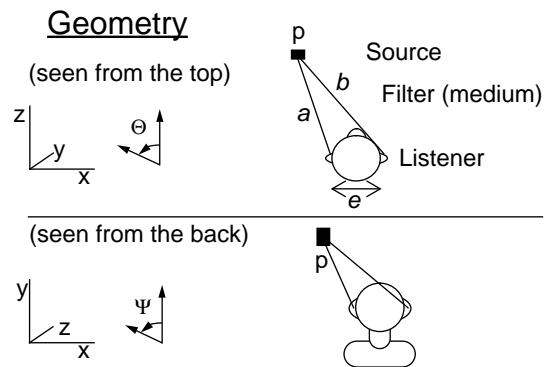
**Table 1: Possible mappings between sound features and spatial cues**

Figure 1 below shows the geometrical model and the basic mapping equations used in current psychoacoustical 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  $\theta$  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 absorption 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  $d^2$ .

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  $\theta$ . The height of a sound source, measured as angle  $\psi$  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 are 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.



**Figure 1a: Auralizer spatial model**

The change in the reverberation signal can be approximated by the addition of more early reflections to the signal, simulating echoes off of the walls and ceiling of the simulated environment.

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

**Figure 1b: Spatial localization cues**

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”

### Spatial Cues

Distance (d) = (b + a) / 2

Loudness  $\propto d^2$

L/R ratio  $\propto (b - a) / e$

Low-pass filter  $\propto d^2$

Direct/reverb ratio  $\propto d^2$

Spatial low-pass filter ratio  $\propto \Theta$

Spatial band-reject filter  $\propto \Psi$

Inter-aural time delay  $\propto (a - b) / e$

Initial/decay reverb ratio  $\propto \Psi$

(many more possible)

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 reader is referred to [9] or [10] for more detailed discussions.

### **Building an Auralizer for DIVE**

We will now describe that environment (the SICS MultiG DIVE) in enough detail to provide the background for our introduction of the software architecture of the DIVE auralizer. For a more in-depth discussion, see [1] and [2].

#### *The DIVE Architecture*

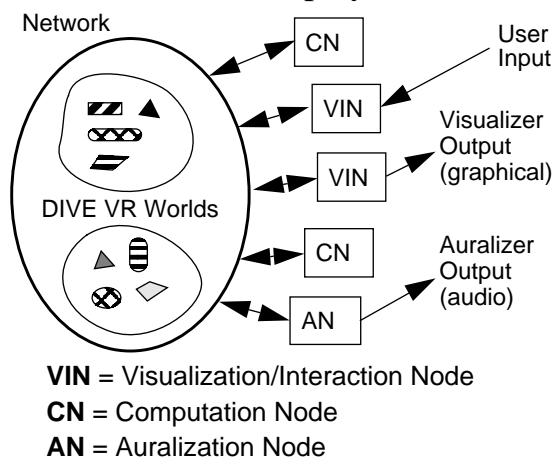
Most VR systems are mainly concerned with a 3-D rendering of a simulated virtual world, and with user navigation among, and interaction with, objects in this world [8]. In these systems, some "database" of exists, along with some data about the "user's" posi-

tion 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 broadcasts change messages via the system's event distribution mechanism. 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).

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. This distributed, multi-world, multi-user, non-server (peer-to-peer) architecture is perhaps the most 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 displays.



**Figure 2: DIVE architecture**

Computation nodes can execute AI processes such as ticking clocks, 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.

The Auralizer node shown in the lower-right part of Figure 2 is the

component that we have added in this project. Like a visualizer, it will “render” the state of the world object data base, but using stereophonic audio instead of video as its output medium.

### *The DIVE Auralizer*

AI programs “register” sounds with the auralizer by sending out messages that include a unique identifier, a sound file name, and a sound index, and a “perspective” (which can be thought of as a sample name and channel and key numbers in MIDI synthesizer terminology). The AI can later play this sound by sending out a message that includes the id and indices, and the relative amplitude of the sound.

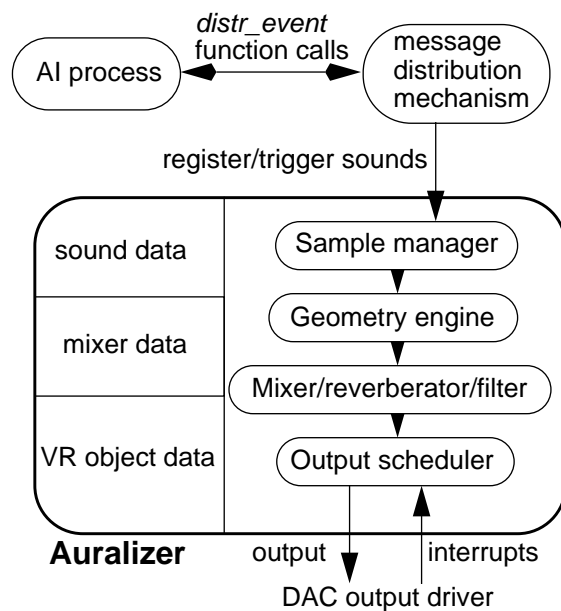
The auralizer runs as a separate process, probably on a network node with high-quality stereo audio output hardware. It maintains a table of the sounds that have been registered by AIs in the current world and responds to sound output messages by playing the chosen sounds, possibly mixing them with other 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. 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 DSP hardware is used).

The architecture of the auralizer 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) the 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 3. The system runs as three “threads” (light-weight processes): the event, the geometry, and output loops.

When an AI registers a sound, the sample manager stores the sample data of the sound, its duration, rate, format, and other data. The second component is the geometry engine. Each time a client plays a sound, it is necessary to get

the coordinates and orientation of the user (defined as the user’s “ear” objects), and of the AI in order to determine the relative position and amplitude of the source. This data is updated while the sound is playing by the geometry thread, as both the source and the listener may be in motion.



**Figure 3: Auralizer Architecture**

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 (mixed according to the geometry engine’s data) to the mixed sound array. The DIVE’s distribution mechanism communicates between AIs and visualizers. It registers handlers for the collection of callback functions that AIs can call with sound output 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.

### **Applications**

A number of application areas have been targeted for our testing and further development of the DIVE auralizer. The aim in general-purpose VR-based user interfaces is to increase the naturalness and reduce the cognitive load of the interface. The initial applications were various ticking clocks and bouncing balls. These were used to debug both the system and its initial spatial models. More complex applications to date include an integrated system using the ANIMA choreography [11] system together with DIVE visualizers and auralizers for dance.

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. We hope to be able to build more fully featured environments for artistic/aesthetic expression, i.e., sound sculptures, music compositions, virtual instruments and interactive experiences such as films and games. A related project aims to make it possible for sight-impaired persons to participate in

activities in synthetic environments.

### **Evaluation**

The DIVE auralizer was developed through several phases during the Summer of 1992. The initial 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 on various nodes. We implemented the spatial cues of loudness, inter-aural balance, 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. The system was ported to the Silicon Graphics Indigo and Sun SPARCstation-10 platforms in April of 1993, where it ran significantly faster and needed no additional hardware for sound output (on the Indigo).

The first auralizer that was built mixed 16-bit 8 kHz monophonic sound samples into a stereo aural image. The current generation runs at 16 kHz and incorporates better stereo reverberation and inter-aural time delay. More sophis-

ticated filters (modeling the HRTF more closely), and dynamic reverberation algorithms are planned.

We have also avoided the issue of AIs that generate their sounds in real time by requiring them to pre-register sound samples. 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. The experimental ANIMA system uses disk-based samples that may be too large to fit in memory, but still cannot handle real-time sample streams passed over the network.

Due to delays in the network distribution system, and the relatively slow (10 Hz) frame rate of visualizers, exact synchronization of visual and aural events is difficult. We are currently investigating ways to improve this.

It remains to be seen just how good we can make our spatial models and maintain real-time performance on “reasonable” hardware. The auralizer is written entirely in C and does not use the DSP coprocessors of the platforms it runs on. More complex reverberators and spatial filters can be expected to “push the envelope” of this restriction.

## **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 more sophisticated modules can be substituted after this initial implementation. Another design criterium was to have loose coupling between the auralizer’s components, so that they can themselves be distributed over a network in various fashions. Examples of the “pluggability” are the ports of the output stage mentioned above, moving 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.

We hope to build better geometry subsystems, fancier spatial models, and distributed mixing components in the future while retaining the basic auralizer design. 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

[1] L. E. Fahlén: “The MultiG Telepresence System.” in *Proceedings of the Third MultiG Workshop*. Stockholm: Royal Institute of Technology, 1991.

[2] C. Carlsson, O. Hagsand: “The Architecture of the MultiG Distributed Interactive Virtual Environment.” *Proceedings of the Fifth MultiG Workshop*. Stockholm: Royal Institute of Technology, 1992.

[3] S. T. Pope, L. E. Fahlén: “Building Sound into a Virtual Environment.” in *Proceedings of the Fifth MultiG Workshop*, Stockholm: Royal Institute of Technology, 1992.

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

[5] E. M. Wenzel: “Localization in Virtual Acoustic Displays.” *Presence: Telepresence and Virtual Environments* 1(1): 80-107, 1992.

[6] J. Blauert: *Spatial Hearing*. MIT Press, 1983.

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

[8] S. Helsel, J. Roth eds.: *Virtual Reality Theory, Practice, and Promise*. Addison-Wesley, 1991.

[9] J. Chowning: “The Simulation of Moving Sound Sources.” *Journal of Audio Engineering Society* 19: 2-6, 1971.

[10] F. R. Moore: “Spatialization of Sounds over Loudspeakers.” in M. V. Mathews and J. R. Pierce, eds. *Current Directions in Computer Music Research*, MIT Press: 65-87, 1989.

[11] T. Ungvary, S. Waters, P. Rajka: “Nuntius: A Computer System for the Interactive Composition and Analysis of Music and Dance.” *Leonardo*, 25(1): 59-68, 1992.