CompuGraphics'97 ©*GRASP*

TOOLS AND WIDGETS FOR SPATIAL SOUND AUTHORING

Jens HERDER
University of Aizu
Fukushima-ken 965-8580, JAPAN
voice: [+81](242)37-2618 fax: [+81](242)37-2706
herder@u-aizu.ac.jp
http://www.u-aizu.ac.jp/~herder

ABSTRACT

Broader use of virtual reality environments and sophisticated animations spawn a need for spatial sound. Until now, spatial sound design has been based very much on experience and trial and error. Most effects are hand-crafted, because good design tools for spatial sound do not exist. This paper discusses spatial sound authoring and its applications, including shared virtual reality environments based on VRML. New utilities introduced by this research are an inspector for sound sources, an interactive resource manager, and a visual soundscape manipulator. The tools are part of a sound spatialization framework and allow a designer/author of multimedia content to monitor and debug sound events. Resource constraints like limited sound spatialization channels can also be simulated.

Keywords: spatial sound authoring, virtual reality environments, multimedia, spatialization, spatial media, visualization, user interface design, man-machine interfaces

INTRODUCTION

More and more applications use virtual reality environments including spatial sound as a user interface. The demand for animations with impressive and immersive sound increases, as audio-visual equipment which can produce such effects becomes increasingly available. Spatial sound has migrated from special platforms to everyone's desktop. This is due to better general-purpose processors, which allow spatial sound processing in software, and hardware support, in the form of audio cards.

Wide distribution of content including spatial sound for virtual reality environments over the internet was made possible with the introduction of the Virtual Reality Modeling Language (VRML 2.0 [Bell *et al.*96]). Even though the specification does not cover all aspects of spatial sound, dramatic effects can be produced. Available tools for producing VRML content have some support for spatial sound authoring, but in general they are not sufficient. We have developed widgets [Conner *et al.*92], user interface objects with encapsulated geometry and behavior, to control and show properties of soundscapes and sound objects.

Spatial sound Spatial sound [Anderson-Casey97] is modeled by many attributes. Directionalization can be modeled by binaural ITD (interaural time difference), IID (interaural intensity difference), and pinnae effects [Begault94]. Distance cues are based on delay, reverberation and loudness. The space, consisting out of objects interacting with sound waves, defines the reverberation and first- (and higher-) order reflections.

Important for this paper is the following definition:

> A "soundscape" is a spatial region in which sound sources, sound sinks, and interactions between objects with sonic results can share common media, geometries, and spatial mappings. The media define such sonic features as the speed of sound within the soundscape, but can be extended to include sonic reflecting materials and reverberation attributes. The geometries and spatial mapping define how sound may be distributed for sophisticated and/or efficient audio rendering.

Authoring: Creativity and engineering For spatial sound authoring, two disciplines converge. An artist or content producer with strong emphasis on creativity brings the ideas or defines what should be done. Complementarily, skills from engineering are needed to implement the space and actually produce the effects. An user interface for artists should be intuitive and easily understandable. For the same reason in computer graphics different color models and naming schemes have been developed [Kaufman96]. For the sound designer, realism is not so important as the achieved impression. For example [Martens97], one might think about an virtual environment for flying starships, like that in movies and interactive games. Usually they produce sound even in outerspace sound waves do not travel through empty space. The user of a spatial sound authoring tool will create a soundscape with sound sinks, sound sources, and objects interacting with the sound waves. The sound objects have to be positioned and might be attached to geometrical objects. Their attributes need to be set. A development cycle comprising visualization, experiencing (i.e., listening with interaction), coverage checking, experimenting, evaluation and editing starts. Spatial attributes like the image breadth also known as localization blur [Blauert96, page 280], are important because they suggest the size of an object. In linear media (e.g., movies) this is not a big problem because sink (i.e., listener) positions are fixed. In interactive media the listener has total freedom to move around a sound objects (e.g., a piano that will be walked around cannot be well expressed by a point sound source).

Requirements Requirements for a spatial sound authoring toolset on which we have concentrated are

- soundscape visualization,
- soundscape manipulation,
- sound object visualization,
- sound object editing, and
- sound resource monitoring.

When developing an authoring tool for spatial sound, the underlying system, described by the spatial sound API, defines a lot of the functionality and might restrict the generality and portability of the application. It is important to keep in mind that the main task for the author is to develop content, and that rendering issues across different platforms should not interfere. An example of modeling rendering issues was given in [Brown-Allard97]: the attenuation of the frequency spectrum of a waterfall (part of a "Jungle Island" demonstration) was modeled by two sound sources with different audible ranges and frequency bands. This made it possible to hear the rumbling of the waterfall in the distance as well as high frequency components when sufficiently close. The effect was impressive, but what if the sound renderer supports distance-dependent frequency attenuation? Would not it be better if the API and also the sound authoring tools hid such details from the user? The same philosophy can be taken for sound occluders and first-order reflections.

PREVIOUS RESEARCH

Spatial Sound Application Programmer Interfaces

The current VRML [Bell *et al.*96] specification has only a sound node to support spatial sound, and does not define soundscape attributes to describe reverberation. A browser might guess the size of the space and then set important reverberation parameters for sound spatialization. The Java3D [Sowizral *et al.*97] specification is in that regard more advanced, supporting a notion of a soundscape, an application area with aural attributes capturing delay times and reflections. These APIs are good for multimedia content, but are not suitable for simulating room acoustics [Savioja *et al.*97][Dalenbäck *et al.*94], which are much more complicated and require a more physical approach — specification of the material and transfer functions of sound objects (e.g., a wall) in the simulated space. Such simulations are not yet done in realtime. Realtime processing becomes possible if the room parameters are processed beforehand [Reilly-McGrath95].

Spatial Sound Authoring Systems

Most spatial sound authoring systems are closed and do not allow users to develop content for different backend configurations. A multiple audio window system [Cohen93] gives each user a visual, egocentric view on a scene and allows realtime interaction and sound object editing based on direct manipulations and cut & paste metaphors.

In some shared virtual environments (e.g., AlphaWorld) [Waters-Barrus97], users not only explore and meet, they also extend and build the space which they inhabit. Building such a space which is part of a larger system includes sound. The restrictions/constraints in such groupware applications are even tighter, to avoid the social infrastructure becoming damaged through the creation of areas which are inaccessible due to resource load on either the server or client side.

Another important topic for authoring auditory scenes is the temporal relationship between sounds. This is addressed in [Darvishi97], which employed a 2D graph based or textual interface.

SOUND SPATIALIZATION DEVELOPMENT ENVIRONMENT

Our prototype environment consists of a library to manage sound processing, a visual soundscape controller (to handle mapping between geometric application space and soundscape), a sound resource allocation monitor, a soundscape visualizer, and an editor for sound objects. The following subsections introduce the modules and the data flow.

Soundscape Control

During the design of a helical keyboard [Herder-Cohen96], we became aware that global control of the mapping between visual space and acoustical space could improve the intended experience. A major part in the design was the soundscape. All keys had to be differentiable by location, especially direction. If the listener is placed in the center, then keys playing in the far upper part or lower part could not be well differentiated by azimuth, and also the volume for them was too low. As a solution to these problems we developed the soundscape deformer, a 3D widget that controls the scene space → soundscape mapping[1]. The scene space can be shifted around, which induces a translation of the soundscape. In that regard the soundscape deformer can be seen as a generalization of stereo panning for 3D (e.g., balance potentiometer of an amplifier, also known as a pan [for panoramic] pot).

The soundscape deformer, shown in Figure 1, provides a visual representation of a linear mapping. A sphere in the center represents the case in which the scene space is directly mapped to the soundscape. The soundscape is fully interactively manipulatable via handles (the small boxes on the outside), bounding box, and orientation axes through a pointing device. Figure 2 shows the soundscape reduced in height, flattening the spa-

[1] the mapping itself can be represented by a 4x4 matrix

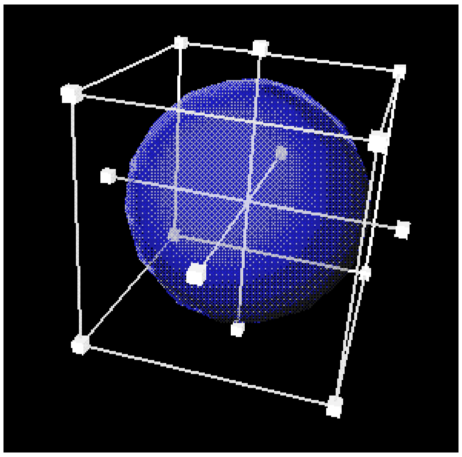

Figure 1: Soundscape deformer

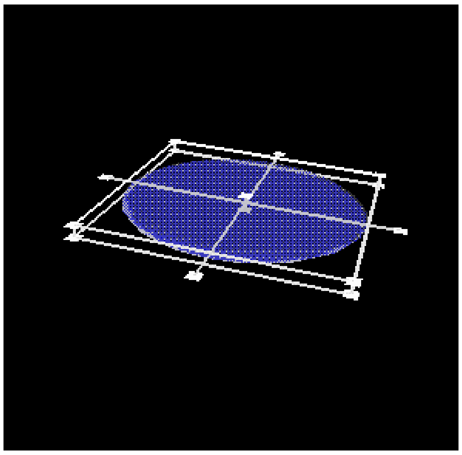

Figure 2: Soundscape deformer: flattening

tial audio position of all sound objects to a plane. Another example, shown in Figure 3, reduces the horizontal dimension, compressing the left↔right attribute. As an extreme example, shown in Figure 4, the sphere can be reduced to a point, giving a diotic soundscape, in which all objects seem to be at one place inside the user's head.

Portable Content: Authoring For Different Platforms

A commercial application or multimedia product might be required to run on different platforms, which heterogeneity must be considered when doing spatial sound authoring.

Output devices Backends vary; a system can use loudspeakers in single, stereo, transaural (stereo with crosstalk cancelation),

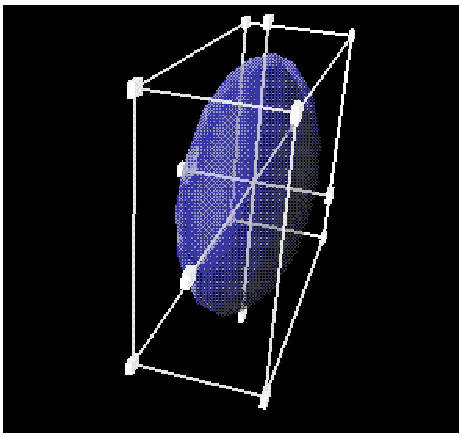

Figure 3: Soundscape deformer: narrowing

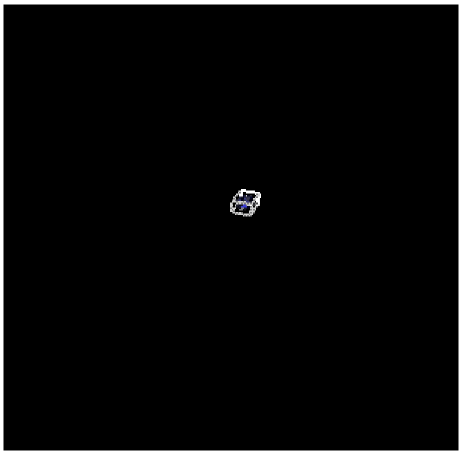

Figure 4: Soundscape deformer: extreme diotic case

and array [Amano *et al.*96] configurations. Headphones or nearphones are also widely used. Across these devices the amount of immersion or believable illusion differs drastically. Despite the fact that most people don't have an absolute tone hearing, the frequency spectra of the output device needs to be adjusted and considered.[2] If a headphone cannot produce a rumbling sensation in the stomach, then the author who creates multimedia content for a broad range of platforms needs to consider that, and might program a different or additional acoustical event.

Spatialization backends The design of spatialization backends depends on the output devices surveyed in the paragraph above. Part of the spatialization process involves processing filter functions (convolution) and reverberation. Such processing can be done either in hardware [Wenzel *et al.*90] or software [Intel97]. In the later case the main CPU load might increase unacceptably if the spatial sound design did not anticipate such scalability. Otherwise the software for spatialization disables resource allocation and the acoustical effect cannot be achieved.

Sound processing Sound processing — in the form of audio (e.g., `wav`) files, MIDI synthesis, or physical models — may use system resources and compete with other processes like spatialization. A good system maintains balance and optimizes for the user based on psychoacoustic metrics.

Monitoring Sound Resource Allocation

How can the above mentioned problems be addressed during the process of spatial sound authoring? One impractical solution would be to have all platforms available and to do tests, but even so not all configurations could be covered.

We propose to monitor resource requests during the authoring process. This will help to inform the author and sound developer about active sources and resource allocation. We have developed a sound spatialization resource manager [Herder-Cohen97], including a monitor for the requests and allocations. The panel shown in Figure 5 gives access to the number of sound sources and sinks, the number of active (i.e., requested) sources, the number of ambient sources, and the number of virtual sources in a scene. Virtual sources represent a cluster of sound sources which can be spatialized as a single source, mixing the audio signals before the spatialization is performed. Ambient sources do not use spatialization resources, but produce a load for the sound generation mechanism.

[2]For example, the head-related transfer function needs to be equalized for the headphone in use. Even better would be individual HRTFs.

Simulating Resource Allocation Via Resource Constraints

How would a certain kind of content be produced on a system with few spatialization channels? Such simulation of spatialization resource manager resources (i.e., the number of spatialization channels) can be dynamically constrained via the control panel (as seen in Figure 5). This allows monitoring of resource allocation across finite capabilities. In the same way of course it is made audible and allows the designer to compare different configurations.

Spatialization Resource Visualizer

The spatialization resource visualizer is an inspector for sound objects in the soundscape, a visual debugger for sound objects in virtual reality environments. These are sound sources and sinks, generalization of listener and microphone. Figure 6 shows on the left a test scene with the corresponding sound objects in the visualizer on the right side. A special case involves virtual sound sources which do not exist in the virtual reality environment scene and are generated during the clustering process of the spatialization resource manager. The set of sound sources can be selected (using the preference menu, seen in Figure 5) to focus on all, active, virtual, or ambient sound sources. A sound source can be displayed using its core range, which is an ellipsoid, representing a zone with maximum intensity [Bell *et al.*96] and its audible range, shown as a translucent ellipsoid, representing the space in which the source is audible.

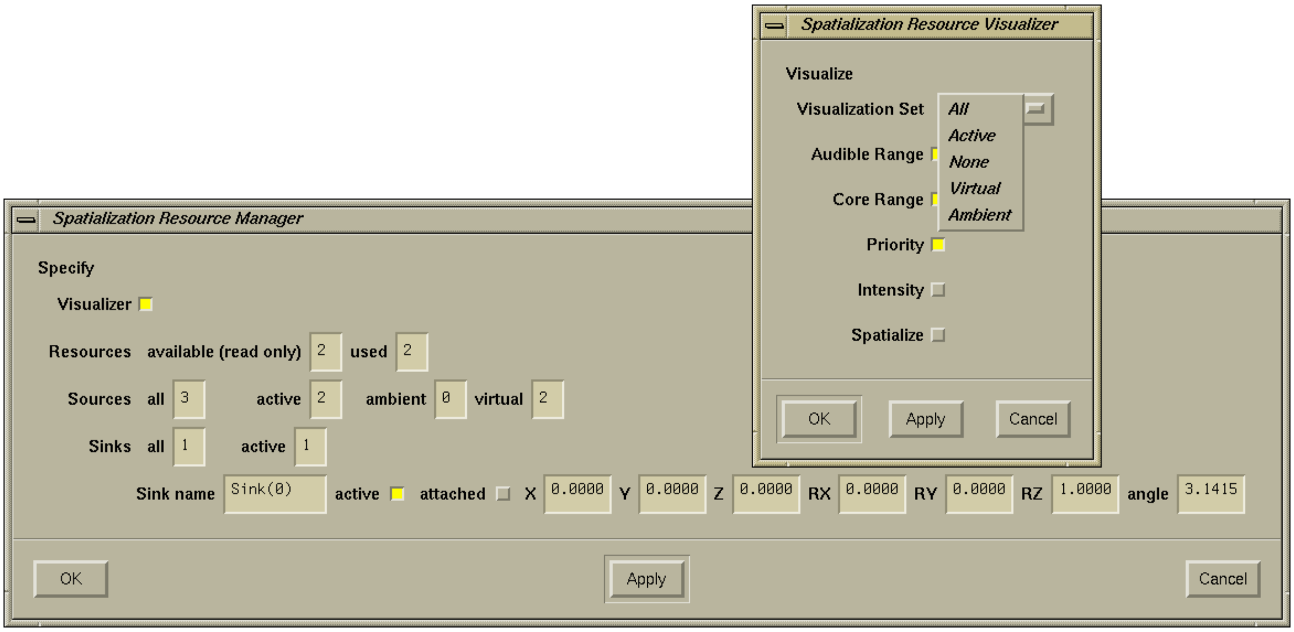

Figure 5: Sound spatialization resource manager panel

Between the two ranges the intensity drops off according to the square of the distance. Priority and intensity of the sound node may be included as textual values facing the user. Different states of a sound node can be conveyed using color codes for the core range.

Sound Source Editor

The editor shown in Figure 7 allows one to edit and monitor sound nodes [Bell *et al.*96] in a virtual reality runtime environment. The user invokes the editor via mouse click on a sound node in the spatialization resource visualizer.

A source radiation pattern is defined by a core range and audible range, represented by the sound node fields `minBack`, `minFront`, `maxBack`, and `maxFront`. The resource allocation algorithm uses the `priority` value to rank the sources. The fields `direction` and `location` change orientation and position of the sound node in its local coordinate space. Changes are immediately manifested in the spatialization resource visualizer. If the field values are modified during runtime by the application, the fields in the attached editor are updated. This allows textual sound behavior monitoring of a scene.

Tool Data Flow

Figure 8 shows the data flow between the system components. All tools keep each other up-to-date. Changes from the virtual reality environment propagate to the sound node editor directly. Requests for sound resources are processed by the sound spatialization resource manager and then visualized by the Spatialization Resource Visualizer (seen on the right of Figure 6). The user can select a resource in the visualizer and invoke the sound node editor for the associated sound node. A change here would propagate back to the visualizer via the runtime environment and resource manager. The resource manager updates the panel, so that numeric information about the resource allocation process is available. The panel can also change parameter of the allocation process, which will also propagate through the tools.

Implementation

Our prototype was developed on an SGI Indigo 2 Extreme, connected to an Acoustetron II from Aureal/Crystal River Engineering and Roland Sound Modules. The Open Inventor graphics toolkit [Wernecke94] was expanded for classes (nodes) to support the spatial sound extensions, which were used for our virtual reality applications. Open Inventor is

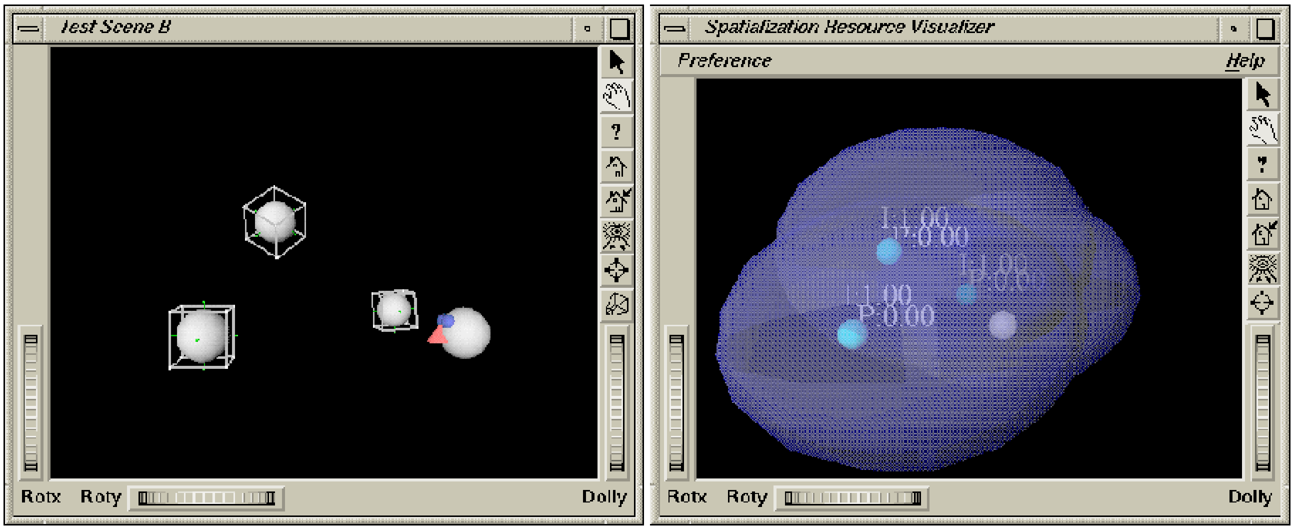

Figure 6: Test scene with sound source visualization

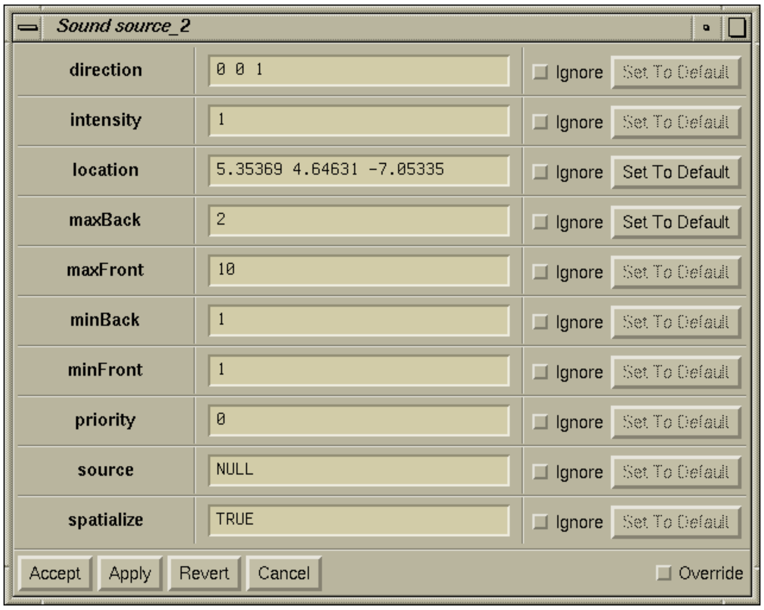

Figure 7: Sound node editor

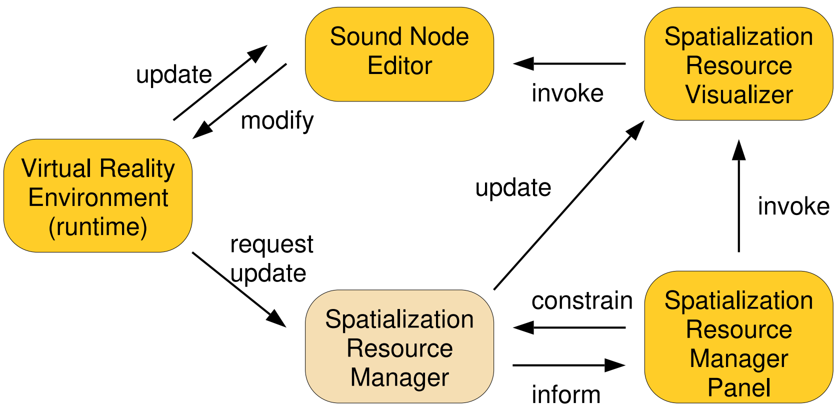

Figure 8: Tool data flow

a superset of the VRML 1.0 standard [Bell *et al.*95], which does not support sound or dynamic behavior of objects. For the sound extensions of Open Inventor, we followed the VRML 2.0 standard [Bell *et al.*96], but added a node for sound sinks. This allows scenes to have multiple sinks and a sink which can be separated from the viewpoint.

CONCLUSION AND FUTURE RESEARCH

My colleagues, students, and I have developed a sound spatialization framework, including visual tools for soundscape manipulation, monitoring, and debugging. The tools enable spatial sound authoring for multimedia content. The system can be easily integrated into an virtual reality authoring system and its principles are widely applicable.

More visual tools could further enhance the spatial sound authoring process. We plan to develop a module for sound source tracing using trajectories and visualizing the convex hull to find out if a source covers a given area for the dynamic behavior. Also a module to visualize statistical data like average orientation, maximum velocity, average velocity, maximum intensity, average intensity, average audible range, etc., would complete the system. For testing and comparing sound spatialization systems, standard tests are required, as are already done for graphics systems.

The described editing facilities cover only directional sound as given by the VRML 2.0 specification. More sophisticated systems would handle sound reflectors, sound occluders [Tsingos-Gascuel97], and reverberation parameters of the space.

ACKNOWLEDGMENTS

The author thanks Michael Cohen and William L. Martens for fruitful discussions. This work is part of the Helical Keyboard project, funded by the Fukushima Prefectural Foundation for the Advancement of Science and Education. The author acknowledges Minefumi Hirose and Taku Suzuki for their help in doing some of the implementation as part of their student and senior projects at the University of Aizu.

REFERENCES

Amano, K., Matsushita, F., Yanagawa, H., Cohen, M., Herder, J., Koba, Y., Tohyama, M. 1996. PSFC: the Pioneer Sound Field Control System at the University of Aizu Multimedia Center. In RO-MAN*'96 - 5th* IEEE *International Workshop on Robot and Human Communication*. IEEE.

Anderson, D. B., Casey, M. A. 1997, The sound dimension, *IEEE Spectrum*, Vol. 34, no 3, pp 46–50.

Begault, D. R. 1994. *3-D Sound for Virtual Reality and Multimedia*. Academic Press. ISBN 0-12-084735-3.

Bell, G., Parisi, A., Pesce, M. 1995. The Virtual Reality Modeling Language, Version 1.0 Specification. `http://vrml.wired.com/vrml.tech/vrml10-3.html`.

Bell, G., Carey, R., Marrin, C. 1996. The Virtual Reality Modeling Language, Version 2.0 Specification, ISO/IEC CD 14772. `http://vag.vrml.org/VRML2.0/FINAL/`.

Blauert, J. 1996. *Spatial Hearing: The Psychophysics of Human Sound Localization.* MIT Press, revised edition. ISBN 0-262-02413-6.

Brown, G., Allard, E. 1997. Sound Bytes: VRML Authoring For Noisy Worlds. In *SIGGRAPH Course Notes*. The Association for Computing Machinery.

Cohen, M. 1993, Throwing, pitching, and catching sound: Audio windowing models and modes, *IJMMS: the Journal of Person-Computer Interaction*, Vol. 39, no 2, pp 269–304. ISSN 0020-7373.

Conner, D. B., Snibbe, S. S., Herndon, K. P., Robbins, D. C., Zeleznik, R. C., Dam, A.van 1992. Three-dimensional widgets. In Zeltzer, D. 1992, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, pages 183–188.

Dalenbäck, B.-I., Kleiner, M., Svensson, P. 1994, A Macroscopic View of Diffuse Reflection, *Journal Audio Engineering Society*, Vol. 42, no 10, pp 793–807.

Darvishi, A. 1997. A Visual User Interface for Creation and Manipulation of Auditory Scenes. In ICAD *'97 — International Conference on Auditory Display*, pages 125–127, Palo Alto, CA; USA.

Herder, J., Cohen, M. 1996. Design of a helical keyboard. In ICAD *'96 — International Conference on Auditory Display*, Palo Alto, CA; USA.

Herder, J., Cohen, M. 1997. Sound Spatialization Resource Management in Virtual Reality Environments. In *ASVA'97 - International Symposium on Simulation, Visualization and Auralization for Acoustic Research and Education*. Tokyo, Japan.

Intel, Inc. 1997. Intel 3D Realistic Sound Experience. White paper. `http://developer.intel.com/ial/rsx/WPAPER/WPAPER.HTM`.

Kaufman, A. 1996, Computer artist's color naming system, *The Visual Computer*, Vol. 2, no 4, pp 255–260.

Martens, W. L. 1997. Sound design for the space opera. Personal communication. `http://www.u-aizu.ac.jp/~wlm/vacuum.html`.

Reilly, A., McGrath, D. 1995. Convolution Processing for Realistic Reverberation. In *98th Convention of the Audio Engineering Society*, Paris.

Savioja, L., Huopaniemi, J., Lokki, T., Väänänen, R. 1997. Virtual Environment Simulation - Advances in the Diva Project. In ICAD *'97 — International Conference on Auditory Display*, pages 43–46, Palo Alto, CA; USA.

Sowizral, H., Rushforth, K., Deering, M., Dale, W., Petersen, D. 1997. Java^{TM} 3D API Specification. Sun Microsystems. `http://www.javasoft.com/products/java-media/3D/forDevelopers/3Dguide/j3dTOC.doc.html`.

Tsingos, N., Gascuel, J.-D. 1997. Soundtracks for computer animation: Sound rendering in dynamic environments with occlusion. In *Graphics Interface '97*, pages 9–16.

Waters, R. C., Barrus, J. W. 1997, The rise of shared virtual environments, *IEEE Spectrum*, Vol. 34, no 3, pp 20–25.

Wenzel, E. M., Stone, P. K., Fisher, S. S., Foster, S. H. 1990. A system for three-dimensional acoustic "visualization" in a virtual environment workstation. In *Proc. First* IEEE *Conf. on Visualization*, pages 329–337, San Francisco.

Wernecke, J. 1994. *The Inventor Mentor*. Addison-Wesley. ISBN 0-201-62495-8.