Chapter 9

Sound Spatialization Authoring

Broader use of virtual reality environments and sophisticated animation 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 handcrafted, because good design tools for spatial sound do not exist. This chapter discusses spatial sound authoring and its applications, like shared virtual reality environments based on VRML. New concepts introduced by this research are an inspector for sound sources, an interactive resource manager, and a visual soundscape manipulator. The visual 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.

More and more applications use virtual reality environments with 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 everyones' 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.*, 1996]). 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. The research described by this thesis has developed widgets, user interface objects with encapsulated geometry and behavior, to control and display properties of soundscapes and sound objects.

Spatial sound Spatial sound [Anderson and Casey, 1997] is modeled by many attributes. Directionalization allows to point towards a sound source. Distance cues are based on delay, reverberation and loudness. The space defines the reverberation and first (and higher) order reflections.

Authoring: Creativity and engineering For spatial sound authoring, two disciplines converse. An artist or content producer with strong emphasis on creativity brings the ideas or better defines what should be done. On the other hand, skills from engineering are needed to define the space and actually produce the effects.

Requirements Requirements for a spatial sound authoring toolset on which this research has focused 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, like 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 for different platforms should not interfere. An example of modeling rendering issues was given in [Brown and Allard, 1997]: the attenuation of the frequency spectrum of a waterfall (part of the "Jungle Island" demonstration) was modeled by using two sound sources with different audible range and frequency band. This made it possible to hear the rumbling of the waterfall in the distance as well as the high frequency components when sufficiently close. The effect was impressive, but what if the sound renderer supports distance frequency attenuation? Would it be better if the API and also the sound authoring tools hid those steps from the user? The same approach can be taken for sound occluders and first-order reflections.

9.1 Previous research

9.1.1 Spatial sound application programmer interfaces

The latest VRML [Bell *et al.*, 1997] 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.*, 1997] specification is in that regard more advanced, supporting a notion of a soundscape, an application area with aural attributes capturing delay times and reflections. The discussed APIs are good for multimedia content, but are not suitable for room acoustics, which are much more complicated and require a more physical approach — specification of the material and transfer functions of any sound object (e.g., a wall) in the simulated space. On the other hand such simulations are not yet done in realtime. Realtime processing becomes possible if the room parameters are processed beforehand.

9.1.2 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 [Cohen, 1993] gives each user a visual, exocentric view on a scene and allows realtime interaction and sound object editing based on direct manipulations and cut & paste metaphors.

In shared virtual environments (e.g., AlphaWorld) [Waters and Barrus, 1997], 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 a case 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.

9.2 Sound spatialization development environment

The developed environment consists of a library to manage all sound processing, a visual soundscape controller (to handle mapping between geometric application space and soundscape), a sound resource allocation monitor, a

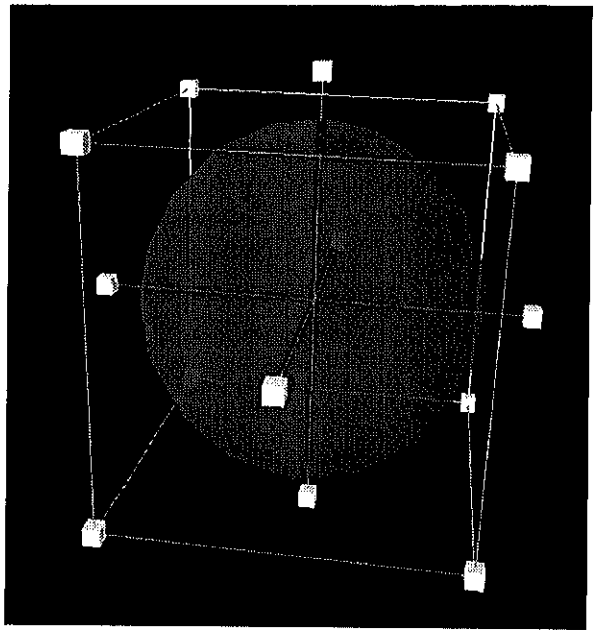

Figure 9.1: Soundscape deformer

soundscape visualizer, and an editor for sound objects. The following subsections introduce the modules and the data-flow.

9.2.1 Soundscape control

During the design of the helical keyboard [Herder and Cohen, 1996], 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, mainly direction. The helix itself became visually a very tall object. 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 $\rightarrow$ soundscape mapping. 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 pot).

The soundscape deformer, shown in Figure 9.1, provides a visual representation of a linear mapping. A sphere in the center represents the case in

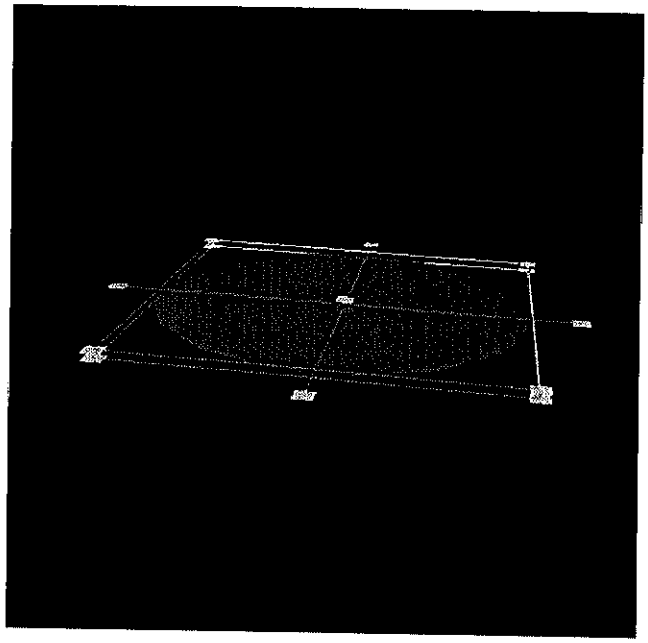

Figure 9.2: Soundscape deformer: flattening

which the scene space is directly mapped to the soundscape. Figure 9.2 shows the soundscape reduced in height, flattening the spatial audio position of all sound objects to a plane. Another example, shown in Figure 9.3, reduces the horizontal dimension, compressing the left↔right attribute. As an extreme example, shown in Figure 9.4, 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.

9.2.2 Portable content: Authoring for different platforms

A commercial application or multimedia product might be required to run on different platforms. Those platforms should be taken into consideration when doing spatial sound authoring.

Output devices Backends vary; a system can use loudspeakers in single, stereo, stereo with crosstalk cancelation, and array [Amano *et al.*, 1998] 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

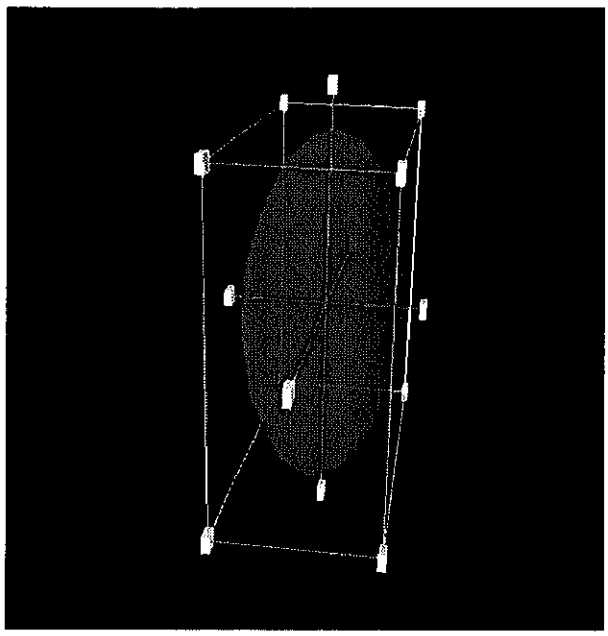

Figure 9.3: Soundscape deformer: narrowing

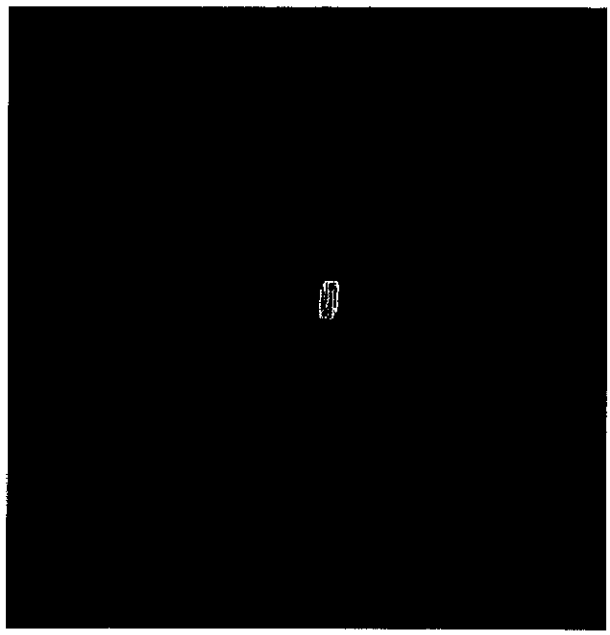

Figure 9.4: Soundscape deformer: extreme diotic case

frequency spectra of the output device needs to be adjusted and considered[1]. 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 remember that, and might choose 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.*, 1990] and software [Intel, Inc., 1997]. In the later case the main CPU load might increase unacceptably if the spatial sound design did not anticipate such problems. 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 the spatialization. A good system maintains balance and optimizes for the user based on psychoacoustic metrics.

9.2.3 Monitoring sound resource allocation

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

We propose to monitor during the authoring process the resource requests. 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 and Cohen, 1997] including a monitor for the requests and allocations. The panel shown in Figure 9.5 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. The audio signals are mixed before the spatialization takes place. Ambient sources do not use spatialization resources, but produce a load for the sound generation.

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

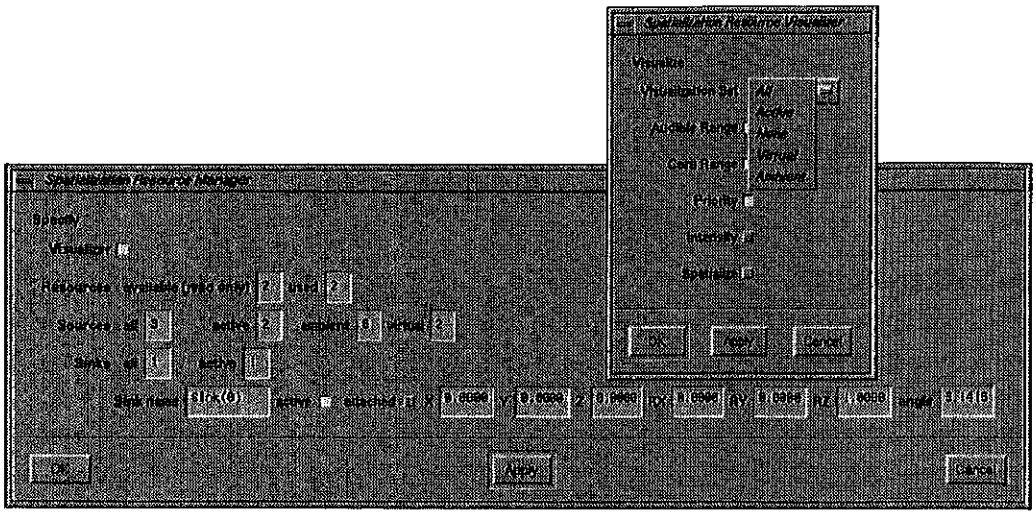

Figure 9.5: Sound spatialization resource manager panel

9.2.4 Simulating resource allocation via resource constraints

How would a certain kind of content be produced on a system with few spatialization channels? The spatialization resource manager resources (i.e., the number of spatialization channels) can be dynamically constrained by interacting with the control panel (as seen in Figure 9.5). 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.

9.2.5 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 9.6 shows on the left side 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 9.5) 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.*, 1997], and its audible range, shown as a translucent ellipsoid repre-

senting the space in which the source is audible. 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 text values facing the user. Different states of a sound node can be conveyed using color codes for the core range.

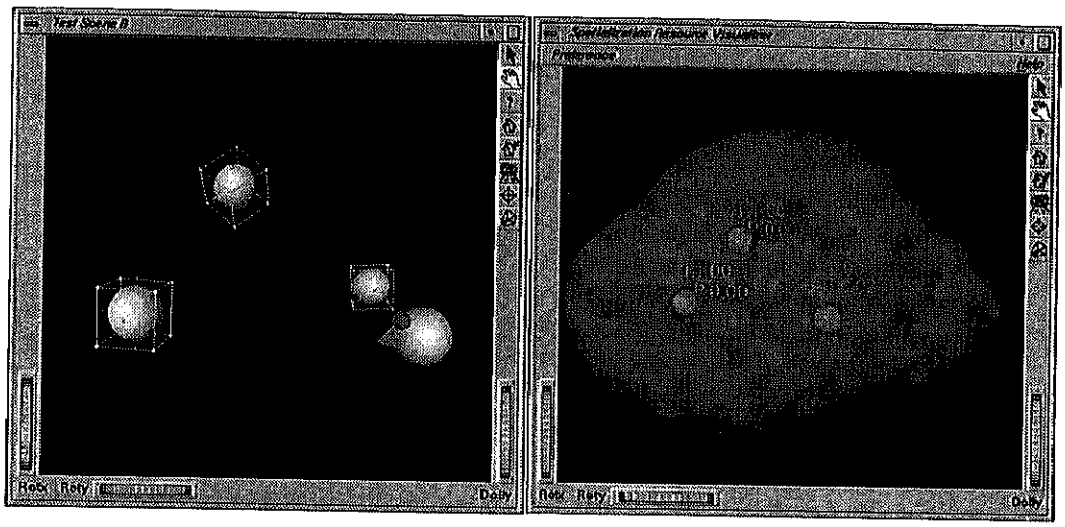

Figure 9.6: Test scene with sound source visualization

9.2.6 Sound source editor

The editor shown in Figure 9.7 allows one to edit and monitor sound nodes [Bell *et al.*, 1997] 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 can be 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.

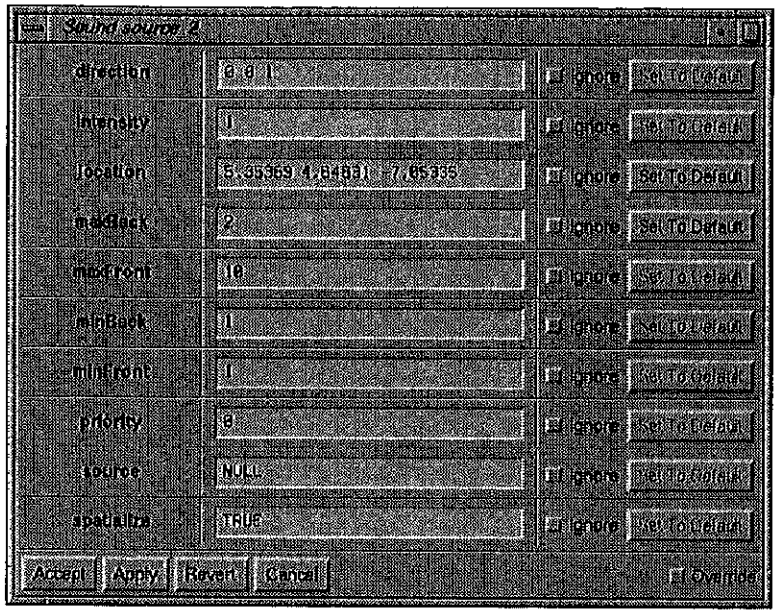

Figure 9.7: Sound node editor

9.2.7 Tool data-flow

Figure 9.8 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 side of Figure 9.6). 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.

9.2.8 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 [Wernecke, 1994] was expanded for classes (nodes) to support the spatial sound extensions, which were used

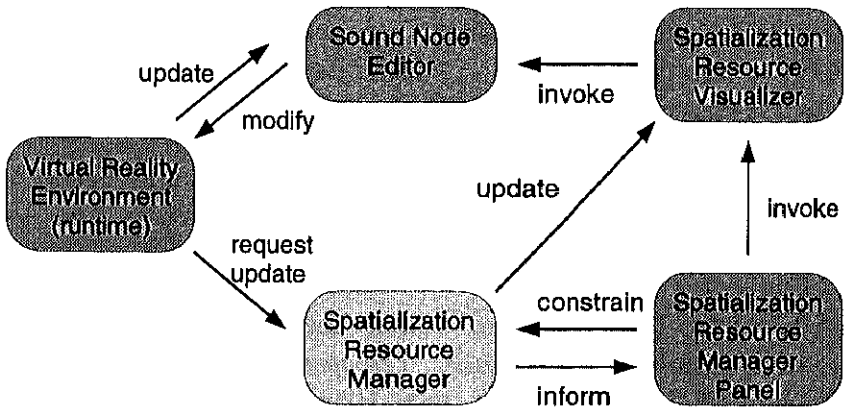

Figure 9.8: Tool data-flow

for our virtual reality applications. Open Inventor is a superset of the VRML 1.0 standard [Bell *et al.*, 1995], which does not support sound or dynamic behavior of objects. For the sound extensions of Open Inventor, we followed the VRML97 standard [Bell *et al.*, 1997], but added a node for sound sinks. This allows to have multiple sinks and a sink, which can be separated from the viewpoint.

Bibliography

[Amano *et al.*, 1998] Katsumi Amano, Fumio Matsushita, Hirofumi Yanagawa, Michael Cohen, Jens Herder, William Martens, Yoshiharu Koba, and Mikio Tohyama. A Virtual Reality Sound System Using Room-Related Transfer Functions Delivered Through a Multispeaker Array: the PSFC at the University of Aizu Multimedia Center. *TVRSJ:* **Trans.** *of the* **V***irtual* **R***eality* **S***ociety of* **J***apan*, 3(1):1–12, March 1998. ISSN 1342-4386.

[Anderson and Casey, 1997] David B. Anderson and Michael A. Casey. The sound dimension. *IEEE Spectrum*, 34(3):46–50, March 1997.

[Bell *et al.*, 1995] Gavin Bell, Anthony Parisi, and Mark Pesce. The Virtual Reality Modeling Language, Version 1.0 Specification, May 1995. `http://www.vrml.org/Specifications/VRML1.0/`.

[Bell *et al.*, 1996] Gavin Bell, Rikk Carey, and Chris Marrin. The Virtual Reality Modeling Language, Version 2.0 Specification, ISO/IEC CD 14772, August 1996. `http://www.vrml.org/VRML2.0.old/`.

[Bell *et al.*, 1997] Gavin Bell, Rikk Carey, and Chris Marrin. ISO/IEC 14772-1:1997: The Virtual Reality Modeling Language (VRML97), 1997. `http://www.vrml.org/Specifications/VRML97/`.

[Brown and Allard, 1997] Geoff Brown and Ed Allard. Sound Bytes: VRML Authoring For Noisy Worlds. In *SIGGRAPH Course Notes*. The Association for Computing Machinery, August 1997.

[Cohen and Koizumi, 1998] Michael Cohen and Nobuo Koizumi. Virtual gain for audio windows. *Presence: Teleoperators and Virtual Environments*, 7(1):53–66, February 1998. ISSN 1054-7460.

[Cohen, 1993] Michael Cohen. Throwing, pitching, and catching sound: Audio windowing models and modes. *IJMMS: the Journal of Person-Computer Interaction*, 39(2):269–304, August 1993. ISSN 0020-7373.

[Herder and Cohen, 1996] Jens Herder and Michael Cohen. Design of a Helical Keyboard. In Steven P. Frysinger and Gregory Kramer, editors, ICAD *'96 — Int. Conf. on Auditory Display*, Palo Alto, CA; USA, November 1996.

[Herder and Cohen, 1997] Jens Herder and Michael Cohen. Sound Spatialization Resource Management in Virtual Reality Environments. In *ASVA '97 — Int. Symp. on Simulation, Visualization and Auralization for Acoustic Research and Education*, pages 407–414, Tokyo, Japan, April 1997. The Acoustical Society of Japan (ASJ).

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

[Sowizral *et al.*, 1997] Henry Sowizral, Kevin Rushforth, Michael Deering, Warren Dale, and Daniel Petersen. Java^{TM} 3D API Specification. Sun Microsystems, August 1997. `http://www.javasoft.com/products/java-media/3D/forDevelopers/3Dguide/j3dTOC.doc.html`.

[Waters and Barrus, 1997] Richard C. Waters and John W. Barrus. The rise of shared virtual environments. *IEEE Spectrum*, 34(3):20–25, March 1997.

[Wenzel *et al.*, 1990] Elizabeth M. Wenzel, Philip K. Stone, Scott S. Fisher, and Scott H. Foster. A system for three-dimensional acoustic "visualization" in a virtual environment workstation. In *Proc. First* IEEE *Conf. on Visualization*, pages 329–337, San Francisco, October 1990.

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