SOUNDSTUDIO4D - A VR INTERFACE FOR GESTURAL COMPOSITION OF SPATIAL SOUNDSCAPES

James Sheridan[1], *Gaurav Sood*[1], *Thomas Jacob*[1,2], *Henry Gardner*[1], *and Stephen Barrass*[2]

[1] Departments of Computer Science and Engineering, FEIT, Australian National University, Canberra, ACT 0200, Australia
Contact: Henry.Gardner@anu.edu.au
[2]ICT Centre, CSIRO, Canberra, Australia
Stephen.Barrass@csiro.au

ABSTRACT

We describe a software system which enables computer-generated soundscapes to be synthesised, spatialised and edited using a gestural interface. Iterative design and testing of the software interface has taken place in a walk-in, immersive, virtual-reality theatre. Sound spatialisation has been implemented for an 8-speaker array using a Vector Base Amplitude Planning algorithm. The software has been written in Java, JSyn and Java3D with native method calls to sound-cards and sensors.

1. INTRODUCTION

The use of a gestural interface to compose and spatialise music is as old as conducting: Although the range of parameters which can be varied during a live, orchestral performance is small (compared with what can be played with on a computer) the meter, expressivity and relative dynamics of the music is critically dependent on the body-language and baton of a master conductor.

Since the Theramin, there has also been a long history of building gestural interfaces for the real-time composition of electronic and computer music in the context of its performance. But the off-line composition of computer music is still traditionally associated with interfaces which are a long step away from free-flowing gestures. This is particularly the case with three-dimensional (3D) spatialised music where the state of the art is driven by the 2D desktop metaphor of the personal computer. For example, systems such as InMotion [1] allow the careful spatialisation of sound tracks but only by adjusting projections of the 3D paths and only by decoupling the time dimension from the path itself.

Virtual reality (VR) systems hold the promise of being able to link full-body gestural input with 3D computer sound generation (as well as 3D computer graphics). This was the motivation behind the development of a prototype 3D spatialising system [2] for an immersive VR theatre at the Australian National University. This prototype system was limited in that 3D audio was not really available, there was no sound synthesis and there was no systematic consideration of the nature of the interface. All of these considerations have led to the development of the software system which is described in this paper.

After an overview of the software system architecture, in Section 2, we describe the three main subsystems: the sound spatialiser (Section 3), the sound synthesiser (Section 4), and the sound driver (Section 5). We discuss the interface, the software architecture and implementation, and the design process including the results of a human computer interface (HCI) study of one part of the software. The project has solved many technical problems and has resulted in a software system which is ready to use in an experimental setting. Its status is summarised in Section 6.

2. SYSTEM ARCHITECTURE

A schematic view of the architecture of our SoundStudio4D system is shown in Fig. 1. The software contains three major modules: PathDrawer4D handles the sound spatialisation, GroovyTubes provides a prototype interface for sound synthesis and BeeHive drives an 8-speaker array using Vector Base Amplitude Panning. The top level of the software is written in Java$^{\mathrm{TM}}$ and uses classes from the standard Java3D$^{\mathrm{TM}}$ package to create and manipulate a scene-graph made up of graphical and audio nodes which represents the composition. It also uses the standard JSyn$^{\mathrm{TM}}$ package to modify the quality of sounds. Using packages from the standard Java platform eliminates the need to learn new languages or become familiar with new protocols to drive either the audio interface or the VR theatre itself. Thus, the system allows a user to program these devices using only minor extensions to those needed to create other Java3D sound applications. This should mean that SoundStudio4D is easy to learn as well as being a robust and maintainable programming environment.

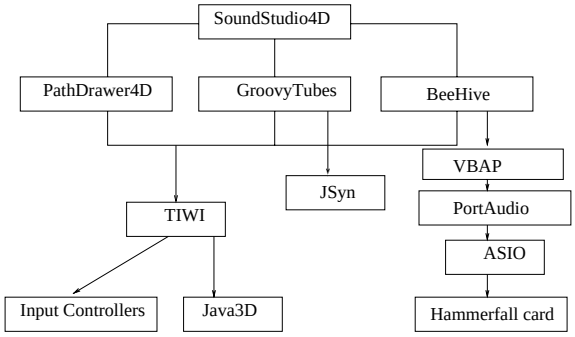

Figure 1: *Overview of the software architecture for SoundStudio4D*

2.1. The Wedge VR theatre

The Wedge is a PC-based, VR theatre which was originally developed at the Australian National University in 1998 [3] as a low-cost alternative to the CAVE [4]. It consists of two back-projected screens arranged at 90 degrees to each other to form a panoramic, body-sized, immersive, space around the participants. Participants see 3D stereo-graphics through LCD shutter glasses that have cross-polarizing filters synchronised with the frame updates for each eye. Because participants are also able to see themselves, each other and the physical environment, the theatre enables collaboration even though only one participant will see the geometrically correct 3D image projections.

The viewpoint of the main user is head-tracked by a Logitech^{TM} ultrasonic tracking unit which also tracks the spatial position and orientation of a six degree of freedom (DOF) mouse. The main input devices are the mouse (4 buttons as well as its 6DOF position and rotation coordinates) and a small keyboard extender. Most importantly for our application, the Wedge has an array of eight speakers, positioned on a (45-degree) rotated cuboid, for producing sounds that can move around the user. The graphical environment of the Wedge together with its input devices can be programmed using classes from the Tracked Interactive Wedge Interface (TIWI) [5] which creates a customised Java3D "ConfiguredUniverse". Details of the audio system are covered in Section 5.

3. SOUND SPATIALISATION: "PATHDRAWER4D"

Suppose that a composer makes a bold, sweeping gesture whilst standing inside a VR theatre running the PathDrawer4D module of our software. How is this gesture to be interpreted? Presumably the gesture will affect the spatial displacement of part of the soundscape in some way. In a "creation" phase, the gesture could correspond to the drawing of a path which will become a sound trajectory. In an "editing" phase the gesture might correspond to grabbing part of the soundscape and displacing it relative to others in the scene. How might the gesture correspond with time (the fourth of our dimensions)? There could be a real-time editing facility which would be something similar to a conductor manipulating a performance. There might be a need to specify time separately with reference to a time-bar or relative to some other parts of the soundscape. How does a user change modes or know what mode the software is presently in?

The sorts of considerations in the previous paragraph show that it is possible to make some headway in the design of a gestural interface for sound spatialisation using the analogies of drawing and conducting. Even so, the range of possibilities is much greater than the interaction taxonomies which have been developed for basic VR techniques of travel, selection and manipulation [6].

3.1. Visual representation of paths in four dimensions

In most current audio applications the time dimension has the focus, with all other things (such as spatialisation or audio effects) relative to this. Even in displays which have a prominent graphical window showing the sound path, it is the time dimension (often displayed in a smaller, bottom window) that is used to explicitly schedule all of the sounds.

Displaying multiple windows in a virtual world is not desirable as it takes away from the immersive nature of the environment. For this reason the space and time dimensions will sometimes need be displayed together. So a path needs to represent not just a series of coordinates (the path geometry) but also the time dimension.

There are many ways to represent the both dimensions at once: Markers could be placed every fixed amount of time. Numeric labels could be used to display the time at particular places on a path. Colour could be used to represent the time dimension. Each of these has their disadvantages: Both markers and numbers can lead to visual clutter. Colours do not provide enough detail to synchronise sounds in a soundscape. When input to an iterative design process, these considerations led us to adopt a numbered view of the time dimensions for editing (to increase resolution) but a coloured view for drawing (to ease visual congestion).

There are also many sorts of desirable sound-path geometries. For example: Straight paths with a constant-velocity sound; Straight paths with a variable-velocity sound; Curved paths with a constant-velocity sound; Curved paths with a variable-velocity sound. Other qualities of the sound such as its volume and pitch could vary along the path. Our consideration of these requirements led us to develop a taxonomy of interaction for the PathDrawer4D application which is described in the following sub-section. We decided to allow users to specify curved and straight and piecewise-curved and piecewise-straight paths. Switching between different path styles turned out to be an important part of our

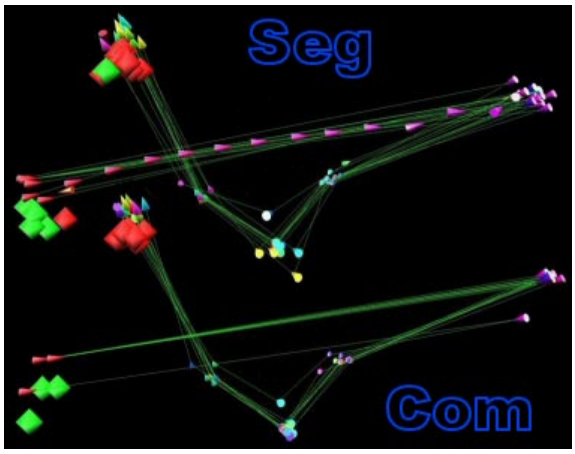

Figure 2: *Overlayed paths from the HCI path-tracing experiment for a master path made up of straight line segments. Results for two tools are shown.*

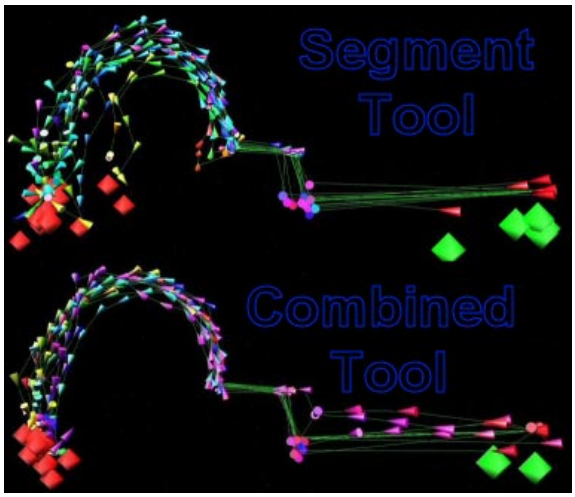

Figure 3: *Overlayed paths from the HCI path-tracing experiment for a master path made up of straight and curved segments. Results for two tools are shown.*

taxonomy and was the subject of an HCI study described below. Figures 2 and 3 show a number of overlayed paths drawn by participants in the HCI study. Although these are congested, they give an idea of the types of paths which can be drawn using the software.

An additional consideration was the mapping between the visualisation and the acoustic playback space. Two cases need to be considered: the first is one where a soundscape is designed in the Wedge theatre space for playback in that space. The second is where the playback space is to be in an auditorium which is different from the Wedge space. In the first case, a naive first preference is to have a one-to-one mapping: holding out your arm and clicking a button would position a sound or sound path exactly at the position of the button click. This mapping has some major practical disadvantages: It makes it difficult to position distant sound paths and it is difficult to design paths which sweep behind the user's head position. For this reason, we decided to build a system whose dominant view was that of an avatar displaced from the user's position (as seen in Fig. 4). This displacement is adjustable and it can be alternated with an "avatar's" view with the avatar coinciding with the position of the lead participant.

3.2. Taxonomy

The following steps were selected as the root levels of the taxonomy of interaction for PathDrawer4D:

Feedback: This was the set of interface elements that could be provided at all times (such as the current task, the reliability of the tracker signal or the playback time).

Select Task: Since there was more than one task that could be done in any sequence, users need to specify what it was they want to do - to, for example, load a path from a menu or click on a path to start editing it.

Create Path: The main function of the interface was to allow the spatialisation data to be specified.

Specify Time Dimension: Whilst the time dimension could be specified during the creation of the path, it could also be done by itself and was given its own "level" within the taxonomy.

Edit Path: After a path has been drawn or loaded it can be edited by moving or deleting points to fix mistakes or adjusting its sound.

Edit Time Dimension: It is necessary to vary timing information independently of the spatial coordinates of a path.

Sound Placement: It was desirable to have multiple sounds on the one path. The ability to change the sound on a path was also desirable to save redrawing the path if the user wanted to try a different sound.

To test the completeness of the taxonomy two commercial, desktop interfaces for sound spatialisation were fitted to it. A cognitive walkthrough of the SoundScene proof of concept prototype [2] was also performed to identify problems in the software. Many issues and problems were identified during this phase such as the difficult nature of selecting path points, inappropriate colour schemes, lack of functionality and so on. Our draft taxonomy was used to find solutions to these problems.

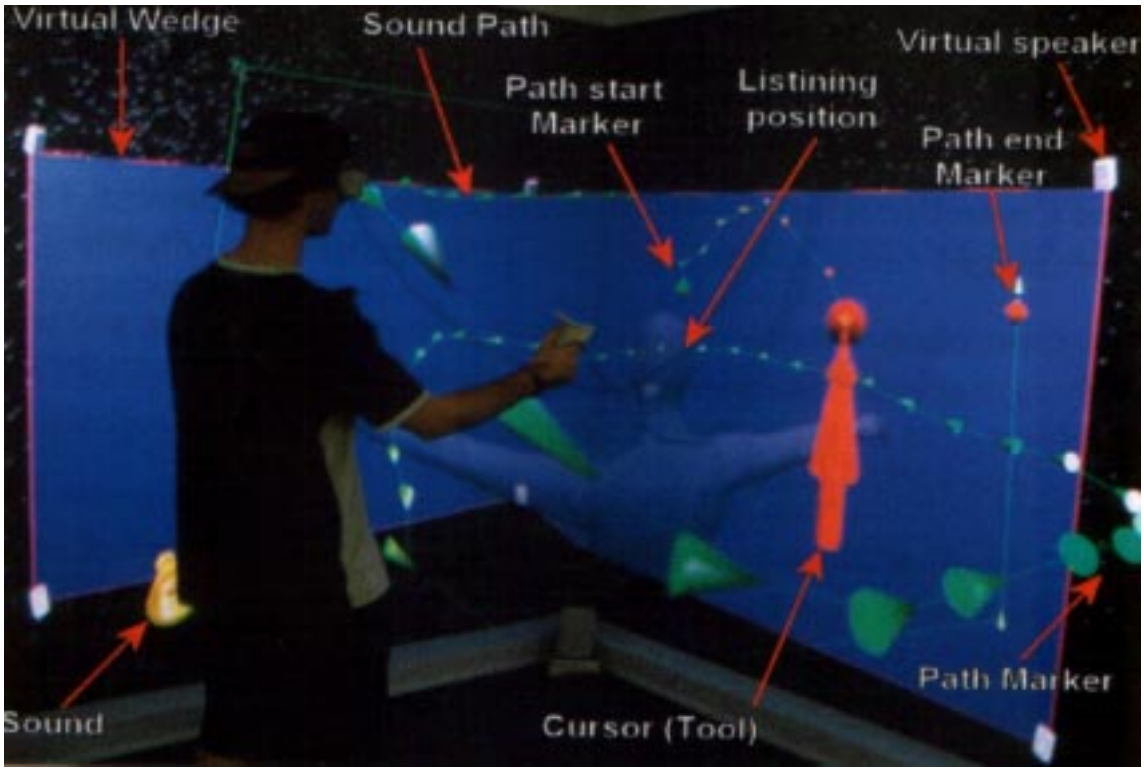

Figure 4: *The PathDrawer4D application in action.*

3.3. HCI studies

The aims of this testing were: to compare two different paths through the taxonomy tree which provided different ways to do the same task; to test whether people with little or no VE experience could adequately perform the tasks needed for 4D sound spatialisation; and to gain insight into general usability issues in the Wedge.

A number of standard HCI techniques were used for the testing. A usability study used a between-subjects design on two alternative tools and a within-subjects design on a number of candidate paths. Subjects were asked to trace out a number of master-paths using one of two drawing tools. The "segment tool" used two different 6DOF mouse buttons to control each of its two modes (for straight and curved segments) with a consistent mapping of the button clicks for each mode. The "combined tool" used only one mouse button for both modes with the modes being detected by the way in which the mouse was used (clicking versus clicking and holding). Overlayed results for two paths are shown in Figs. 2 and 3.

During these tests a "think aloud" method was used to try to understand the users' conceptual model. Previous research into multimodal interaction has found that users often express commands by combining voice and hand gestures when dealing with spatial tasks [7]. For this reason the use of "thinking aloud" was assumed to have minimal impact upon user performance.

Apart from these studies, other HCI methods were used for things like questionnaire design, test execution and so on.

Usability testing employed 10 participants for each of two tools for tracing 7 paths. Results showed some slight preference (statistically significant on some paths but sensitive to outliers) for the "combined tool" which we interpret as possibly meaning that the cognitive load of switching between buttons on our controller is higher than using a mixture of clicking and click-dragging with one button. The most important results of the testing were the discovery of systematic usability issues with the theatre and with one of the tools. Several of these problems have since been rectified, but, although the virtual environment still looks promising for sound-scene sculpturing, it should be said that immersive theatres of the Wedge/CAVE type do not appear to be suitable for "precision" drawing applications.

4. SOUND SYNTHESIS: "GROOVYTUBES"

The next development for SoundStudio4D was to introduce special effects that can be applied to sounds as they move along sound paths. Typical effects include echo, flanging, reverberations and wah-wahs. These effects usually vary

sound characteristics such as the rate, filter bandwidth and pitch to create the special effect. JSyn is a Java sound synthesis and signal processing toolkit that can be used to create sound effects in real time. We integrated JSyn into SoundStudio4D in the Wedge by implementing a real-time switching mechanism to change the effect as sounds moved into a particular region of a sound-path.

The design of special effects on sound paths requires the development of an interface for specifying multiple effect parameters in space and time. This is a more difficult problem than the sound spatialisation interface because of the lack of clear metaphors: presumably gesturing in an upwards direction would have some effect on sound quality (louder or higher?) depending on the mode of operation. But the exact form of interaction, and the nature of the visual feedback, is by no means clear.

Rather than developing a systematic taxonomy, our approach to this part of the interface has been to choose one promising candidate and to subject it to a process of iterative design. Our candidate is the "Groovy Tube", which consists of three parts: the effect algorithm and its parameters; the interaction with the effect parameters; and a visualisation of the effect parameters[8].

Special-effects algorithms may alter single or multiple parameters depending on the effect. We were interested in developing an interface for designing complex sounds and, therefore, wanted to modify more than one parameter at the same time. Our initial experiments with JSyn showed we could modify 3 parameters of a filter algorithm at interactive rates in the Wedge. Our prototype used the following three parameters:

1. High pass filter cut-off frequency
2. Filter resonance (Q)
3. Playback rate of the sample

The movements of the tracked 6DOF mouse are recorded in a data structure for replay and further editing. We experimented with two different approaches for storage and playback of the effects parameters. The first approach was to store samples in the EnvelopePlayer object within the JSyn API. This structure requires samples to be stored as alternate values of time and parameter and we found accessing and writing to such an object is inherently complicated for multi-parameter effects. This led us to try an alternative approach where envelope data is stored in a two dimensional array with effect values stored at the corresponding time index for the path. During playback, the system accesses the data and sets the values of the sound attributes directly. The complexity of searching and rewriting the parameter value at a certain point in time is minimised.

The visualisation of the effect parameters as they change along the sound-path provides a way to get an overview and to edit the effect. We have designed and compared three possible visualisations of the parameters along a sound-path: spheres, fins and cubic prisms.

Of these three, the cubic prisms visualisation addresses the problems of high polygon count and visibility from different directions raised by the other visualisations. A cubic prism is a 3D cubic structure that maps the parameter values at each point to the width of the prism. The prism is constructed from a triangular strip array. The geometry is built from points along the path. The first step is to calculate the tangent to the path at each sample point. The next step is to calculate orthonormal vectors perpendicular to the tangent. The representation of three filter envelopes is possible on the same sound path but leads to overlapping. We are able to differentiate between the overlapping envelopes using separate colours for each envelope. The array of effect values at each sample point is mapped to the points at the next sample to create the Triangle Strip Array.

An example GroovyTube is shown in Fig. 5. The effects parameters were specified interactively using the x, y and z movements of the 6DOF mouse. The effects parameters change the sound as it moves along the path.

Once a Groovy Tube has been specified it can be edited using the visualisation. The effect parameter for the location closest to the mouse pointer can be resized using the relative movement of the mouse pointer. The magnitude of the parameter is scaled by a multiple of the relative motion of the mouse. Interpolation is used to create smooth transitions from neighbouring points to the new value.

5. THE SOUND-CARD DRIVER: "BEEHIVE"

BeeHive is an audio system that maps sounds from the Java scenegraph to the 8-speaker array in the Wedge. It was designed primarily with the goal of allowing users to be able to manipulate one or more sounds in real-time and to be able to hear the results of the manipulation in 3D space. Beehive has an advantage over other systems in that it integrates both audio and graphical systems and allows them to be created inside the one scenegraph.

The system uses a Java3D AudioDevice to provide an interface for the 8-speakers in the Wedge. The AudioDevice is a custom implementation of the Java3D AudioDevice3D class, which extends the functionality from two channels to eight.

The mapping from the scenegraph to the speaker array is done using the Vector Base Amplitude Panning (VBAP) algorithm. VBAP is a technique for positioning virtual sound sources to multiple loudspeakers developed by Ville Pulkki [9]. The number of loudspeakers can be varying and they can be placed arbitrarily in 2D or 3D positions, ideally all equidistant from the centre of the 2D or 3D array. VBAP works on the principle that any virtual source in space can

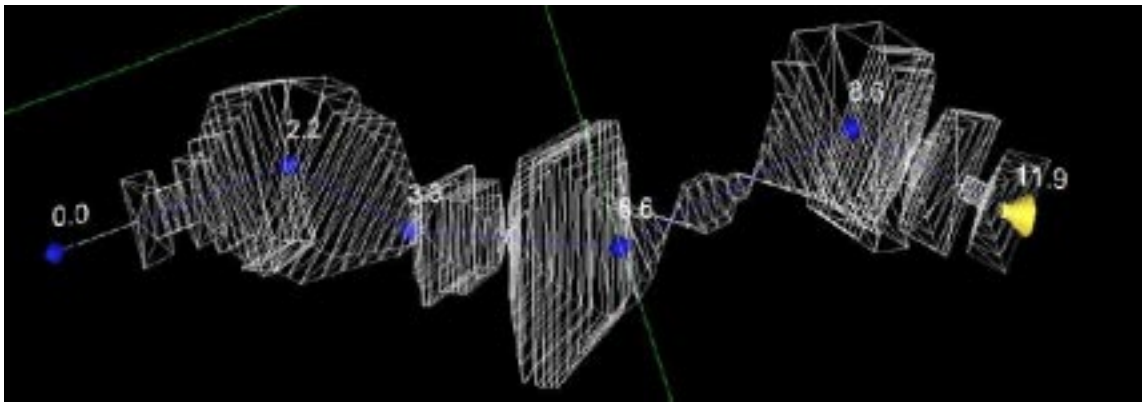

Figure 5: *Example of a prototype groovy tube.*

be represented by sources from one, two or three speakers in an array.

The underlying system for BeeHive is a C++ implementation that interfaces with the Java3D device class using the Java Native Interface (JNI). The implementation is a PortAudio program developed on top of Steinberg's Audio Streaming Input Output (ASIO) API [10], and serves as a mixer to allow sounds to be streamed to the sound card. PortAudio [11] is a powerful library that can interface with lower levels APIs such as ASIO and DirectSound. It allows the porting of applications between different platforms, and essentially eliminates the need to develop host-specific applications.

BeeHive provides an advantage over domestic and commercial sound systems, by creating truly 3D spatial sound as opposed to a surround "sensation". In addition to the impressions of left-right and front-back, the system also captures the up-down experience, which creates an extra degree of immersion.

6. SUMMARY AND CONCLUSIONS

The SoundStudio4D is an immersive VR interface for composing spatial soundscenes using direct spatial gestures. It has been developed to explore the idea that an immersive direct manipulation interface could have advantages over existing desktop interfaces for designing spatial sound paths. The SoundStudio4D has three main parts:

1. the PathDrawer4D interface for drawing sound paths in space and time
2. the GroovyTubes visualisation of special effects parameters in space and time
3. the Beehive Java3D AudioDevice that maps sounds from a Java scenegraph to the Wedge speaker array using the VBAP algorithm

PathDrawer4D enables the direct manipulation of spatial sound using spatial gestures, and provides a visualisation that allows the composition of more complex sound scenes than is possible with existing, 2D, desktop-based, sound tools. However, in evaluations of two different PathDrawer4D tools we found evidence that, although the interface is direct, it is not very precise. GroovyTubes provides a framework for designing even more complex sound scenes with multi-parameter special effects that vary in space and time. This required the exploration of 3 different visualisation schemes, and investigation of interaction techniques. Beehive has the advantage over other systems that it enables both the graphics and audio to be controlled by the same scenegraph, which reduces programming complexity, message passing, and synchronisation problems. The Wedge speaker array is a (45-degree) rotated cuboid that has the advantage that the important left-right and front-back positions are more strongly focussed by the arrangement of the speakers.

We are currently working on the integration of JSyn with Java3D so that all programming of visual, audio and synthesised elements can be done in one language. From the results of the evaluation of the PathDrawer4D we have begun thinking about alternative gestural interfaces that allow more expressive control with less precise specification. The grand vision is to build an environment for composing new kinds of soundscenes and music that are not possible with existing interfaces.

7. REFERENCES

[1] Human Machine Interfaces Inc., "InMotion 3D Audio Producer 1.0," 2004, `http://www.sonicspot.com/inmotion3daudio/inmotion3daudio.html`. Last accessed 29 January 2004.

[2] Rod Harris, "Creating sound scenes in a virtual environment," Tech. Rep. 1, Australian National University, Canberra, Australia 0200, 2001, Internal report - available on request.

[3] H. Gardner and R. Boswell, "The wedge virtual reality theatre," in *Proceedings of the Apple University Consortium Conference,Townsville, Qld, Australia, Sept. 23-26, 2001*. Apple University Consortium, Sept 2001, pp. 2.1–2.6, ISBN 0-947209-33-6. http://auc.uow.edu.au/.

[4] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. De-Fanti, "Surround-screen projection-based virtual reality: The design and implementation of the cave," in *Proceedings of SIGGRAPH 93, Anaheim, California, USA, 1-6 August 1993*. Association of Computing Machinery, New York, August 1993, pp. 135–142.

[5] David Walsh, "The Wedge and TIWI Guide," 2004, `http://ephebe.anu.edu.au/tiwi/tiwiguide/index.html`. Last accessed 30 January 2004.

[6] Doug A. Bowman and Larry F. Hodges, "Formalizing the design, evaluation and application of interaction techniques for immersive virtual environments," *Journal of Visual Languages and Computing*, vol. 10, pp. 37–53, 1999.

[7] Sharon Oviatt, "Ten myths of multimodal interaction," in *Communications of the ACM*, pp. 74–81. ACM Press, 1999, Vol. 42, No. 11, November.

[8] Gaurav Sood and Stephen Barrass, "Groovy tubes: an interface for designing sound effects in space and time," in *New Directions in Interaction: Information environments, media and technology, Conference Proceedings 2003 OZCHI Conference.* University of Queensland, November 2003, pp. 246–249.

[9] Ville Pulkki, "Spatial sound generation and perception by amplitude panning techniques," Tech. Rep. ISBN 951-22-5532-4, Department of Electrical and Communications Engineering, Helsinki University of Technology, P.O.Box 1000 FIN-02015 HUT FINLAND, 2001, `http://lib.hut.fi/Diss/2001/isbn9512255324/isbn9512255324.pdf`. Last accessed 30 January 2004.

[10] Steinberg Soft und Hardware GmbH, "ASIO SDK Download," 2004, `http://www.steinberg.net/en/support/3rdparty/asio_sdk/index.php?sid=0`. Last accessed 29 January 2004.

[11] R. Bencina and P. Burk, "PortAudio - An Open Source Cross Platform Audio API," 2004, `http://www.portaudio.com`. Last accessed 30 January 2004.