A Java Virtual Sound Environment

Andrew Eales

Wellington Institute of Technology
Petone, New Zealand

andrew.eales@weltec.ac.nz

Proceedings of the 15th Annual NACCQ, Hamilton New Zealand July, 2002 www.naccq.ac.nz

ABSTRACT

Multiple loudspeakers can be configured to create a three-dimensional virtual sound space that allows a sound to be placed at any point within the virtual sound space. The accurate placement of the position of a sound plays an important role in virtual-reality applications and allows composers to realise compositions that exploit sonic spatialization. Software that represents and controls sound placement within a virtual sound space typically uses expensive commercial or dedicated audio hardware to mix and route audio signals. This paper discusses the perception of sound spaces, the acoustical properties of spatial sound, and describes a Java-based virtual sound environment. A teaching laboratory equipped with multimedia workstations provides the required hardware. A single master workstation controls the mixing and routing of audio by controlling the loudspeakers on multiple slave workstations. The master machine controls the slave loudspeakers using the networking capabilities of Java. Synchronization between the workstations is achieved using MIDI Time Code.

1. INTRODUCTION

A virtual sound space created by multiple loudspeakers allows a sound to be positioned at any point within the virtual space. Listeners perceive a virtual point of origin created by the relative intensities of adjacent loudspeakers. The illusion of the position of a sound source can be effectively coupled with visual media, and plays an important role in virtual reality systems. A spatial sound system also provides a development environment for composers to exploit spatial effects. Research using spatial sound systems has been dependent on expensive digital hardware or dedicated systems (Eales, 1994) that process, mix and route audio signals. Increases in processor speed and network bandwidth provide the possibility of distributed environments that are suitable for spatial audio applications. This paper proposes the use of an existing network, such as a networked computer-teaching laboratory to create a virtual sound space. Software is written in the Java programming language which supports distributed applications and three-dimensional graphics. Third-party Java tools that implement MIDI capabilities are also freely available.

2. MULTIPLE NETWORKED LOUDSPEAKERS

The topology of the system resembles a parallel processing environment where a master process delegates tasks to slave processes. One machine acts as the master and synchronizes the audio performance of multiple slave machines.

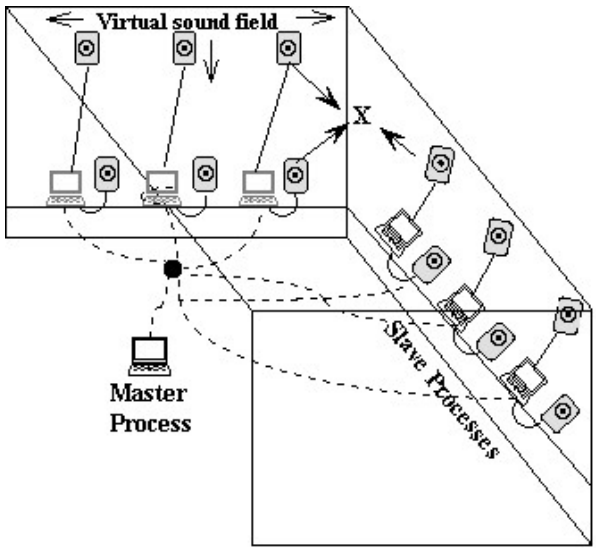

Figure 1. A virtual sound space created by multiple loudspeakers

The master process also interacts with the user interface and communicates the desired position in the virtual sound space to the relevant slave processes. Figure one illustrates a virtual sound source x, created by calculating the intensities of the two or three nearest loudspeakers to the position of the desired virtual sound. The extent of the sound field is determined by the loudspeaker positions and exists behind the loudspeakers. A network for creating spatial sound in real-time must address two problems. Firstly, the synchronized playback of digital audio across the network, and secondly, the routing of audio to the required loudspeakers.

2.1 AUDIO SYNCHRONIZATION

MIDI (Musical Instrument Digital Interface) is a well-documented communications protocol that controls the operation of audio hardware. A MIDI input and a MIDI output port are found on most computer soundcards. The synchronization capabilities of MIDI allow each slave machine to play digital audio that is synchronized with all other slave machines. Sequencer software that traditionally allows multiple MIDI tracks to be synchronously performed has recently allowed digital audio tracks to be used alongside MIDI tracks. Audio tracks are internally synchronized with MIDI tracks using a timer on the soundcard. A variety of MIDI synchronization protocols exist. Lehrman and Tully (1993) discuss the development of MIDI synchronization. MIDI Time Code (MTC) is the most commonly used synchronization scheme. MTC is a digital representation of the analogue SMPTE (Society of Motion Picture and Television Engineers) timing signals used to synchronize MIDI tracks to video frames. The frequency of MTC is always expressed in terms of frames per second. Each frame generates four quarter-frame messages. At a commonly used frame-rate of thirty frames per second, MIDI will generate 120 quarter-frame timing pulses every second. The master process broadcasts these timing pulses to slave processes so that playback occurs in lockstep with the master.

2.2 AUDIO ROUTING

Slave machines are responsible for calculating and adjusting the intensities of their own loudspeakers. Delegating this responsibility to slaves ensures that processing loads are balanced across the network. Calculation of loudspeaker intensities requires a local representation of the loudspeaker topology. Processing requirements are illustrated by data-flow diagrams used in the real-time development methodology advocated by Ward and Mellor (1985). A sequencer running on the master machine transmits MIDI Time Code via the MIDI output port on the soundcard. A Java application intercepts the MIDI timing data and determines the current position within the sound field from the user interface. This data is then routed across the network to slave machines.

Data packets are transmitted at the same rate as the generated time code i.e.120 times per second. Microsoft Windows environments only allow a single application to have access to the MIDI ports. Multi-client drivers or software that implements virtual MIDI ports attached to a single physical port are required to intercept the sequencer output. Figure three illustrates the process that occurs when slave machines receive a data packet.

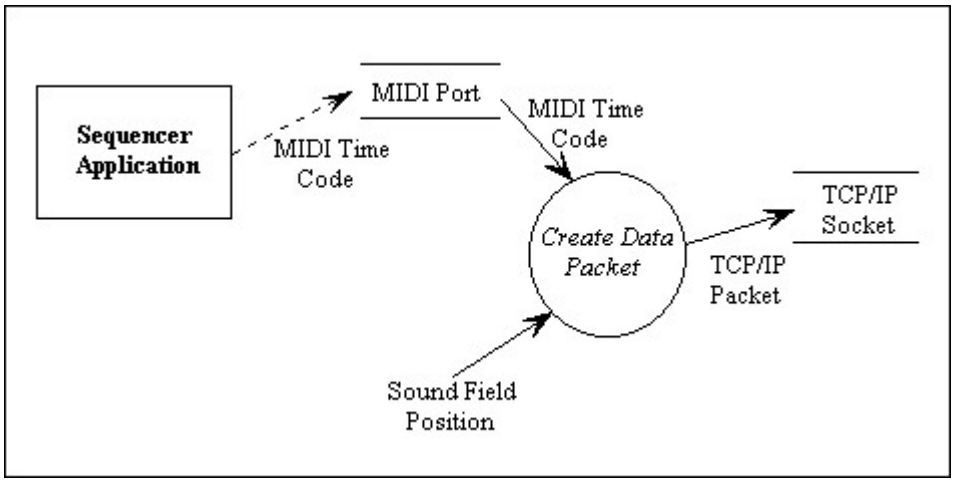

Figure 2. Master process data flow diagram

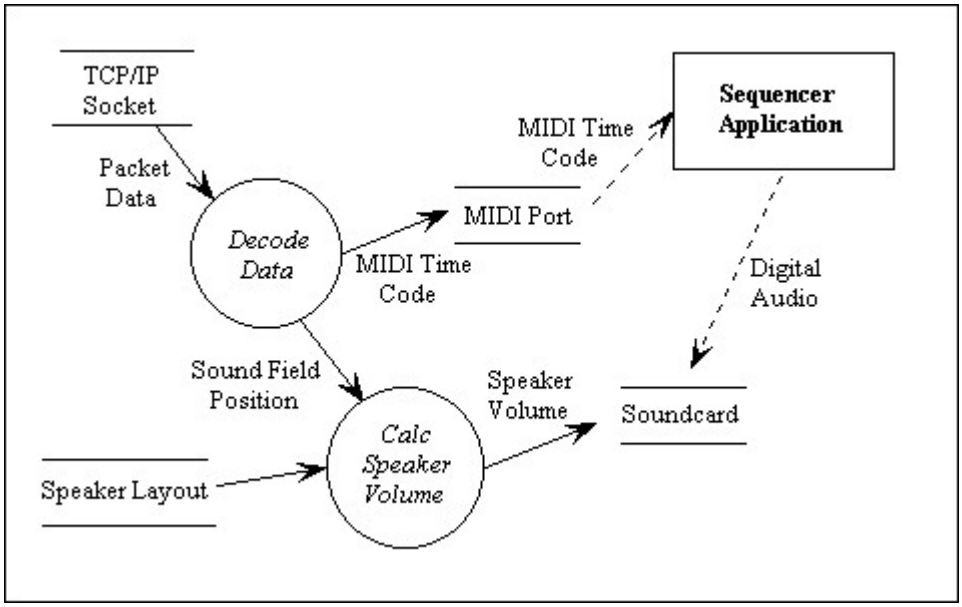

Figure 3. Slave process data flow diagram

A Java application uses the position within the sound field to calculate the appropriate settings for the loudspeakers and passes the MIDI timing signal via a MIDI port to the sequencer application performing the audio. Routing decisions are not required by the system as audio is performed continuously using all speakers. The illusion of routing is given by simply adjusting the output levels of different speakers.

3. MIDI SOFTWARE TOOLS

A variety of commercial and freeware MIDI performance and development tools are available. Software such as Sonar from Cakewalk Music Systems supports different types of MIDI synchronization and also allows synchronization to other types of digital media. Audio can be combined with digital slide shows and video. MidiShare (GRAME, 2002) developed by the French “Centre

National de Création Musicale" (GRAME) is a free MIDI operating system and development environment that supports a variety of programming languages, including Java. Master and slave processes written in Java can use MidiShare code to intercept and process the MIDI data stream as described in this paper. It is also possible to use the native Java support for MIDI included in versions three and four of the Java language specification. However, as an operating system capable of creating multiple MIDI processes, the MidiShare system offers greater flexibility, and is more likely to support future enhancements to the system's capabilities.

4. SPATIAL ACOUSTICS AND PSYCHOACOUSTICS

Calculation of the required speaker intensities is influenced by the physics of sound as well as the perception of sound. Subjective loudness is proportional to the cube root of the intensity of the sound (Stevens, 1955). Linear movement of a virtual sound source is achieved by a non-linear adjustment of speaker intensity that must take the angle from the desired virtual sound source to the speaker into account (Moore, 1989). The different physical, physiological and psychoacoustical factors that influence auditory perception are well documented (Blauert, 1983). Interaural timing differences created by sound reaching each ear at different times and the masking effects produced by the head obstructing sound waves are subtle and require the exact position of the listener to be known. Research has indicated that reverberation (Kendall, *et al.* 1989) and pitch shifts also influence spatial perception. Changes in pitch suggest a Doppler shift and thus provide the illusion of a moving sound source.

The use of digital audio allows digital processing algorithms that implement spatial auditory cues (such as reverberation and Doppler shifts) to be applied to the audio stream. Digital signal processing algorithms such as those discussed by Lindley (2000) can modify the audio output stream in real-time. Current processor speeds allow real-time audio processing but may introduce an unacceptable latency should the processing time exceed the available time between successive MIDI timing signals. Using a network connection to only transmit virtual sound field positions from the user interface may solve latency problems. MIDI timing signals can be directly routed by connecting the MIDI output of the master machine soundcard to the MIDI input ports of slave machines. This solution requires that the MIDI signal be split and may result in signal degradation. The MIDI specification recommends that the length of transmission cables should not exceed fifteen metres (MIDI Manufacturers Association, 1996).

5. USER INTERFACE DESIGN

An accurate graphical representation of a virtual sound space requires a dynamic three-dimensional representation that continually updates the user's perspective. A three-dimensional model with an additional z-plane representation such as the design by Farmer (1998) is required to accurately represent a point within the sound space. The processing required by three-dimensional graphics routines in Java can cause an unacceptable latency. Compiled Java will execute faster, but may still not reach the required level of performance. A more interesting alternative to a Windows-based interface is to develop a proprietary user interface design that closely mimics the virtual sound space. An interactive hardware model that uses sensors to detect positions within the sound space can accurately track positions within the sound space. Hand or finger movement within the model generates positional co-ordinates. An example of such a device is the MIDI sensor chair (Paradiso) developed by the MIT Media Laboratory. Proprietary interface hardware avoids the processing overheads required by a Java-based Windows representation.

6. CONCLUSION

A spatial sound system built using an existing network and freely available software tools provides a low-cost spatial sound environment. By utilizing a distributed design where audio playback and routing calculations occur locally, processing loads are balanced across the network. The size and frequency of data transmissions is well within the theoretical bandwidth capabilities of current network technologies. Transmitted data is restricted to positional information and MIDI timing signals, with audio playback occurring locally. System performance will depend on the available bandwidth and the topology of the network. The system described in this paper provides an inexpensive research environment for audio perception, human computer interaction, music performance, virtual reality, and the development of real-time systems.

REFERENCES

Blauert, J. (1983). Spatial Hearing. Cambridge: MIT Press.

Eales, A.A. (1994). A Windows SurroundSound System - B.Sc (Hons) research project. Grahamstown: Rhodes University.

Farmer, D. (1998). Spatial Audio Prototype. [Web page] http://www.cs.colorado.edu/~farmer/spat/spat.html

GRAME. MidiShare documentation and code [Web page] www.grame.fr\Midishare

Kendall, G.S., Martens, W.L., Decker, S.L. (1989). Spatial Reverberation in Current Directions in Computer Music Research. ed. Mathews, V., Pierce, J.R. Cambridge: MIT Press.

Lehrman, P.D., Tully, T. (1993). MIDI for the Professional. New York: Amsco Publications.

Lindley, C.A. (2000). Digital Audio with Java. Upper Saddle River: Prentice-Hall.

MIDI Manufacturers Association. (1996). The MIDI 1.0 Specification. La Habra: MIDI Manufacturers Association.

Moore, F.R. (1989). Spatialization of Sounds over Loudspeakers in Current Directions in Computer Music Research. ed. Mathews, V. and Pierce, J.R. Cambridge: MIT Press.

Paradiso, J. Sensor Chair [Web page] http://web.media.mit.edu/~joep/TTT.BO/chair.html

Stevens, S.S. (1955). Measurement of Loudness. Journal of the Acoustical Society of America, 27.

Ward. P., Mellor, S. (1985). Structured Development for Real-Time Systems. New York: Yourdon Press.