A Distributed Approach to Surround Sound Production

Submitted in fulfilment of the requirements of

Master of Science

by

Adrian Wilfrid Smith

Computer Science Department
Rhodes University
Grahamstown 6140
South Africa

December 1998

Acknowledgements

I would like to thank Richard Foss, my supervisor, for his continued effort and support over the duration of my degree. Many thanks also go to the members of the Rhodes University Computer Science Department for their support.

I would also like to thank Lyn Leask, for her ceaseless encouragement and for proof-reading the drafts of this thesis.

Abstract

The requirement for multi-channel surround sound in audio production applications is growing rapidly. Audio processing in these applications can be costly, particularly in multi-channel systems. A distributed approach is proposed for the development of a real-time spatialization system for surround sound music production, using Ambisonic surround sound methods. The latency in the system is analyzed, with a focus on the audio processing and network delays, in order to ascertain the feasibility of an enhanced, distributed real-time spatialization system.

Contents

Chapter 1

Introduction

Our directional sense of sound incorporates complex mechanisms to localize sound sources in our environment. The study of directional hearing is not new, but a fair amount is not known about how we perceive directional sound. The recording industry has been able to capture the sound from live music performances for much of this century. New methods of recording and new recording technology have resulted in an increase in the quality of the recordings made. This quality included not only the audio quality of the recording medium, but also the spatial quality of the recording. One-channel monaural sound (mono) was replaced with two-channel stereophonic sound, which provided a richer and more spacious sound. However, the spatial quality of the reproduced stereo recordings was still far inferior to the sound experienced in the real performance, and the development of sound systems that could provide realistic sound from all directions was pursued. These systems are called surround sound systems.

Surround sound can be used in many diverse applications. Motion picture theatres have surround soundtracks for most major film releases, and the home theatre is demanding a higher quality of audio for video releases. Currently, the Digital Versatile Disc (DVD) standard is a technology that can provide enough data storage to be able to incorporate full length motion pictures and multiple channels of audio on one disc, and true multi-channel sound is now available for the consumer market. Surround sound is also used purely for the delivery of high quality music, and again, the DVD can supply enough capacity to carry multi-channel surround sound audio for this purpose. Surround sound is also useful for applications outside of music and soundtrack production. Virtual Reality environments (VR) have been largely focused on the 3D graphics components in simulating virtual worlds. These worlds are created to present an experience to the user that is similar to the real world. A fundamental goal of these systems is to increase the realism of the virtual world, and this can be accomplished with the inclusion of appropriate surround sound audio. An example of one of the early surround audio systems developed for virtual reality was the Convolvotron, developed by the National Aeronautic Space Administration (NASA) in the United States (Begault, 1994). Their VR system provides simulation and training facilities for astronauts, and the integration of convincing surround sound into these applications renders a more believable environment in which to train, especially for mission critical tasks. The popularity of video conferencing is also increasing, and there is application for surround sound in video conferencing applications. This is particularly applicable to videoconferences that involve many people, for example two groups of people in two separate boardrooms. Surround sound can be used to provide the spatialisation of a remote speaker's voice that would correspond to his physical position at the other end of the videoconference.

Many audio applications and computer music tools for music production are now available for modern desktop PC platforms. Most of these applications can only produce stereo sound. This could be attributed to the dearth of inexpensive multi-channel audio cards over the last few years. Recently however, has there been an interest in developing applications for creating, manipulating and reproducing surround sound in a multimedia environment, accompanying the release of affordable multi-channel audio cards such as the Event Electronics Darla (Event Darla Manual, 1997). There is a strong need for a PC-based application that provides the capability to create surround sound music compositions. This thesis suggests a distributed approach to the implementation of such a system, using a standard PC for the user interface of an environment for the creation and reproduction of surround sound music compositions. A distributed system may be able to provide a sound spatialisation environment that is superior to a standalone implementation on a PC, because of the computationally expensive audio processing involved. A distributed surround sound audio spatialisation system has been developed in order to provide such a tool to the computer musician, and to investigate the feasibility of a distributed approach.

Spatialized sound can be rendered over multiple speakers or via headphones. Headphone-based binaural surround sound produces acceptable soundfields for most applications, but also has several disadvantages that make its use for music presentation impractical. The demand for improved multi-speaker spatial quality led to the quadraphonic formats, and to the Dolby Surround Sound formats that were adopted for motion picture soundtracks. However, certain aspects of the soundfields generated by these systems which are adequate for soundtrack reproduction are undesirable for music reproduction. Ambisonic surround sound technology was pioneered in the 1970's and does provide surround sound that fulfills the requirements for surround sound music production. The unsuitability of surround sound systems used in motion picture theatres is discussed in Chapter 2, and motivations for using Ambisonics as our music surround sound technology of choice are presented.

Ambisonics technology provides methods for encoding audio into surround sound, and decoding that surround sound to an array of speakers. Chapter 3 discusses the two parts of the technology and the professional studio ambisonic surround sound format, B-Format, which is used to represent surround sound. Ambisonics also provides other formats for the delivery of surround sound audio to the consumer.

The real-time signal processing performance once obtainable only on dedicated hardware is now available on standard desktop PCs. In order to process audio into ambisonic surround sound, digital audio signal processing can be implemented in software with a fast processor. Ambisonic processing can be computationally expensive, and only a limited real-time system would be possible on a standard PC. A distributed approach to the implementation of a spatialisation system is therefore proposed in Chapter 4, which places the user interface on a PC, and most of the heavier ambisonic processing on a server machine, in an attempt to minimize the latency that would occur in an ambisonic spatialisation application. This distributed approach was developed in order to provide an enhanced real-time spatialisation system with the potential to be expanded by the introduction of further processing for increased spatialisation and audio quality.

Chapter 5 deals with the client-server approach used to distribute the spatialisation system and the software implementation details on both the server and client platforms. The use of a 3D mouse can be used to provide 3D input data in an intuitive way, and a user interface for interacting with a surround sound system is described. The performance of this system is explored in depth in Chapter 6. Firstly the overall latency in the system is considered, and the resulting response allowed by the system for the real-time spatialisation of audio data is investigated. Experiments have been conducted to measure the network and processing delays inherent in the system, and some optimization measures are suggested for reduction of the overall system latency. The way in which the spatialisation system was distributed in Chapter 4 is reviewed, and the possibility of implementing a standalone spatialisation system on a powerful PC is again weighed up against the client-server approach where latencies are incurred due to the network. The proposals in Chapter 4 are validated for the distributed approach because of the extensibility of the spatialisation and audio quality in the spatialisation system.

Qualitative topics relating to the performance of the spatialisation system include the real-time response that a user of the system is likely to perceive, and a discussion of some surround sound music production and compositional issues.

In conclusion, this thesis finds that the distributed spatialisation system described in Chapters 4 and 5 can provide a more enhanced spatialisation system than would be possible on a standard PC. General findings of this thesis are presented in the concluding chapter, and the possible extensions to the implemented system are discussed.

Chapter 2

The Choice of a Surround Sound Technology for Music

We perceive the natural world with three-dimensional sight and three-dimensional directional sound. From early music composition in the Renaissance period through to 20th century composition, an interest has been shown in the spatial aspects of sound. A simple example of this is the use of a double choir in several 15th century religious music compositions. The two choirs would be placed in separate locations in the acoustic space and parts of the composition divided between the two groups. The practicality of placing musicians in different parts of the building has its limitations, and in most cases, mainstream music compositions have been presented in front of the audience without spatial extension.

Ever since techniques emerged that allowed live music performances to be captured sonically, advances in recording technology have been focused on improving the capture of sound in performances, and the re-creation of that sound environment in the consumer's home. These methods have included the encoding and reproduction of sound into signal channel monaural format and dual channel stereo format, reproduced over one and two loudspeakers respectively. The development of surround sound was a natural progression in this field, and attempted to capture and reproduce more of the sonic information present in the original performance acoustic environment.

The spatialization of sound was also used widely in experimental electronic music during the 20th century. As opposed to recording and reproducing the spatial sound, the composer would create a spatial experience and render it to an audience over a number of speakers. An example of such a composition is Iannis Xenakis's *Concret PH* which was reproduced by an eleven-channel sound system to over 400 speakers mounted on the curved walls of an auditorium at the Brussels Fair in 1958 (Roads, 1996). The use of spatialization in electronic and computer music is a flexible technique in an electronic composer's toolbox, as is the ability to manipulate electronically the timbre of waveforms and other variables in the compositional environment. The available soundstage for a computer musician is now extensible to a 360-degree horizontal surround soundstage. If surround sound includes a height component, a fully three-dimensional soundfield is realizable.

Until recently, only dedicated hardware circuit boards provided enough signal processing power for digital audio applications. Modern multimedia-ready computers have the capability in terms of processing power, and audio interfaces to provide surround sound recording, composition, production and reproduction of spatialized sound. These systems can be completely configurable because of their implementation in software.

In order to develop a computer music surround sound application, a choice of which surround sound format to use must be made. This decision is usually dependent on the individual application. The human auditory mechanisms that enable the perception of directional sound need to be explained, before describing the more common surround sound systems and their unsuitability for music production. A brief section outlining the most important directional hearing mechanisms is presented below.

2.1 Human Spatial Hearing

Human directional hearing mechanisms are not yet completely fully understood. However, we do know the main ways in which we perceive the direction of a sound source. The branch of science called Psychoacoustics deals with the perception of physical sound events by the human brain. The detailed explanation of the principles of directional hearing is beyond the scope of this study, but the major mechanisms for directional hearing are briefly described. The reader is referred to Begault's "3-D sound for Virtual Reality and Multimedia" (Begault, 1994) for a more in-depth description of directional psychoacoustics.

The physical acoustic events that stimulate psychoacoustic responses in the context of directional sound are known as directional cues. One of the most important directional cues is based on the relative sound intensities detected by both ears. The sound emanating from a sound source will usually be detected by both ears. However, if the sound source is not directly in front of or behind the listener, each ear receives a different intensity of that sound. This is known as an Interaural Intensity Difference (IID) and is the main mechanism used to identify the position of a sound source. The sound will also reach one ear slightly before it reaches the other. This is the other major directional cue, and is known as the Interaural Time Difference (ITD). This difference is detected by the human auditory system as a phase difference between the sound reaching the left and right ears (Begault, 1994). The arrival of the sound from two sound sources, A and B, is shown in figure 2.1.

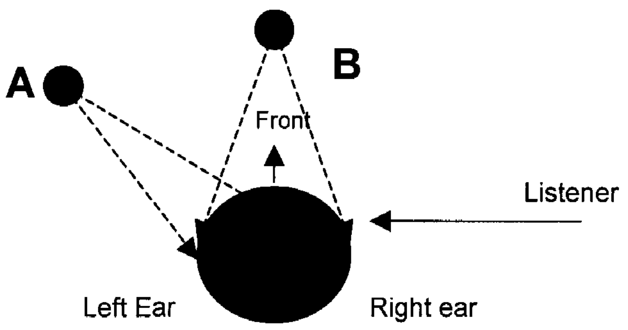


Figure 2.1 *The arrival of sound at the ears of the listener*

In Figure 2.1, sound source A is placed in a position left and to the front of the listener. The sound information emitted by source A will reach the left ear before it reaches the right ear. Source A will also sound quieter to the right ear than the left. These two cues determine the direction of the sound source perceived by the listener. In comparison, source B is placed directly in front of the listener. Both ears will receive identical acoustic information from the source, and the perceived direction will be determined as being directly in front of the listener.

The direction of a sound source is also determined psychoacoustically by the result of the interaction of the sound source with the acoustic environment. The diagram below shows a single sound source, with some of the reflections generated in an enclosed acoustic space. The reflections are generated by the listening environment and happen in a very short time immediately after the emission of the sound source. They are not audibly detected, but they do provide the brain with extra directional cues with which to locate the source. These reflections are called early reflections. Figure 2.2 shows an emitting sound source and the direct arrival of the sound at the listener position, as well as the early reflections generated by the acoustic environment

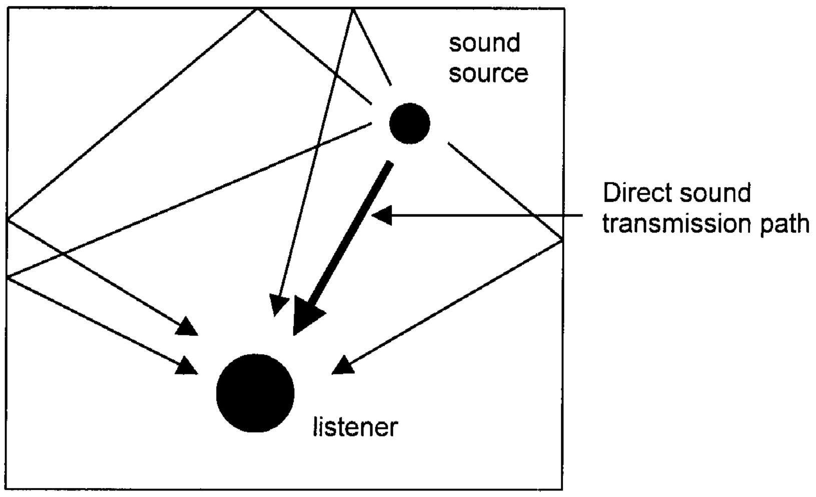


Figure 2.2 *Early Reflection Generation in an Acoustic Space*

The ITD and IID directional cues provide information for localizing (or determining the position of) a sound source. The early reflections also provide directional confirmation. Ambient sound accompanying the direct emission of sound from the source is caused by early reflections and by more delayed reverberation, and also constitutes an important part of spatial hearing.

The spatial cues described above are sometimes not enough to localize a sound source. An example is illustrated below.

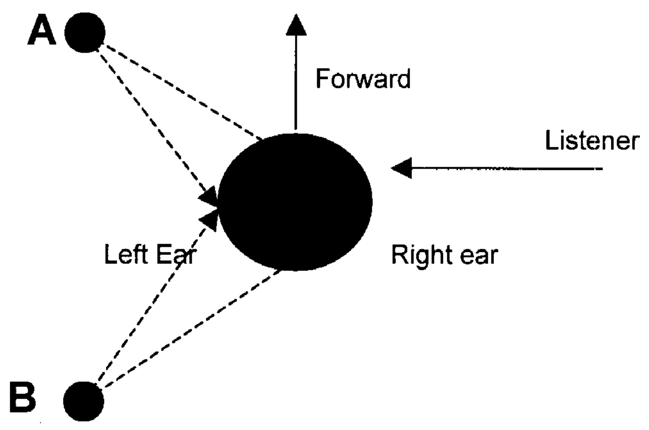


Figure 2.3 *Confusion between the localization of two sources*

The ITD and IID values for sound source A would be the same as those for source B, and listener confusion between whether a source is at the front or at the back can occur. In reality the situation is more complex than this because of the forward-pointing pinnae (the external part of the ear) and the acoustic shadowing of the right ear by the head. However confusion can occur very easily. Often, head movement is a way of confirming the position of a sound source. Other cues that provide information about the location of a sound source include the resonance effects that occur in the ear. This occurs in both the inner and outer ear (Burgess, 1992[a]).

2.2 3D Binaural Sound and Surround Sound

There are two main methods for presenting a listener with a surround sound experience, over headphones, and over an array of speakers surrounding the listener. Begault suggests using the term '3D sound' for spatial presentation over headphones, and 'surround sound' for spatial listening over multiple loudspeakers (Begault, 1994).

3D sound reproduced over headphones delivers two independent channels to the ears of the listener. The audio contained in these channels is processed to simulate the sound input that the listener's ears would receive in a real sound environment. The outer parts of the ears (or pinnae) filter the incoming sound. This filtering can be described by a Head Related Transfer Function (HRTF), which forms a major part of 3D soundfield synthesis. For every sound source position, there is a unique HRTF for both left and right ears. A binaural format can be synthesized by applying these HRTFs to an audio source. The disadvantage of using HRTFs is that each individual has a slightly differently shaped outer ear, and therefore requires their own unique set of HRTFs. A generalized set of HRTFs can be used, but will not provide directional cues optimally for each individual listener.

The orientation of the virtual soundfield in 3D sound is static. If the listener's head turns, the orientation of the 3D soundfield will turn with the head movement. This is undesirable for the perception of a realistic soundfield. For example, if a static virtual sound source is present in the 3D soundfield, and the listener turns his head, the virtual source will rotate with the change in head orientation. The only way to correct for this is to provide head-tracking of the head movement. Any change in the orientation of the listener's head would cause an adjustment of the 3D soundfield orientation presented over the headphones. Head tracking is not a feasible option as it is expensive and cumbersome to implement.

3D sound over headphones is also not suitable for a concert setting, where many people are sharing the same spatial audio experience. Each member of the audience would have to be provided with a set of headphones to experience the 3D soundfield. This becomes somewhat impractical for an audience of more than two or three people. Another disadvantage of using 3D sound in a concert setting is the need for individual sets of HRTFs for each person. Generic HRTF's built from population samples can be used, but they are not as effective as personalized HRTFs.

We have opted to use surround sound presented over an array of speakers instead of 3D sound for our spatialization system reproduction environment, because of the limitations of 3D sound mentioned above. Having said this, 3D sound is capable of producing a very realistic spatial sound experience. The impressive achievements of LakeDSP [1]* in this field demonstrate this capability, and this technology is very useful in virtual reality systems and head-phone based personal movie theatre systems such as LakeDSP's TheatrePhones technology.

2.3 Existing Surround Sound Technology

2.3.1 Mono, Stereo and Quadraphonics

Monaural sound recording was the first type of audio signal capture developed. This was a single channel format that was intended for reproduction over a single loudspeaker. Efforts to capture the depth of the soundstage more effectively than a monaural sound recording resulted in the advent of stereophonic sound recording. This technique uses a two-channel format and requires two speakers for optimal reproduction of the recorded material. There are however certain constraints in the placement of stereo speakers if the stereo material is to be perceived correctly by the listener. An ideal stereo speaker setup is shown in Figure 2.4, where the speakers are separated by 60 degrees with respect to the user.

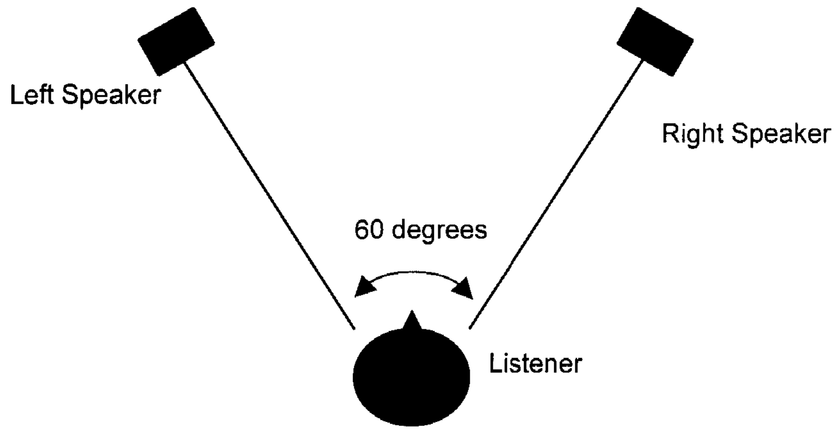


Figure 2.4 *Speaker Separation for Optimal Stereo Reproduction*

With this speaker separation, the best stereo reproduction is obtained. This setup is capable of producing a fairly wide soundstage. If the angle between the speakers is larger

* [ ] is used to denote a World Wide Web reference

than 60 degrees, a sonic 'hole' becomes noticeable between the two speakers, and if the speakers are placed at an angle of less than 60 degrees, the listener loses the desirable width of the soundstage. Another disadvantage of stereo reproduction is that the listener must be located at equal distances from each speaker. If the listener is closer to the left speaker than the right, the left speaker will sound louder and the balanced soundstage will degrade. Much orchestral music is recorded with stereophonic microphones and reproduced without tampering with the spatial quality captured by the microphones. However, one can artificially produce a stereo track from a mono signal by sending a different level of the same signal to the left and right outputs. The bigger the difference in amplitude, the more the sound will seem to emanate from the louder side of the stereo field (due to the Interaural Intensity Difference auditory mechanism). This is called panning. A large amount of music is produced with this technique, especially popular music where the notion of a soundstage often does not exist. This, strictly speaking is not true stereo as is captured by a stereo microphone, but rather what is known as pair-wise mixing, or amplitude panning. Nevertheless, pair-wise mixing is an effective method for placing sound across the stereo soundstage, provided that the speaker setup is as shown in the above diagram.

In an attempt to extend the stereo soundstage to a 360-degree surround soundstage, quadraphonic systems were developed. Quadraphonics applied the concepts of stereo reproduction to four speakers that were separated from each other by 90 degrees. A quadraphonic speaker setup is shown in Figure 2.5.

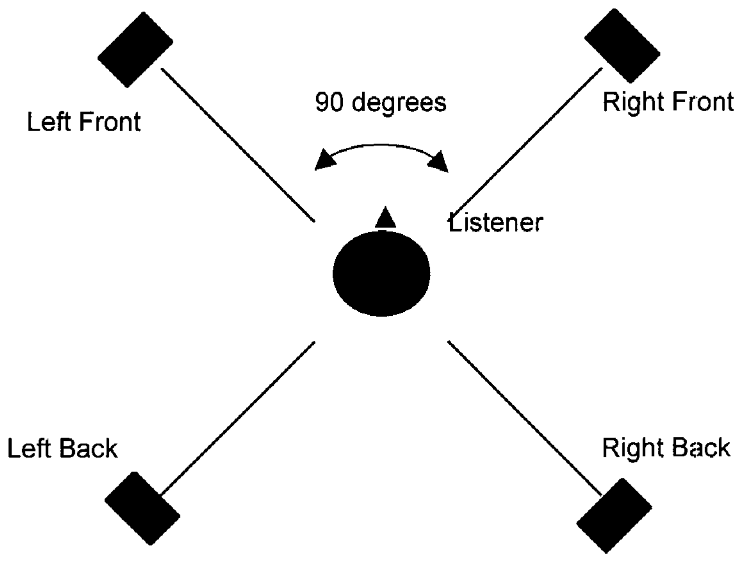


Figure 2.5 *Quadraphonic Speaker Setup*

The surround sound production in a quadraphonic system is fairly poor. As mentioned earlier in this chapter, directional hearing does not depend only on intensity cues, but also on phase and other factors. Because sound is placed in a quadraphonic system with level only (or pair-wise mixing), and because the angle between adjacent speakers is bigger than 60 degrees, the sound imaging between two speakers is not very good. The result is that there are audible 'holes' between adjacent speakers. There is virtually no inter-speaker imaging on the sides of the array and poor localization at the back of the speaker array. This lack of imaging can be demonstrated with a stereo system by orientating oneself 90 degrees from centre front so that one ear faces the pair of speakers. Any imaging in a stereo mix played over this system will suddenly be perceived as very poor. For these reasons, quadraphonic systems were not successful in the long term.

Quadraphonics also suffered from a lack of a delivery medium to deliver four channels of audio. Quadraphonic records and radio broadcasts attempted to encode these four channels into two channels. This is not possible without sacrificing a significant amount of the information contained in those four channels (Elen, 1991).

2.3.2 Motion Picture Theatre Surround Sound

The Dolby Surround sound system (Bamford, 1995) is a four-channel format used for motion picture sound. The four channels in the format are left and right stereo signals, a surround channel and a centre front channel. Initially, the stereo pair was used for the soundtrack, and the surround channel was usually fed to a number of surround speakers at the back of the theatre. This configuration became problematic regarding the dialogue in the motion picture. The dialog would be placed in the centre of the stereo pair by feeding equal levels of the dialog to the left and right speakers, thereby creating a 'phantom' dialogue channel in the centre of the stereo soundfield. However, this only worked for the audience that sat at an equal distance away from both speakers. All off-centre listeners would perceive the dialog coming from the speaker nearest to them instead of the middle of the stereo soundstage, as is usually preferable for motion picture presentation. The centre channel was therefore added to the surround format for dialogue. The surround channel is used for ambient sound and special effects, and there is no accurate localization perceptible outside of the frontal stereo stage of the speaker setup.

Dolby Digital evolved from the analogue Dolby Surround sound system. This digital surround format is a 5.1 channel system, consisting of left front, right front and centre channels as with Dolby Surround, but in addition, two full bandwidth left and right surround channels. There is also a limited-bandwidth Low Frequency Effects (the .1 channel of the 5.1 channel system) channel which provides audio information to a subwoofer. The 5.1 channels are compressed with a 'lossy' compression method, reducing the total bandwidth of the 5.1 channels to less than is required for one uncompressed channel. The coding method used to do this is called AC-3, and Dolby Digital is sometimes known as Dolby AC-3 (Davis, 1993).

There are other theatre sound systems such as Lucasfilm's Digital Theatre Systems (DTS) and the IMAX surround sound system that provide surround sound for motion picture presentation. Several home theatre formats are also available such as Dolby Pro-Logic, which encodes the four-channel Dolby Surround data into two channels.

All of the surround sound systems mentioned above are used in conjunction with motion picture presentation. The surround sound channels are used mostly for creating ambience, or for special effects. Their capability is adequate for the motion picture industry, but because they mostly rely on level-only localization, they have the same failings as quadraphonic systems such as poor inter-speaker imaging and inaccurate, unstable virtual sound source localization.

These systems are not suitable for surround sound music composition and reproduction. A surround sound technology for the production of music compositions should allow virtual sound sources to be placed accurately in the speaker array, even if the image lies between speakers in the array. The system should exhibit at least a planar 360-degree soundfield that treats all directions equally (unlike Dolby Surround where frontal images are good and images at the back of the speaker array are not convincing). The listeners in the audience should also receive fairly accurate sound localization wherever they sit within the auditioning environment. **Ambisonics** is a surround sound technology that provides these features.

2.4 Ambisonics

2.4.1 An Overview of Ambisonics

Ambisonics is a technology that satisfies many of the requirements of a high quality music spatialization system. Ambisonic surround sound encoding and decoding techniques apply the mathematical theories formulated by Gerzon (Gerzon, 1992[a]) in an attempt to satisfy several important directional sound perception mechanisms, such as intensity and phase. This technology, described in detail in the next chapter, essentially provides methods for capturing the information in a soundfield, and for reproducing the soundfield over a speaker array. The soundfields generated by Ambisonic technology are superior to the above technologies in many ways. Ambisonically generated soundfields provide accurate and stable virtual sound source localization, and the 360-degree soundfield produced is seamless, in other words, there are no 'holes' between adjacent speakers as is the case with the theatre surround systems such as Dolby Digital. The listening position, although optimal in the centre of the array, is not limited to the centre, and the 'sweet-spot' covers a fairly large area within the speaker array. An example of these features would be where a sound source is placed ambisonically on one side of the array between two speakers. The sound image will seem to emanate from a direction between the speakers, and not from the speakers themselves. If the listener moves from the central position to another position within the array, the sound image will seem to come from the same direction. This image stability, which is poor in theatre surround systems, is essential when presenting surround sound music to an audience.

Several Ambisonic formats exist which need to be decoded for reproduction over an array of speakers. The shape of the speaker array over which ambisonic material is reproduced is flexible, and the number of speakers can be varied. With more speakers, the soundfield reproduction is improved, but for practical purposes, four speakers are enough for the generation of a convincing horizontal surround soundfield.

Another feature that Ambisonics provides, unlike the other surround systems described here, is 'with-height' or 'periphonic' surround sound. Height information can be encoded into Ambisonic formats, and reproduced over a speaker array. A minimum of six speakers is used for periphonic surround sound reproduction. A useful capability of Ambisonic systems is their ability to move virtual sound sources within the speaker array, and not only around the perimeter where the speakers are placed. These ambisonic soundfield attributes are not common in theatre surround systems.

Because of the suitability of Ambisonic soundfields for surround sound music presentation discussed above, we chose to use Ambisonics as the surround sound technology in the spatialization system described in Chapter 5.

2.4.2 Existing Ambisonic Systems

2.4.2.1 Ambisonic Hardware

Many analogue ambisonic devices were built as ambisonic technology progressed in the recording industry. These devices include the Soundfield microphone (Borwick, 1987) that captures the compete soundfield into an Ambisonic surround sound format, Ambisonic decoders, which render the surround sound format over an array of speakers, and converters which transform the surround sound format into a stereo or monaural format. Other ambisonic mixing devices were developed to allow the manipulation of Ambisonic audio information during the sound post-production stages. An example of a device of this nature is the pan-rotate control patented by Michael Gerzon (Gerzon, 1984), which takes an analogue monaural audio input, and allows the panning of this source around the 360-degree perimeter of the soundfield. The pan-rotate controls enable an ambisonic mix to be built from source material that wasn't originally recorded in an ambisonic surround sound format. The functionality of this control is similar to the basic surround encoding capability provided by the spatialization system described in Chapter 5.

More advanced digital technology is now available, and companies such as LakeDSP and Meridian Audio [2] are producing high performance hardware for ambisonic format encoding and decoding. An example of a high-end ambisonic decoder is Meridian's 565 DSP Surround Processor, which in addition to Ambisonics, provides surround decoding of the other more popular surround sound formats.

2.2.4.2 Ambisonic Software

With an increase in the processing power available for Digital Signal Processing on desktop PCs, a fair number of ambisonic software applications have been developed in recent years. Some of the more advanced applications currently come from LakeDSP. An example of LakeDSP's software is Aniscape, which runs on their proprietary PC platform and simulates virtual acoustic environments. Farina recently developed a system for the software encoding and decoding of Ambisonics on a standard PC (Farina, 1998). The LAmb (Live Ambisonics) system was implemented on a Silicon Graphics Indy2 by Gow [3] for the live performance of spatialized music using Ambisonic technology. A different type of implementation is the CSound Ambisonic 'orchestra' written by Richard Furse [4] for Barry Vercoe's CSound DSP music language [5]. This provides a batch-processing environment in which one can spatialize monaural sound into Ambisonic surround sound, and decode this format to a number of speaker feeds. A real-time version of Csound has now been released, and this would allow the specification of a real-time spatialization system.

We chose Ambisonics for implementing a spatialization system for the reasons given in this chapter, and use this spatialization method to test the feasibility of our approach to distributed surround sound processing.

Summary

A technology for simulating directional soundfields must provide several spatialization cues for the listener. Many of the motion picture surround sound systems use amplitude only as a directional cue for the listener, and therefore provide fairly poor encoding and reproduction capabilities for musical compositions. Much improved soundfields can be created and reproduced with a technology called Ambisonics, which does provide many of the features required for music creation and reproduction. Ambisonics is described in detail in the next chapter.

Chapter 3

Ambisonics

Humans perceive sound with several psychoacoustic mechanisms. The direction from which a sound is perceived to originate is determined by many separate and sometimes conflicting variables. The most important of these is the amplitude of the source. If the left ear perceives a louder version of the source than the right, the source seems to come from the left side. This fact was the basis (and the downfall) of the quadraphonic sound systems and the surround sound systems that evolved out of Quadraphonics, such as Dolby Digital. These systems rely mainly on level to achieve virtual sound source localization.

Ambisonics is a technology that provides methods for encoding surround sound and reproducing that surround sound over an array of speakers. Essentially, in decoding this surround sound format to an array of speakers, Ambisonics attempts to reconstruct the wavefronts existing in the original soundfield. If these sound wavefronts are similar enough to the original wavefronts in the soundfield, then the human perception of this soundfield should resemble the perception in the original soundfield closely. Complete soundfields can be recorded, artificially created, processed in various ways, mixed (layered), and reproduced in Ambisonics. Several formats can be used for Ambisonic material, depending on the context in which the material is to be used. For example, B-Format is suitable for studio work, whereas UHJ (B-Format and UHJ are detailed later) is a more practical option for Ambisonics in home theatre.

As part of the research of the National Research Development Corporation (N.D.R.C.) of Britain in the early 1970's, this technology suffered a lack of marketing, and is therefore unfamiliar to many people. There have been many avid supporters of Ambisonics during the rise of the more common motion picture surround sound systems such as LucasFilm's Digital Theatre System (DTS). In an age where the development of high capacity digital media is making good progress, the demand for Digital Versatile Discs (DVDs) that provide music encoded in true surround sound is increasing. With more storage space available on DVDs, the production and distribution of multi-channel ambisonic material is becoming more practical.

3.1 The Soundfield Microphone

In order to create true surround sound recordings, all of the information in the acoustic environment needs to be captured. Various methods were developed for stereo recordings including the use of the Blumlein M-S stereo microphone configuration (Borwick, 1987), which uses a coincident pair of microphones (two microphones mounted as closely as possible). The aim is for each microphone to record a different aspect of the soundfield from a common position. One microphone is omni-directional for capturing sound from all directions, and the other, a figure-of-eight response microphone. The two signals output from this microphone are, in stereo terms, a 'sum' channel (L+R), and a 'difference' channel (L-R). From these channels, left and right stereo channels can be derived by addition or subtraction of the two signals respectively.

The Soundfield Microphone (Gerzon, 1975) was developed in order to capture not just stereo information, but to accurately record and represent sound emanating from all directions. It is an extension of the Blumlein coincident pair method, and allows complete

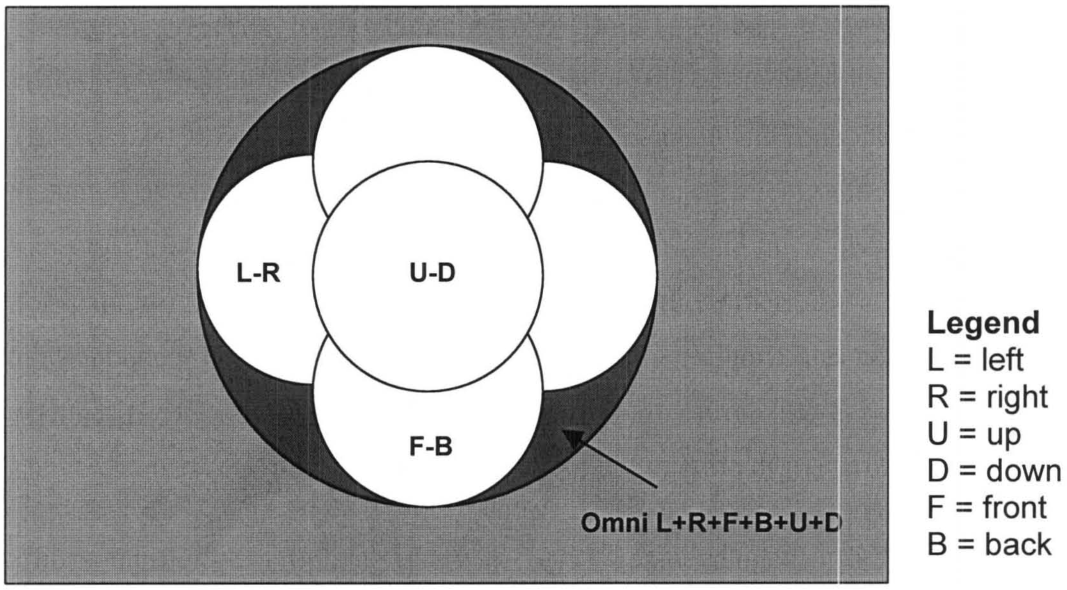


Figure 3.1 *Response pattern of the Soundfield Microphone*

soundfield recording by including two other figure-of-eight microphones. The practical coincidence of high quality microphone capsules is not possible, so the four microphones, one omni-directional, and three figure-of-eight microphone capsules (aligned along the x, y and z Cartesian axes) are placed in a tetrahedral array. The coincidence of the microphone capsules is made possible by electronic correction of the individual microphone signals.

The four outputs of the Soundfield microphone are an omni-directional channel W, and three directional channels X, Y and Z. These signals are equivalent to the four audio signals that comprise the main format used in Ambisonics for the representation of spatialized audio information. This format is known as B-Format.

3.2 Ambisonic B-Format

Ambisonic B-Format was designed to be a professional format for use in studios (Gerzon, 1984) and its design features allow the handling and processing of spatialized material in an elegant fashion. B-Format comprises four audio signals termed W, X, Y and Z. With these signals, all directional sound in a three-dimensional soundfield may be represented. If only horizontal surround sound is required, signals W, X and Y are used. Sound from all directions is represented by the W signal, and the X, Y and Z signals carry sonic information for the components of directional sound along these axes within a soundfield. These signals are the same as those output by the Soundfield microphone.

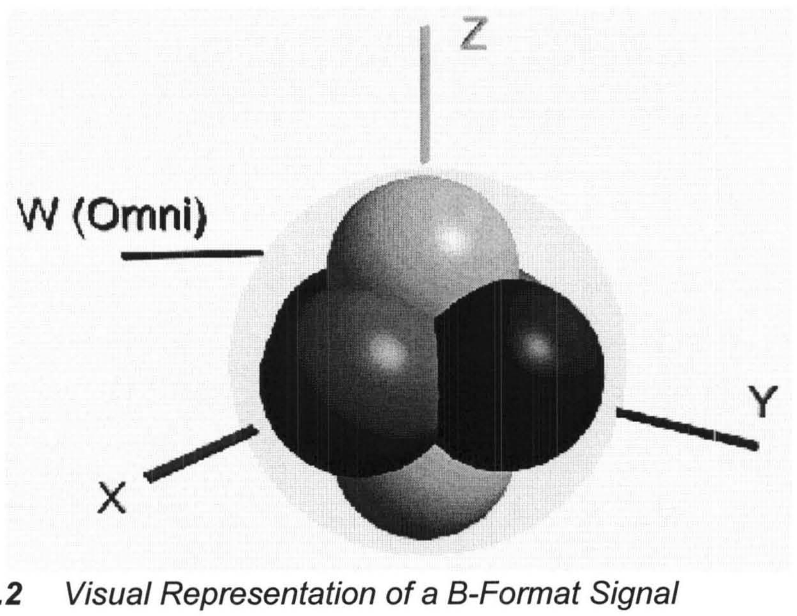


Figure 3.2 *Visual Representation of a B-Format Signal*

3.3 B-Format Signal Processing

3.3.1 Mixing of B-Format

One of the important features of B-Format as a surround sound format, is that independent soundfields may be mixed to form composite soundfields. This is very useful if one wants to overlay soundfields from different sources, for example, combining the B-Format recording from an orchestral concert with another soundfield, perhaps a natural spatialized recording of a waterfall. A more familiar example could be the incorporation of real cannonfire at precisely specified locations in the 360-degree soundfield into a B-Format recording of Tchaikovsky's *1812 Overture*.

Two B-Format signals are mixed by combining the corresponding components of each signal, i.e. the X components are added together, the Y's added together, etc. This is obviously more computationally expensive than mixing two ordinary stereo signals.

It will be shown later, that an artificial B-Format soundfield can be created from any audio source. In this way, the system described in this thesis can spatialize separate audio sources, and combine them into one B-Format soundfield.

3.3.2 B-Format Soundfield manipulation

Other processing may also be applied to B-Format signals. A B-Format soundfield may be manipulated in a way similar to the transformations possible with 3D graphics objects. These include the rotation, tilting or tumbling of the complete soundfield, and other transformations like shearing, stretching, and mirroring of the soundfield. While the musical value of such effects is still to be explored fully, an immediate application for this kind of sound manipulation is in virtual reality systems. The rotation of the soundfield in combination with the rotation of a 3D graphic scene would increase the realism of the virtual world.

An illustration of three possible manipulations of a soundfield is depicted in Figure 3.3.

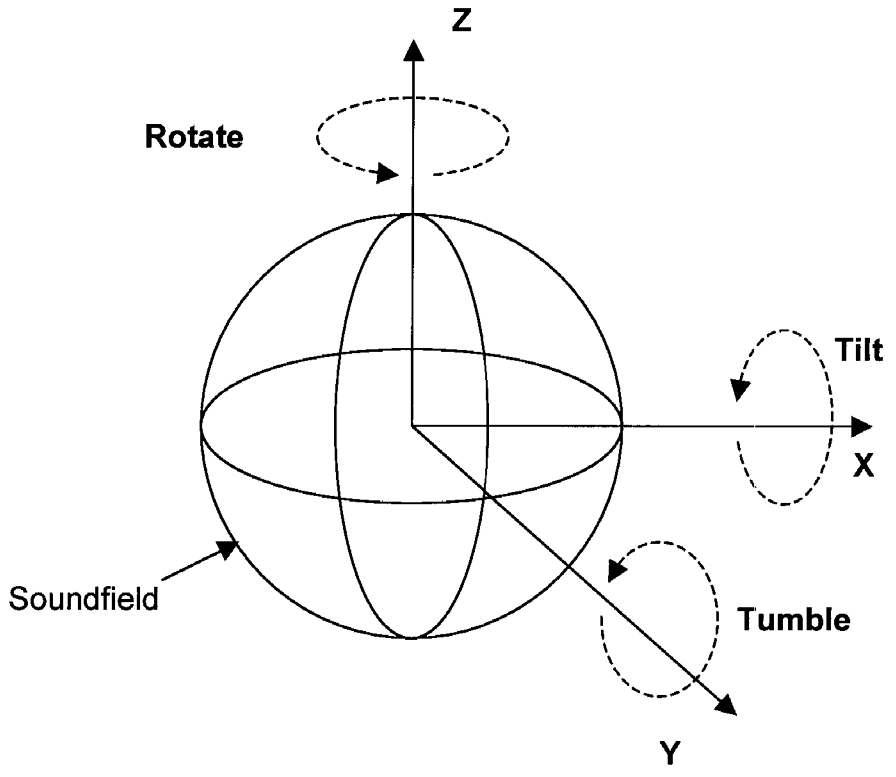


Figure 3.3 *Manipulation of a Soundfield*

Given a B-Format audio signal containing components, **W**, **X**, **Y** and **Z**, Malham demonstrates the manipulation of this signal to result in a rotation of the soundfield about the Z axis (Malham, 1993).

$$\begin{aligned} X' &= X \cos A - Y \sin A \qquad A = \text{angle of rotation} \\ Y' &= X \sin A + Y \cos A \\ Z' &= Z \\ W' &= W \end{aligned}$$

Equation [3.1]

W′, **X′**, **Y′** and **Z′** form the rotated B-Format signal.

As with 3D graphics transformations, any number of manipulations may be combined into one matrix, allowing many transformations to be applied to the soundfield with one matrix multiplication (**k1** – **k16** are constant coefficients of the composite matrix).

$$\begin{array}{lcllllc}
X' & = & k1 & k2 & k3 & k4 & X \\
Y' & = & k5 & k6 & k7 & k8 & Y \\
Z' & = & k9 & k10 & k11 & k12 & Z \\
W' & = & k13 & k14 & k15 & k16 & W
\end{array}$$

3.4 Artificially Encoding Sound into B-Format

A major part of Ambisonics is the encoding of sound, be it via a soundfield recording or artificial synthesis, into ambisonic B-Format. Sound is usually encoded into B-Format by a Soundfield microphone. However, in many cases it is desirable to encode monaural audio sources into B-Format directly. Analogue ambisonic equipment has been designed for this purpose. An example is an ambisonic panoramic potentiometer (panpot) (Gerzon, 1984), which will pan a monaural audio source around the perimeter of the B-Format soundfield. The outputs of this panpot are B-Format signals, which are then decoded for reproduction over a speaker array.

One may digitally encode a monaural sound into a simple B-Format surround sound representation, with the simple formulas below (Malham, 1990). The monaural source is placed at a position within the three-dimensional soundfield designated by a set of 3D coordinates X, Y and Z.

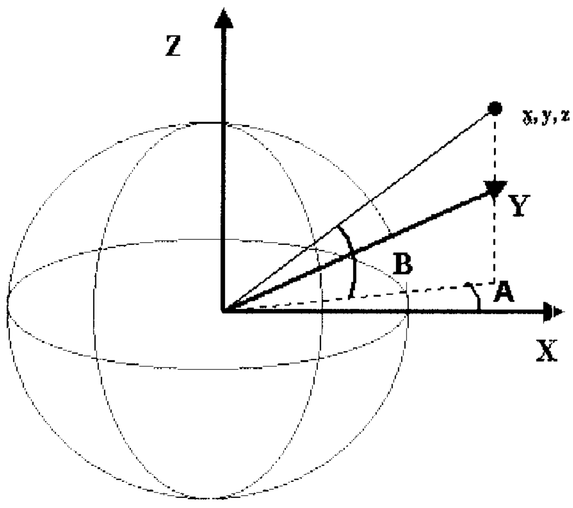


Figure 3.4 *Coordinate System for Virtual Sound Source*

X = Input signal × **cos A** × **cos B**
Y = Input signal × **sin A** × **cos B**
Z = Input signal × **sin B**
W = Input Signal × **0.707**

where: A - anti-clockwise angle between the centre front position and the desired sound location

B - angle of elevation between the centre front position and the desired sound location

Equation [3.2]

The centre of the soundfield, where the X, Y and Z axes meet, will be referred to as the origin, and is the ideal location for the listener. The encoding equations may be simplified if the encoding of audio is done on the surface of or within a unit sphere (a sphere of radius one from the centre of the soundfield). The 3D coordinates of the virtual sound source are limited to a unit sphere by ensuring that the following relationship is true:

$$\sqrt{(X^2 + Y^2 + Z^2)} <= 1$$

Equation [3.3]

If the 3D coordinates are outside the unit sphere, the directional data is not decoded properly during reproduction, and sounds will be drawn toward the individual speakers in the array (Malham, 1993).

The simplified B-Format encoding formulas may be written thus:

$$\mathbf{X} = \textbf{Input signal} \times X$$
$$\mathbf{Y} = \textbf{Input signal} \times Y$$
$$\mathbf{Z} = \textbf{Input signal} \times Z$$
$$\mathbf{W} = \textbf{Input Signal} \times \mathbf{0.707} \qquad \text{Equation [3.4]}$$

X, Y and Z are the 3D coordinates of the virtual sound source in the coordinate space of a unit sphere centred on the middle of the soundfield, and the resulting encoded B-Format signal comprises **X, Y, Z and W**. The input signal is modified by a factor of 0.707 to form the W component in order to provide a more equal distribution of B-Format component levels. This is a convention on which many B-Format decoders (especially the older analog ones) are based (Malham, 1990).

As a virtual sound source moves in from the surface of the unit sphere toward the origin, one would expect the sound to increase in intensity. However, as can be seen from the above equations, the intensity of the sound will actually decrease when moving toward the centre.

Malham suggests a modification of the W component to keep the sound levels fairly constant:

$$\mathbf{W} = \textbf{Input Signal} \times \mathbf{(1 - 0.293(x^2 + y^2 + z^2))} \qquad \text{Equation [3.5]}$$

x, **y** and **z** are the sound source 3D coordinates.

In some cases, an increase in sound level may be desirable when moving toward the centre within the unit sphere, and this can be accommodated with a further modification of the W component.

The above encoding method only caters for the placement of sounds on the surface of or within the unit sphere. One way of achieving the localization of sounds outside of this sphere, as shown in Figure 3.5, would be to find the point of intersection of the unit sphere and a line joining the origin and the desired virtual sound source coordinates.

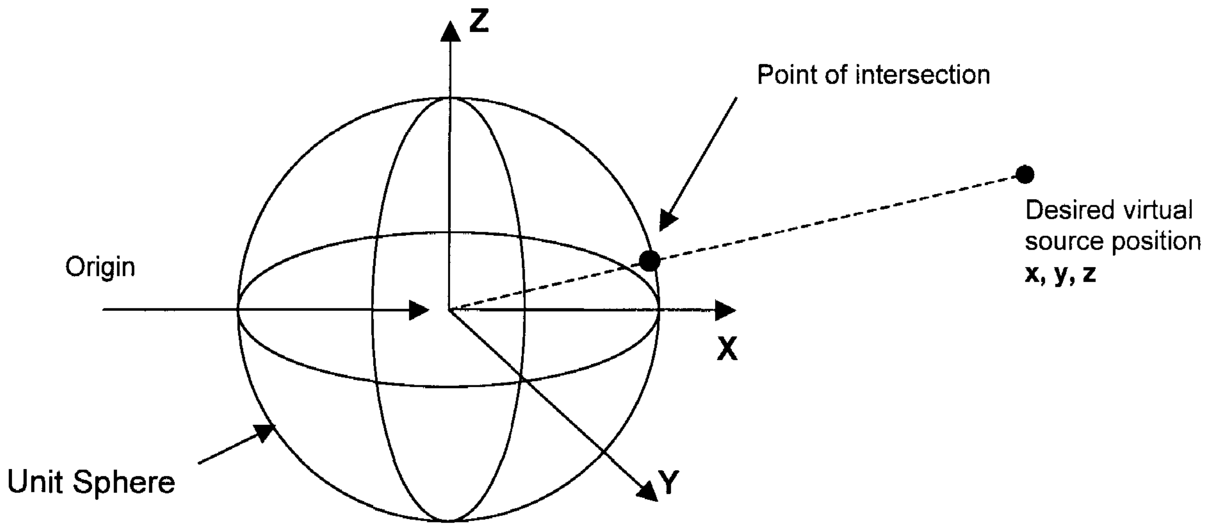


Figure 3.5 *Encoding 3D coordinates outside the unit sphere*

A sound source is encoded into B-Format at this position, and the components of the B-Format signal can then be scaled down according to the distance between the desired virtual sound source position and the origin, using an inverse square power law (Begault, 1994).

The above encoding method does not incorporate further DSP for increased localization accuracy and realism. There are other factors that affect the perceived quality of directional sound in nature, many of them dependent on the distance between the sound source and the listener. Air absorbs much of the high frequency range of sound and as distance increases, the high frequencies become more muffled. Bass frequencies also vary with distance. A low frequency sound source near to a listener will sound more 'boomy' than it does when further away from the listener. Doppler effects can also be added, which would vary the pitch of rapidly moving sound sources, for example, simulating the sound of a car rushing past with the associated variations in pitch.

An important part of the realism of a surround sound system lies in the ambience that is created in the reproduction environment. It is possible to artificially introduce early reflection patterns into a B-Format signal with the aim of increasing the perceived localizational accuracy. The ambience of the environment would be introduced by applying digital reverberation to the W component. Further DSP may be used along with 3D room models to recreate the reflections and ambience of specific acoustic spaces. An example of aurilization software that performs these operations is the Aurora software, developed by Angelo Farina at the University of Parma, Italy [6].

3.5 Reproduction of B-Format

The decoding of B-Format and the rendering of the result over a speaker array is the other fundamental part of Ambisonic technology. The goal of the decoding process is to recreate the directional sound information present in the encoded B-Format signal, thereby rendering a believable approximation of the soundfield. The decoding process is completely independent of the recording and encoding of B-Format. The B-Format representation assumes nothing about the reproduction process. Thus, the size, shape, or number of speakers in the reproduction array is variable and this information is not needed during the B-Format encoding or processing. This is unlike stereo, where it is assumed that the listener will have his speakers placed in the recommended stereo positions, i.e. equidistant from the listener, and placed 30 degrees either side of centre front.

Only analogue decoders were available for decoding B-Format until recently. These decoders are only configurable to a degree, and allow the number of speakers in the speaker array, and the physical layout of that array, to be specified. Usually a speaker array comprises a number of speakers of the same type, and the shape of the array must be fairly regular. With adequate processing power available on standard PCs, B-Format decoding can now be done in software. The decoding of B-Format in software allows the decoder to be configurable in every way. A software decoder can correct for unusual speaker placements in irregularly shaped arrays, or to allow different types of speakers to be used in the same array. In practice, only a few speakers are used in the reproduction array, though theoretically thousands of speakers would be required to reproduce exactly the original soundfield (Malham, 1991). A small number of speakers nevertheless approximate the soundfield very well. A minimum of four speakers are needed for horizontal B-Format reproduction (the Z component is ignored in this case), and a minimum of six speakers for full 3D reproduction. The main layout used in the software decoder for the system described later in this thesis is a cuboid array of eight speakers.

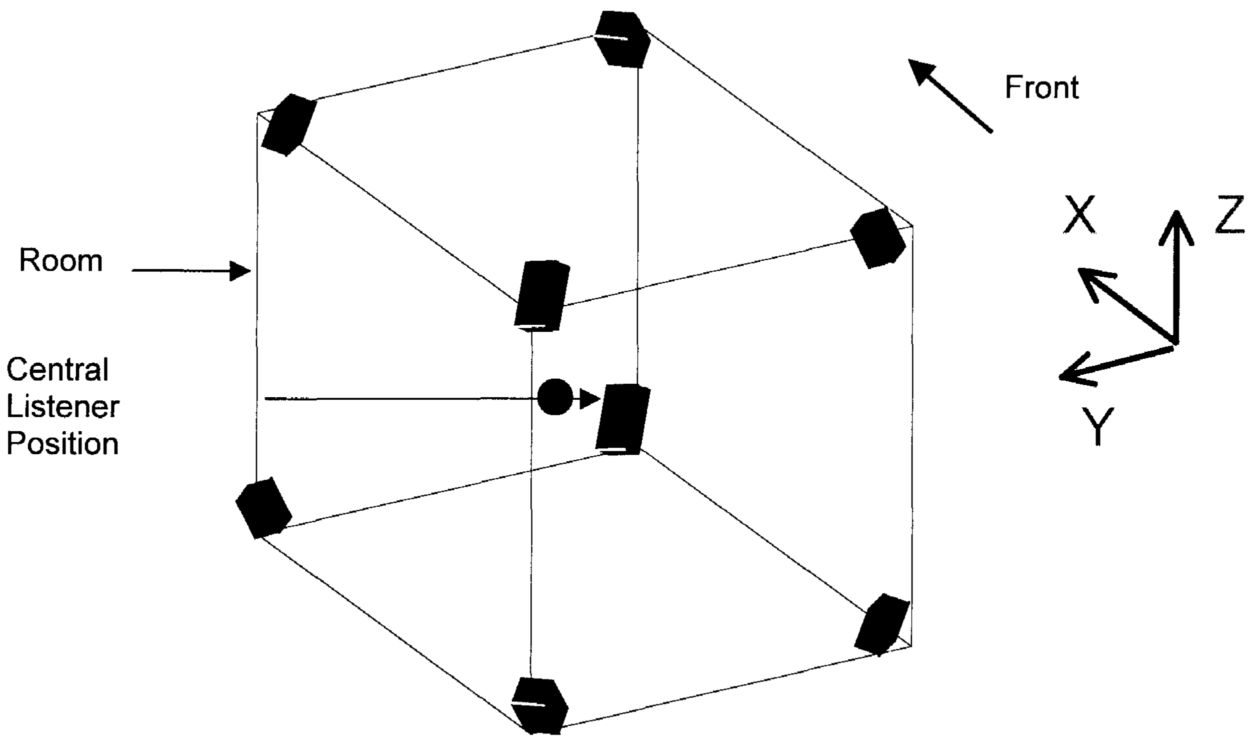


Figure 3.6 *Cuboid Speaker Array*

In this regular array, all of the speakers are at an equal distance from the listener. An example of the decoding of B-Format to this type of array is given in Equation 3.6. Each speaker receives a mix of the various components of the B-Format signal determined by its position in the array. The equation shown below is basically an amplitude matrix for mixing together different quantities of W, X, Y and Z.

The speaker feeds ($F_1 \ldots F_n$) for a cuboid array are of the form (Farina, 1998):

$$F_i = \tfrac{1}{2} [G_1 \mathbf{W} + G_2 (\mathbf{X} \cos(\alpha) + \mathbf{Y} \cos(\beta) + \mathbf{Z} \cos(\gamma))] \qquad \text{Equation [3.6]}$$

The vector (originating from the centre of the array) along which each individual speaker is placed is separated from the x, y and z coordinate axes by angles α, β and γ respectively. Different auditory cues are used in directional hearing depending on the frequency of the sound. G_1 and G_2 therefore have different values for below and above 500 Hz. Some ambiguity remains however as to exactly what these values should be. Farina (Farina, 1998) suggests several combinations for different room sizes, and for low and high frequencies. Michael Gerzon's gains for high and low frequencies in a studio environment for horizontal only surround sound are quoted by Farina as follows:

	G_1	G_2
High Frequencies	$\sqrt{8/4 \cdot N}$	$\sqrt{8/2 \cdot N}$
Low Frequencies	$\sqrt{8/6 \cdot N}$	$\sqrt{8 \cdot 2/3 \cdot N}$

N = Number of speakers in array

Figure 3.7 *Horizontal B-Format Decoder Gains for different Audio Frequencies*

For full 3D surround sound rendering, Gerzon suggests these values [7]:

Frequency	G_1	G_2
> 500 Hz	$\sqrt{2}$	$\sqrt{2}$
< 500 Hz	1	$\sqrt{3}$

Figure 3.8 *Periphonic Decoder Gains for High and Low Audio Frequencies*

As with most multi-speaker reproduction systems, there are ideal conditions for the listening environment which should be adhered to if the full potential of the reproduced soundfield is to be exploited. An anechoic room is suggested so that other reflections in the room do not interfere with the auditory space. For example, if reflections of speaker outputs from concrete walls and floors are generated within the soundfield, they can be interpreted by the listener as false localization cues, and will cause a degradation of the auditory illusion. All of the speakers used in the reproduction array should be identical, and should be flat-frequency response speakers. For a regular cuboid array, as is used in this study, the listener should be located in the centre of the array, i.e. equidistant from all speakers in a position known as the 'sweet spot'. Although the soundfield is reproduced optimally for a listener located in the sweet spot, in practice this spot extends over a large area in the middle of the speaker array.

3.6 Other Ambisonic Formats and the Delivery Mechanisms

When Ambisonic theory was established, the need for a delivery mechanism for surround sound arose. B-Format, while perfect for many studio applications, requires at least 3 channels and is unwieldy for a consumer delivery format. This is especially true when regarding the lack of compatibility of B-Format with commonly used monaural (one channel) and stereophonic (two-channel) equipment. A format was therefore developed, named the UHJ system (Universal HJ, originating from HJ, which was one in a series of experimental surround technologies researched by the British Broadcasting Corporation)

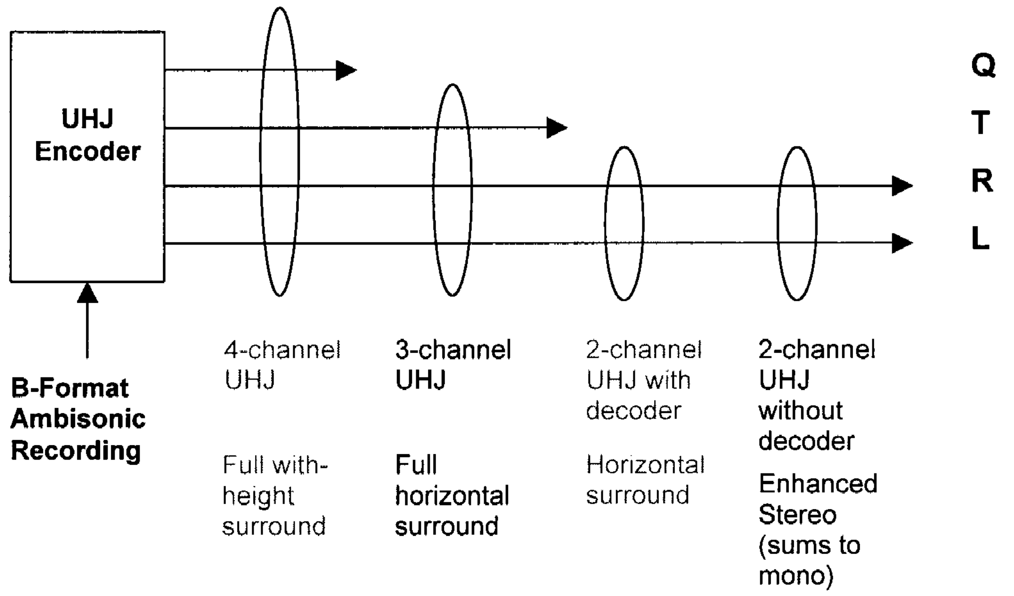

UHJ Encoder
Q
T
R
L
B-Format Ambisonic Recording
4-channel UHJ
Full with-height surround
3-channel UHJ
Full horizontal surround
2-channel UHJ with decoder
Horizontal surround
2-channel UHJ without decoder
Enhanced Stereo (sums to mono)

Figure 3.9 UHJ encoding and Delivery of Ambisonic Material (Elen, 1998)

(Gerzon, 1983). This system allows the hierarchical encoding of Ambisonic B-Format. This means that an increasing number of capabilities are available depending on the number of transmission channels available. Figure 3.9 shows the various types of UHJ format into which B-Format can be encoded (Elen, 1998).

If four transmission channels are available, 4-channel UHJ can be decoded to provide fully periphonic (with-height) surround sound. If only 3 channels are available, the Q channel is not used, and full horizontal sound can be decoded. Similarly, only L and R channels may be used and decoded. This option provides good horizontal surround sound, although with some defects. These defects can be made less noticeable if they appear in the rear of the soundfield where human directional perception is less sensitive. If a UHJ decoder is unavailable, 2-channel UHJ exhibits excellent mono capability, and provides an enhanced stereo soundfield when played without decoding on stereo equipment.

The most commonly used type of UHJ is the 2-channel version, which does not necessarily require a decoder. A large number of 2-channel UHJ music recordings have been released on compact disc, and this number exceeds the number of music CDs released in any other surround [8]. UHJ can also be encoded and decoded in software, but the process is more complex than the software encoding and decoding of B-Format.

The demand for high density CD's that hold sound only is growing. Digital Versatile Discs (DVDs) provide the extra capacity needed for the distribution of multi-channel surround sound. High-Quality Audio Discs (HQADs) are DVD discs that hold audio information only and multi-channel surround sound music releases are now possible with this medium. The DVD standard specifies the Dolby AC-3 multi-channel perceptual encoding of multi-channel audio on DVDs. This audio is in 5.1 format. However, Dolby AC-3 and similar systems such as MPEG-2 employ 'lossy' compression on the audio streams to make the audio less bandwidth intensive.
The delivery of Ambisonic material on DVD raises several issues. It would be preferable to employ lossless compression on the audio streams, so that no audio is lost even though

there is a significant saving in the amount of transmission data needed for multi-channel audio. Meridian Lossless Packing (MLP) was developed for this purpose, and makes enough space available on a standard DVD for 89 minutes of 6-channel, 24-bit audio sampled at 96 kHz (Fox, 1998). The option of storing an Ambisonic hierarchically-encoded multi-channel stream (UHJ) would also be important in the specification of the HQAD. This would allow surround information to be extracted with an appropriate decoder.

A group of audio engineers called the Acoustic Renaissance for Audio (ARA) has recently made a proposal to the DVD steering committee, which is outlined here. The first request is to include an optional flag indicating that lossless compression has been implemented on the audio. If this is not possible, ambisonic audio quality will be degraded, as is the audio quality of Dolby Digital and DTS, which do employ 'lossy' compression encoding schemes. With very high quality audio, especially at a professional level, this is not acceptable. The other proposal is to include an optional flag that denotes whether a hierarchical-encoding scheme such as UHJ has been employed. In this way, ambisonic material can be encoded and distributed on DVD. The disadvantage of using UHJ encoded material is that the consumer needs a special decoder at the reproduction stage.

If ambisonic hierarchical encoding schemes cannot be used, either because of the exclusion of the flag denoting hierarchical encoding, or because of the need to allow consumer reproduction without UHJ decoding, there are some other options available. The ambisonic material can be decoded into speaker feeds, and then encoded into Dolby AC-3. However, the feature of ambisonic decoders allowing one to configure the decoder for any speaker placements will be lost. 5.1 surround systems have 'recommended' positions for their speakers, and the ambisonic material will have to be pre-decoded for those speaker positions. The advantage of this is that ambisonic material may be delivered over the common 5.1 home theatre systems. This format is known as GD-Format (G-Format, Decoded).

Elen suggests another type of G-Format encoding called G+2 Format, with several features not provided by the above formats. It provides speaker feeds encoded into a 5.1 surround format (GD-Format), and a 2-channel UHJ stream on the same disc (Elen, 1998). The UHJ stream can be used for mono or enhanced stereo reproduction, or it can be converted to planar surround sound with a UHJ decoder. Furthermore, B-Format can be recovered from GD-Format and then decoded to a speaker array for higher quality surround sound than that delivered by 2-channel UHJ.

Summary

Methods for the encoding of sound into Ambisonic surround sound and the reproduction of this encoded sound over a speaker array have been described. The directional sound contained in a soundfield is encoded into Ambisonic B-Format, which consists of four audio channels, namely W, X, Y and Z, and can be recorded from a natural soundfield or synthesized from monaural audio. This format is reproduced over an array of speakers by decoding of the B-Format. This decoding process is flexible, and can cater for many speaker array layouts. Other Ambisonic surround sound formats also exist which are more suitable for the delivery of ambisonic surround sound to the consumer. Analogue Ambisonic technology has been in existence for many years, however now the audio processing can be done digitally in computer software. There are significant processing costs to this approach, and the processing overheads resulting from ambisonic audio processing are explored in Chapter 4.

Chapter 4

Processing Implications of an Ambisonic Spatialization Application

The development of an Ambisonic spatialization application raises many issues concerning digital audio processing. Several stages exist in the transformation of monaural audio into Ambisonic spatialized sound, and the processing times required by these stages may not be trivial. The spatialization of audio in real-time also presents some of the problems common to real-time interactive multimedia applications, for example the buffering of audio streams which are subject to varying latency.

The audio path through the various processing stages is broken down into the sections where the major sources of latency in such a system exist. Experiments to time the execution of these sections are then conducted so an idea can be formed of the potential total latency inherent in a spatialization system.

The required real-time nature of the proposed spatialization application relies on minimum latency throughout the system, and several options are available for the implementation of a real-time system. The possibility of an implementation on a standalone desktop PC is explored in terms of what delays in processing and buffering are likely. A decision was made to distribute the application with a client-server architecture with a view to reducing the digital audio processing time in some of the audio processing stages.

The audio information is segmented into buffers for digital processing. The times measured for the computation of the segments of code detailed in this chapter were measured as follows: each section of code under test was run 10000 times, the lapse between start and finish was timed manually, and this time was divided by 10000. This method produces a fairly accurate timing of the calculation time for each section of code.

The diagram below illustrates the main sections needed for the spatialization of sound from a monaural audio source. Each section has an associated delay and these delays are explored in the following sections. The system aims to perform the encoding, any additional processing, and the reproduction of the spatialized sound with an overall latency low enough to provide the user with a real-time or near real-time response.

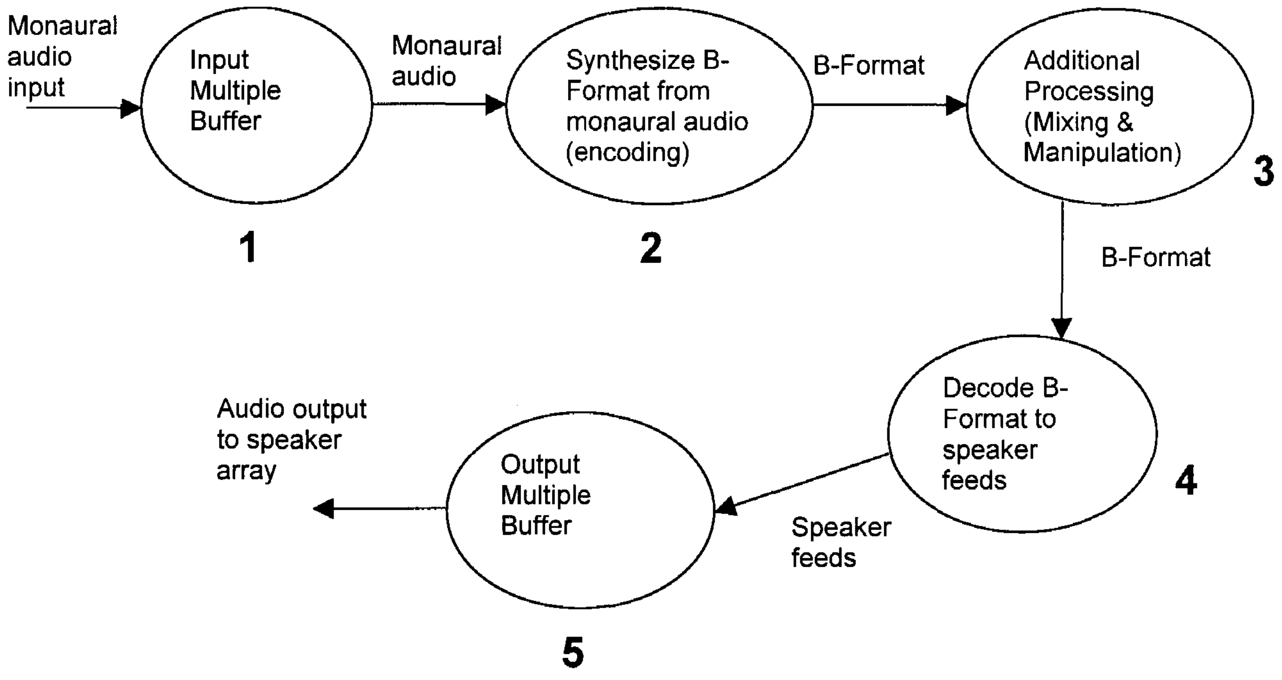

Monaural audio input
Input Multiple Buffer
1
Monaural audio
Synthesize B-Format from monaural audio (encoding)
2
B-Format
Additional Processing (Mixing & Manipulation)
3
B-Format
Decode B-Format to speaker feeds
4
Speaker feeds
Output Multiple Buffer
5
Audio output to speaker array

Figure 4.1 *Data flow diagram of the audio path in an Ambisonic spatialization application*

The software processing of audio in digital audio applications will always create small delays in a continuous audio stream. A stream of digital audio is processed in real-time by processing one small chunk of audio at a time. This chunk is simply a buffer containing audio data, and an audio stream consists of many of these buffers. The size of an audio buffer must be small in a real-time application, so that the time taken to capture each audio input buffer is not too long. On the other hand, the buffer size cannot be too small either, otherwise the limitations of the CPU and software will come into play. A standard audio buffer size of 4K bytes was therefore used during implementation under the Windows 95 operating system running on a Pentium 200MHz machine. A smaller buffer size of 3K bytes was found to work well in the experiments presented in Chapter 6.

4.1 Multiple Buffering of the Input and Output Audio Streams

Multiple buffering stages, shown in Figure 4.1 as parts 1 and 5, are needed to guarantee smooth audio input and output streams. The capture of an audio input into a stream of many buffers needs to be implemented so that no audio information is lost. If a single buffer is used to capture all of the audio, there is a small delay during which the full buffer is processed, and a small amount of incoming audio data will be lost. A supply of empty buffers is therefore maintained at the input, so that when the currently filling buffer is completely full, there are other buffers for the immediate continuation of audio capture. The only delay inherent in this procedure is the time taken to fill each audio buffer with incoming audio data. Only then will each individual buffer be processed.

Within the audio application, each individual buffer takes a finite time to process, causing a delay in the audio stream. Other factors affect the continuous audio stream, such as operating system interference. These delays are not easy to quantify, and are usually subject to a large amount of variation. If the audio stream is simply input and output without considering the effect of these delays, audible gaps in the stream will result. Multiple buffering is also needed at the audio output stage in a digital audio application. Accumulating a number of processed buffers before commencing audio output has the effect of absorbing these delays thereby producing a smooth audio output stream. The output multiple buffering causes a significant delay in the system depending on the number of buffers accumulated.

If the application provides only the output of audio, the delay in buffering the audio for output is not too important. An example of this is the playback of an 8-track piece. A delay of two or three seconds before the start of playback is perfectly acceptable. However if the system needs to process audio in real-time, and especially if an audio input needs to be processed and output in real or near real-time, the latency due to buffering the input and output becomes a problem. It will be shown in Chapter 6 that these delays overshadow any other delays in the Ambisonic spatialization implementation described in Chapter 5. The optimization of these stages is essential in providing a real-time audio application. This can be done by reducing the size of the audio buffers that are processed. The capture time of an individual buffer is decreased, and reduces the output buffering time which is dependent on the size of buffer used. However, this solution may not be effective on the output buffering stage. If the buffer size is reduced the output buffering cushion size and its resultant effectiveness in smoothing the output stream, is affected. It is often necessary to increase the number of accumulated buffers if the buffer size is reduced, and maintain a consistent total output buffering size. The problems with reducing the buffer size highlighted at the beginning of this chapter will also have an effect on the real-time nature of the system.

These buffering delays are unavoidable in all digital audio applications regardless of the specifics of the implementation. For both standalone applications and distributed applications multiple buffering is essential, and the delays involved are similar.

4.2 B-Format Synthesis from Monaural Audio Data

The simple encoding of monaural audio and associated 3D coordinates into B-Format is not computationally expensive. Malham's formulas quoted in Chapter 3 are repeated here for convenience.

$$\mathbf{X = Input\ signal} \times X$$
$$\mathbf{Y = Input\ signal} \times Y$$
$$\mathbf{Z = Input\ signal} \times Z$$
$$\mathbf{W = Input\ Signal} \times \mathbf{0.707}$$

X, Y and Z are the 3D coordinates representing the desired position of the virtual sound source.

A monaural 4K buffer is converted into a four track B-Format buffer of length 16K bytes with the above equations. The time taken to fill this B-Format buffer was measured at

2.75 milliseconds on a Pentium 200 MHz machine. The delay caused by this encoding is trivial compared to other delays in such an application. These formulas are used to perform simple B-Format encoding. In order to create an artificial soundfield that approximates how sound in the natural world is perceived, other processing needs to be added to the B-Format synthesis process. Early reflections can be digitally generated and incorporated into the original B-Format signal. B-Format can also be enhanced with digital reverberation to colour the virtual environment with ambience. More advanced reverberation DSP techniques use 3D graphical room modeling and ray-trace sound paths in the virtual room to produce accurate reflections of sound sources. Bass boost for close proximity virtual sources and high frequency filtering for sources located further away also requires digital processing of the B-Format signal. The implementation of these signal processing tasks would increase the overall calculation time needed to synthesize B-Format quite substantially. The delays incurred by this extra DSP would become significant on a lightweight processor, and would hinder a real-time system.

4.3 B-Format Reproduction over a Speaker Array

The decoding of four B-Format channels into a number of speaker feeds is performed for each speaker with the following equation detailed in Section 3.5.

$$F_i = \tfrac{1}{2} [\, G_1 \mathbf{W} + G_2 (\mathbf{X} \cos(\alpha) + \mathbf{Y} \cos(\beta) + \mathbf{Z} \cos(\gamma)) \,]$$

The coefficients for the X, Y and Z components are pre-calculated according to the position of each speaker, and the feeds F_i must be calculated in near real-time to minimize the delay of the audio path through the decoder. In the implementation of the spatialization system, fully-periphonic ambisonic surround sound is required, and we have opted to use a cuboid array of eight speakers. Therefore, eight speaker feed buffers must be derived from each four track B-Format buffer. The time to calculate eight speaker feed buffers of length 4K bytes from one B-Format buffer of length 16K bytes using the above formula was measured at 12.2 milliseconds on the Pentium 200MHz PC. A 3K buffer would cause a delay of 9 ms which has the same order of magnitude as the processing time for the 4K buffer. Even though the multiple buffering would constitute a large proportion of the delay in an audio application, in a standalone PC spatialization application the decoding delay would be significant when considered in addition to the other processing delays explained further in this section.

Multi-channel wave devices are usually configured under Windows 95 to be stereo devices, and not individual monaural wave outputs. For example, an 8-channel audio card exists in the Windows environment as four stereo devices. Therefore, stereo buffers have to be written to these devices. The eight speaker feed buffers derived above must be interleaved into four 8K stereo buffers, before they can be output to the audio card. The processing time on a Pentium 200MHz PC for the interleaving of eight 4K buffers into four 8K buffers was measured at 1.5 milliseconds. This delay is insignificant in relation to the other audio processing requirements.

4.4 Mixing and Manipulation of B-Format Signals

4.4.1 Mixing of B-Format Audio

Two B-Format signals are mixed by adding the individual components of each signal together. The X components are mixed together, the Y's together, etc. The mixing algorithm used is explained in the implementation chapter, and is reproduced here.

```
void Mix (short* a, short* b, short* mix, int bufsize)
{
        float scale_a = 0.5, scale_b=0.5, result;
        for (int i=0; i < bufsize/2; i++)
        {
                result = scale_a * float(A[i]) + scale_b * float(B[i]);
                // limit the sum before going from Fp to 16bit
                if (result < -32768.0f) result = -32768.0f;
                else if (result > 32767.0f) result = 32767.0f;
                mixedbuf[i] = short(result);
        }
}
```

The time taken for this algorithm to mix two 16K B-Format signals was measured to be 4.67 milliseconds on the PC. This is a computationally inexpensive operation. However, the computational delay is increased when more than two B-Format soundfields are mixed. The same code was modified to mix four and eight 16K B-Format buffers, and the results are shown below.

Number of 16K B-format buffers	Mixing Time (ms)
2	4.67
4	7.20
8	11.54

Figure 4.2 *Processing Delay on a 200 MHz PC for Mixing B-Format signals*

If many B-Format signals are to be mixed, the processing time required will increase accordingly. The execution time for mixing eight B-Format signals will add a detrimental delay to the obligatory real-time response of a spatialization application. The B-Format signals that require mixing would be stored on disk. There are disk access overheads and disk space problems that arise once more than 2 B-Format signals are being mixed. These can become an added burden on the processor of the machine.

4.4.2 Manipulation of B-Format

Audio processing can become very costly when performing real-time soundfield manipulations on a B-Format signal. The section of code shown below has the functionality for rotating, tilting, tumbling the soundfield, and changing the amplitude gains of the four components of the B-Format signal.

$$X' = \mathrm{gainX} \times (X \cos A \cos C - Y \sin A \sin C - Z \cos B \sin C)$$

$$Y' = \mathrm{gainY} \times (X \sin A \cos B + Y \cos A \cos B)$$

$$Z' = \mathrm{gain}\,Z \times (X \cos A \sin C - Y \sin A \sin C + Z \cos B \cos C$$

$$W' = \mathrm{gainW} \times W$$

A = angle of rotation B = angle of tilt C = angle of tumble

X′, Y′, Z′ and W′ must be calculated for every sample in each audio buffer. When the soundfield is rotated, tilted or tumbled (or any combination of the three), the values of A, B and C will change, and the resulting B-Format components will reflect that change. The gains labeled gainX, gainY, gainZ and gainW are the respective amplitude gains for each of the B-Format components. The user of the ambisonic system would have control of the manipulation angles and the four gains. The execution time for this code was timed, and resulted in a measurement of 20 milliseconds on the Pentium 200MHz for a 4K buffer. This processing time can increase the overall latency of a spatialization system considerably.

Any conceivable manipulation can be applied to a soundfield. Operations such as mirroring, distortion, stretching, translation and other geometrical transformations are possible on a B-Format soundfield. To make these options available to the user of a spatialization application, more processing would be incurred by the increased complexity of the soundfield transformation equations, and there will be a subsequent increase the processing demands of the application.

4.5 Resultant System Design

The results presented in this chapter highlight the main delays caused by the software processing involved in an Ambisonic system. The aim of the system described in Chapter 5 is to spatialize sound, with the aid of a suitable input device, so that the movements of the spatialized sound follow the movements of the input device in near real-time. Thus the audio source must be input, processed into a spatial form, any manipulations on that spatial sound must be calculated, and the spatial sound decoded and output to the speaker array. All of this should occur with a minimal delay so that the user perceives a real-time response.

The buffering of the audio input and output streams will introduce considerable delays into a real-time spatialization application to the extent that these delays are more significant than the total delay generated by the processing stages in the audio path presented above. The outcome is that these buffering delays completely limit the real-time response of the system. Because these delays can only be reduced within the limits explained earlier, it is important to minimize the processing at every possible stage in the audio path in order to realize a system that provides near real-time response. The real-time response of the system can only be reduced if these delays are diminished in some way. The use of smaller buffers than 4K has been discussed, and though this would be a theoretical solution, the overheads of the operating system and hardware would begin to have an impact as the buffer size decreases.

4.5.1 A Spatialization Application on a Stand-alone PC

A spatialization application would be entirely feasible on a standard 200 MHz standalone desktop computer. This application could provide basic functionality for the capture, spatialization and reproduction of ambisonic surround sound. However, the functionality and flexibility of the application while maintaining a real-time system is limited by the processor power available on the PC. The measured times of execution of the main processing parts of such a system could sum to a total latency of 50 to 60 milliseconds without taking the audio stream buffering into account. While this delay is not satisfactory in a real-time system, the delay perceptible to the user would not be serious. This matter is discussed in detail in Chapter 6.

The ambisonic encoding and decoding methods presented here and in Chapter 3 fulfill the requirements of a basic spatialization application. Although a PC-based spatialization application is possible using these simple formulas, the enhancement of these methods by incorporating additional digital audio processing would provide a system with the capability to create and reproduce increasingly realistic soundfields, and would be limited only by the amount of processing power available. There would not be much room for the growth of the system in terms of enhancing the quality of the spatialized audio. The audio quality necessitated in the professional audio industry has shifted from the quality used for Compact Disc (16-bit, 44.1 kHz) and Digital Audio Tape (16-bit, 48 kHz), to 24-bit audio sampled at 96 kHz. This has ramifications in the software audio processing environment, where desktop PC processing power and disk capacities are limited. The implementation of audio in a spatialization system of a higher quality than CD-quality is a natural extension. Three times the processing power and disk space would be needed for a system working with 24-bit 96kHz audio. Hence, many constraints may be imposed on a standalone PC-based spatialization system implementation.

4.5.2 A Distributed Approach to Developing a Spatialization Application

In the light of the findings presented in this chapter, our main objective was to implement a system that would handle any audio processing with the greatest possible efficiency. This was essential due to the limiting nature of the audio input and output buffering delays. An efficient way of handling audio processing is with a very fast processor. A Silicon Graphics Octane computer was available as well as an effectively dedicated fast network link, and this would provide the means to process data more efficiently than the PC. We therefore proposed a client-server approach to the spatialization application where the audio spatialization path is distributed between the Pentium 200 MHz PC (the Client) and the SGI Octane (the Server). An enhanced real-time spatialization system would be made possible by using this powerful processor for Ambisonic audio processing. The distribution of the spatialization audio path is described in Figure 4.3.

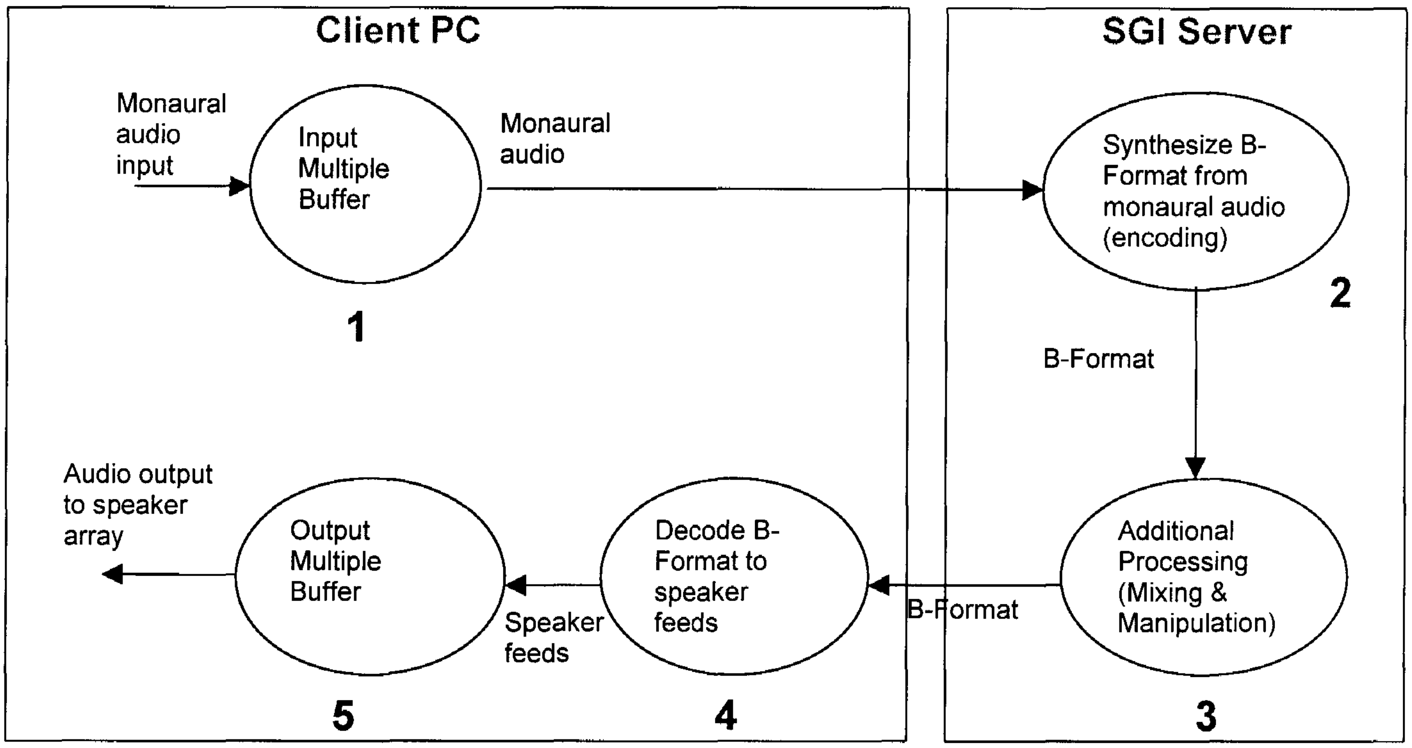

Client PC
SGI Server
Monaural audio input
Input Multiple Buffer
1
Monaural audio
Synthesize B-Format from monaural audio (encoding)
2
B-Format
Audio output to speaker array
Output Multiple Buffer
Speaker feeds
Decode B-Format to speaker feeds
B-Format
Additional Processing (Mixing & Manipulation)
5
4
3

Figure 4.3 *Data flow diagram of the audio path in an Ambisonic spatialization application*

4.5.2.1 The Client

It is necessary to first decide what parts of the system should remain on the client workstation. The spatialization environment runs on the client workstation. Therefore all interaction with the user should be implemented at the client workstation. This includes the Graphical User Interface for interactive spatialization, any audio input, which will be the audio sources used for spatialization, and the output of the spatialized sound over a speaker array. The audio input and output multiple buffering (sections 1 and 5 in Figure 4.3) reside as closely as possible to the physical audio devices, and so are also implemented on the client.

4.5.2.2 B-Format synthesis

Encoding a monaural audio signal into a surround sound B-Format signal is computationally inexpensive. This task can exist on either the client or server workstations. If we wish to place only the processor intensive computations on the server, the low cost B-Format synthesis can be placed on the PC immediately after the capture of the monaural audio. However, it would then be necessary to transfer all four audio channels of the synthesized B-Format signal over the network to the server machine for any post-processing of the B-Format signal. This would waste network bandwidth. Instead, a better solution is to place the B-Format synthesis on the server, so that only captured monaural audio stream exists on the network in the direction from client to server. An advantage of having the B-Format soundfield synthesis on the fast server is that soundfield creation can be enhanced with more intensive audio processing and a more realistic soundfield created. This can be achieved by introducing other directional cues such as early reflections, and ambient effects with digital reverberation. With a powerful processor, these tasks are not expensive to implement in real-time. The B-Format soundfield synthesis, shown as section 2 in Figure 4.3, has been placed on the server, and the information transferred from the client to the server via the network consists of the monaural audio information to be spatialized. A trivial amount of data depicting the desired 3D positioning of the monaural audio in the synthesized soundfield accompanies the monaural audio network transmission.

4.5.2.3 Additional Processing - Mixing and manipulation

When a B-Format soundfield is synthesized from a monaural audio source, it would be useful to keep a record of that spatialized version. This B-Format track can be stored on the server disk, and a spatialization system can provide facilities for layering (or over-dubbing) spatialized tracks. The nature of B-Format allows several B-Format signals to be down-mixed into one composite B-Format signal. This provides an ideal way of layering many spatialized tracks. The mixing process becomes costly as the number of tracks to be mixed increases. A user may want to spatialize 8 to 16 tracks and layer them over each other. These would be stored on the server disk, and the real-time mixing can be done efficiently by the server. For this reason, this B-Format mixing stage is placed on the server.

One method of layering B-Format tracks is to create a mix-down to a master B-Format track during the recording of each track. This will ensure that only 2 B-Format tracks need mixing, the master B-Format track, and the recording B-Format track. Those tracks that have already been layered would, by this time, exist in composite form as the master B-Format track. This would save much of the mixing delay, and one might consider allowing the client to perform this task. There is, however a disadvantage to this setup. Any spatialization information that has been mixed into the master track cannot be extracted. If an individual B-Format track that forms part of the composite is to be deleted, then the whole master track will have to be deleted. A solution to this problem is to store the individual B-Format tracks separately on the server disk, and when needed, mix these down into a composite B-Format track. By sacrificing a large amount of disk space, and some extra processing overhead, the mixing of many individual B-Format tracks can be done in real-time. This is only possible with an efficient processor, hence the placement of this functionality on the server machine.

Other processor intensive audio transformations are the various manipulations of a B-Format signal. These may include rotations, other geometric transformations, and also audio level adjustments that should be done in real-time. Because of the heavy processing required, these manipulations are applied to B-Format signal on the server machine. This processing would be done on the composite B-Format signal resulting from the previous processing stage. Both soundfield mixing and manipulation are grouped as processing stage 3 in Figure 4.3.

4.5.2.4 Decoding B-Format to Speaker feeds

The previous sections have grouped all of the processing mentioned on the server machine. The decoding of B-Format to a set of speaker feeds introduces a significant delay when running on a Pentium PC. This processing stage would be much more efficient if it were placed on the server side. A set of speaker feeds could then be transferred over the network back to the client machine for output. When considering the location of the B-Format decoding stage, the load on the network in the direction from the server to the client should also be carefully considered. As mentioned in Chapter 3, the B-Format encoded surround sound format does not assume anything about the reproduction array. The surround information may be transformed into a set of speaker feeds with appropriate decoding equations. When decoding B-Format to a set of speaker feeds, information about the configuration of the speaker array is used to derive the correct speaker feeds. It follows that the configuration of the speaker array at the client side must be decided upon, and an appropriate set of speaker feeds produced specifically for that setup. For example, the array might have four speakers and those speakers are placed at 45, 135, 225 and 315 degrees from centre front. If the speaker positions are changed, then the decoding code must be modified on the server for the new positions. However, the client workstation is the control platform for the whole system, and the speaker array also exists at the client side of the distributed system. Hence, the client should have complete control over the nature of the speaker array, and therefore the configuration of the B-Format decoder. The other reason why decoding on the server side is not optimal, is that the number of speaker feeds that must be transmitted over the network from the server to the client will vary with the number of speakers in the reproduction array. In placing the B-Format decoding on the Client PC side, the number of audio channels being sent back from the server is always four (the four components of a B-Format signal). Once they arrive at the client PC, they can be decoded to any configuration of speaker array. The processing required by the decoding process alone is easily manageable by a mid-range Pentium computer, and the delay caused by this processing is considered to be worth retaining the benefits of transferring B-Format data from the server instead of speaker feed data.

Summary

The spatialization system has been distributed in a way that provides fast processing of audio on the server, and also allows a constant amount of audio data to be carried by the network. As explained in Chapter 6, the network does not cause a large delay when using high-speed network hardware. This spatialization system could be setup to allow the spatialization application to work on a range of multimedia PC's. These would require specialized hardware, but the devices used in this implementation are fairly cheap, and perform well in the context of sound spatialization. The distribution some of the processing in the system provides a spatialization system that has the potential processing power at hand to generate, mix and manipulate very high quality soundfields in real-time. The system could be extended in terms of audio and spatialization quality, and the server is powerful enough to support these audio processing enhancements. As the server is essentially a centralized spatialization engine, the future of such a system might lie in a multi-user spatialization environment. A number of users would be able to log on and make use of the superior spatialization processing provided by the server.

The implications of the processing involved in a spatialization application have been explored within the contexts of stand-alone and distributed implementations. Even though a stand-alone PC-based system is practical, a distributed approach has been chosen for reasons explained in the above sections. The implementation of the client-server spatialization system is described in the next chapter.

Chapter 5

A Client-Server Architecture for Ambisonic Processing

In order to interact with a three-dimensional environment, the use of devices designed specifically for use in three-dimensional systems need is preferable. The Pegasus FreeD 3D mouse (Pegasus FreeD User Manual, 1996) is a suitable inexpensive peripheral which can be used in the development of an intuitive interface for 'with-height' surround sound interaction. The client-server implementation of a surround sound spatialization environment applies the ambisonic processing formulas detailed in Chapters 3 and 4, and the behaviour of various parts of the system is explained using the Ward and Mellor diagrammatic modeling toolset (Ward & Mellor, 1985). An overview of how to use the system to create a spatialized composition is presented, and then the suitability of various ambisonic surround sound delivery formats for music and video is discussed.

5.1 A Human Interface for Surround Sound

5.1.1 3D Input devices

Conventional mixing desks have been designed for panning audio in a stereo field, by means of, for example, rotational knob controls. When one requires the specification of the desired location of an audio signal in a three-dimensional space, these controls are no longer adequate. An intuitive way of providing three-dimensional coordinates is therefore necessary. With the adoption of the mouse in Graphical User Interfaces (GUI's) of windowed operating systems, a way of navigating fluidly in two dimensions was made possible. This device provides a suitable way of specifying two-dimensional coordinates for both graphics and audio in our case of sound placement in a surround sound environment. However when three-dimensional coordinates are to be captured, there is no satisfactory way of using the mouse on its own, or in combination with the keyboard to provide the means to navigate intuitively in three dimensions. Three-dimensional input devices are essential for a natural interface when controlling objects in a three-dimensional virtual environment.

There are several commercially available devices that can convey three-dimensional information in various ways. These include devices designed specifically for three-dimensional space use (such as the electromagnetic trackers made by Polhemus and Ascension) as well as devices aimed at the consumer entertainment market (for example, 3D mice and joysticks). While the Polhemus range provides accurate tracking and six degrees of freedom (three position and three orientation) the major drawback of using these devices in our application is their prohibitive price, even though their superior quality makes them the 3D input technology of choice in many virtual reality systems. The consumer range of '3D' input devices consists mainly of controllers that resemble ordinary joysticks, with enhanced features specifically designed for flight applications. Most of these devices are not capable of delivering three-dimensional coordinates.

However a low cost device manufactured by Pegasus Technologies does have the ability to capture three-dimensional coordinates with more than acceptable accuracy. The Pegasus 'FreeD' 3D wireless joystick was developed with the technology used to design the Mattel Powerglove in the early 1980's. The device consists of an 'L-frame' ultrasonic and infrared receiver unit, which is connected to the parallel port of the computer and is mounted on top of the monitor. The transmitter is wireless, and transmits positional information to the receiver with ultrasonic signals. Information about the status of the two buttons on the transmitter is conveyed by means of infrared signals.

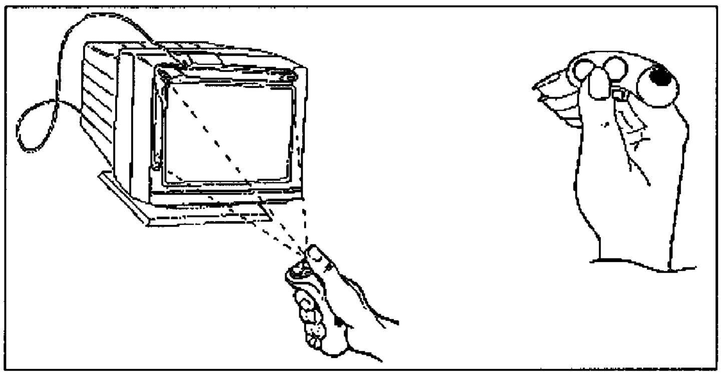
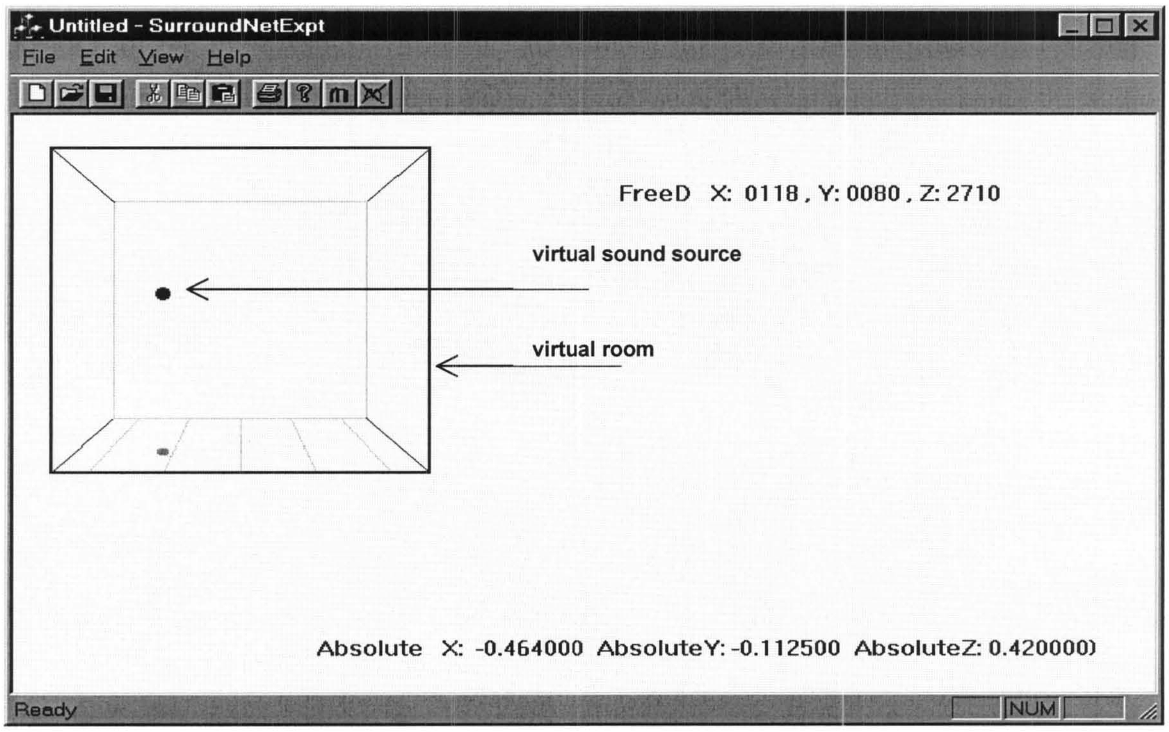


***Figure 5.1** The Pegasus FreeD wireless 3D joystick.*

The software packaged with the device includes a mouse driver for Windows 95 and a FreeD Dynamic Link Library (DLL). One may programmatically capture the current position of the 3D joystick with either the mouse driver or by using the DLL. Three-dimensional coordinates can then be attached to a section of an audio track and the audio can be spatialised with this information.

5.1.2 Graphical User Interface

A graphical user interface representation of the position of the 3D mouse (and therefore the position of the audio source) is helpful in consolidating the position of the sound source in the user's mind. Even if the soundfield synthesis and reproduction were perfect, because of natural confusions that occur in the human auditory system, a visual confirmation of actual position is important. A simple 3D perspective view of a room with a shaded circle representing the position of the 3D mouse (and the virtual sound source) was found to be adequate.



FreeD X: 0118, Y: 0080, Z: 2710

Absolute X: -0.464000 AbsoluteY: -0.112500 AbsoluteZ: 0.420000)

Figure 5.2 *Virtual room and sound source representation*

This representation gives the user a visual object to manipulate in order to control the audio source localisation. The interface also proved useful in the evaluation of the Ambisonic decoder implementation. One can locate the source visually, and compare the audio localisation with the position specified in the visual interface.

As well as a visual representation of the virtual room and sound source, the client human interface provides a sequencer-like interface that enables the user to build up compositions with many spatialised tracks by laying the tracks individually.

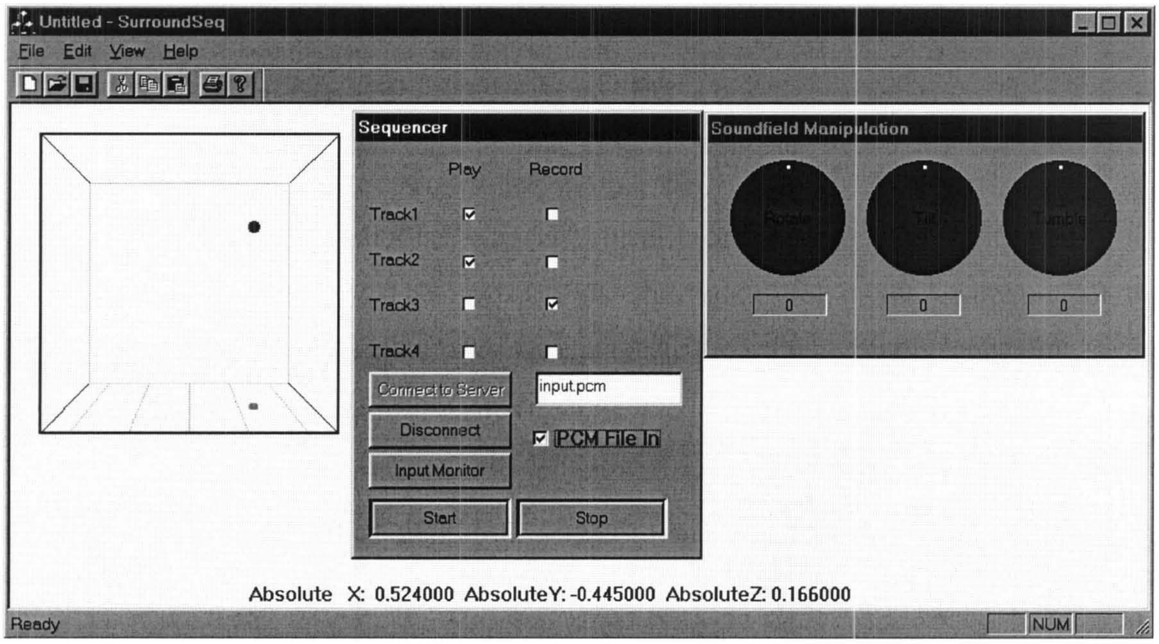


Figure 5.3 *Spatialisation Environment Human Interface*

Record and playback options for each track are available, as well as an 'Input Monitor' mode for basic monitoring of the audio input (after ambisonic processing) without recording any information. The audio input can be sourced either directly from the sound card input, or from a monaural audio file in raw PCM (Pulse Code Modulation) format. The Soundfield Manipulation dialog box contains knob controls for rotating, tilting and tumbling the complete soundfield.

5.2 Hardware Topology

A Silicon Graphics Octane with two R10000 CPUs running IRIX 6.4, with a SCSI hard disk drive was used as the spatialization server. This hardware provides processing power for the calculations involved in transforming monaural audio into ambisonic surround sound. These include processing the streaming audio into Ambisonic B-Format in real-time, the storage, retrieval and real-time mixing of spatialised tracks, and for other Digital Signal Processing (DSP) such as soundfield transformations, reverberation effects and early echo synthesis. Fast disk access and large amounts of disk space were also essential for efficient recording and playback, and the storage of spatialised audio. A network interface on the SGI Octane was present with both 10M and 100M Ethernet capability.

The workstation on which the client software was implemented was an Intel Pentium 200MHz PC with 64MB of RAM running Microsoft Windows 95. In order to implement a surround sound system, an audio interface for the PC with at least four monaural outputs was needed. Toward the beginning of this research only a handful of multi-channel audio devices were commercially available, including products like the eight-channel Digital Audio Labs V8. Many of these, though, were expensive. Event Electronics has recently shipped the Darla audio card (Darla User Manual, 1997), which was one of the first relatively inexpensive multi-channel audio cards available. It provides two analog inputs and eight analog outputs and is therefore well suited for the implementation of a surround sound creation and reproduction environment. The eight outputs of the Darla were output to an eight-speaker ambisonic reproduction array via an eight-channel power amplifier.

The network component consisted of a Cisco Catalyst 1700 switch to which both client and server workstations were connected by 10 Mbit/s or 100 Mbit/s Ethernet UTP (Unshielded Twisted Pair) cable.

5.3 Behavioural Specification

The system renders ambisonic surround sound from monaural audio data and associated three-dimensional coordinates. From this data, an ambisonic B-Format spatialised version of the monaural audio data may be encoded into ambisonic B-Format, and this B-Format data then decoded to produce eight speaker feeds, which are output to a speaker array. A simple data flow diagram of the system is shown below.

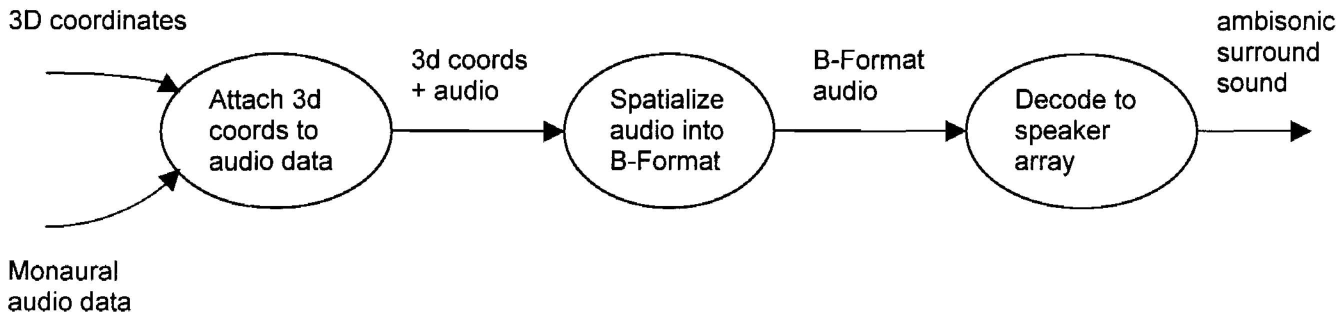


Figure 5.4 *General system data flow*

As mentioned in the previous chapter, the spatialization engine was transferred to a server machine that would be able to handle the costly processing requirements of ambisonic B-Format synthesis, and other digital audio processing.

This spatialization environment is intended to allow the user to spatialize sound in near real-time. When the 3D joystick is moved within a specific coordinate space, the sound should follow a corresponding path around the room. To this end, the spatialization processing should be carried out efficiently and the reproduction of the spatialised result should be output with minimum delay.

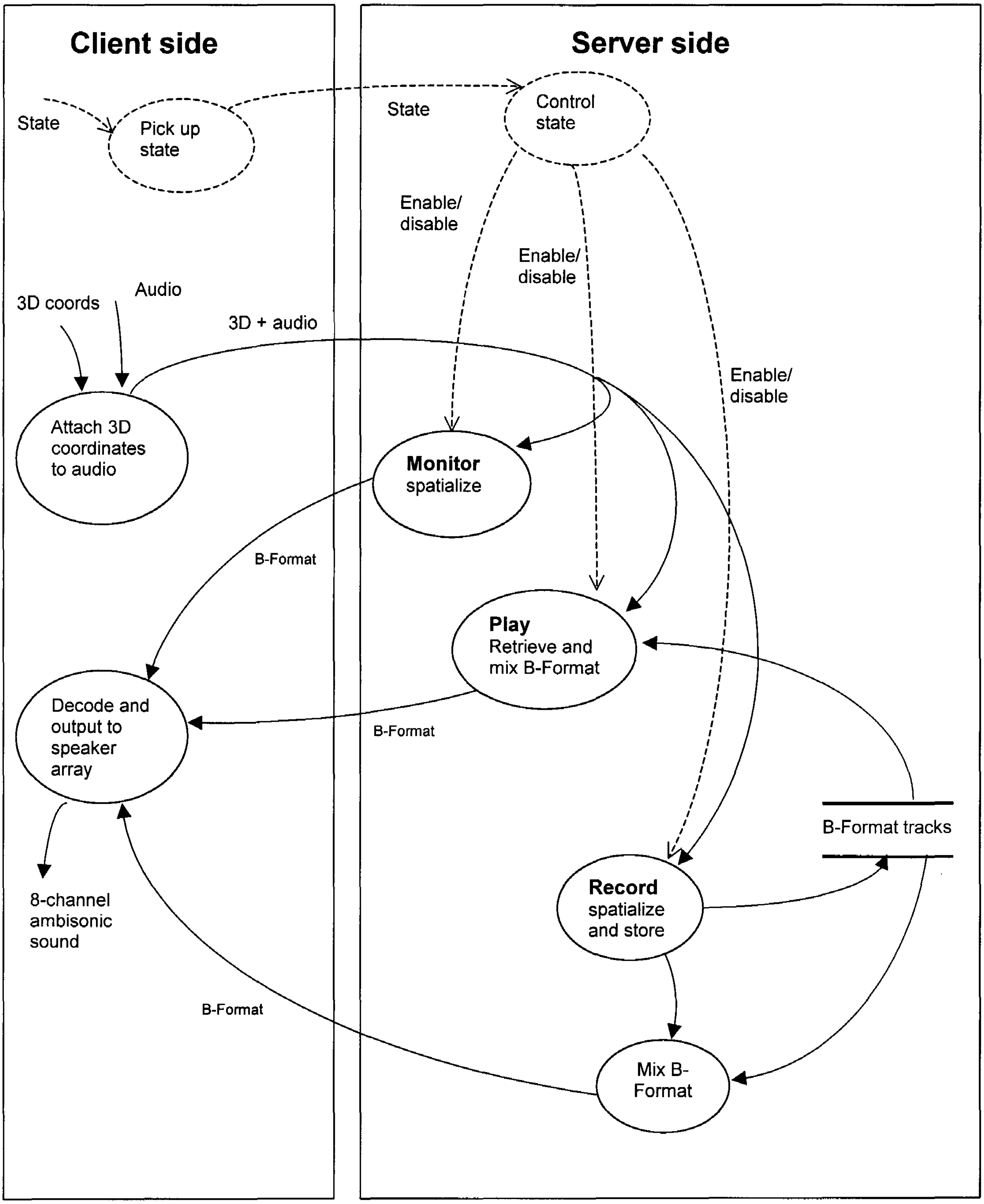


Figure 5.5 *System Behavioural Model*

The user is able to select one of three modes of using the system. The first is for real-time monitoring of audio source which is spatialized over the speaker array at a location corresponding to the position of the 3D mouse. The second and third are recording and playback functions for the storing and reproduction of spatialized audio. In the sequencer window of the human interface the user may specify the track to be recorded (none, if playback only is required), and which previously recorded tracks are to be played back. Here, any track being recorded is stored, and is mixed together with all of the tracks specified for playback. The resultant B-Format track is then rendered over the speaker array. A high-level behavioural model of the system utilizing the Ward and Mellor modeling tools (Ward & Mellor, 1985) is depicted in Figure 5.5.

The path of a monaural audio track that is simply encoded into Ambisonic B-Format and reproduced (for example, when monitoring) may be traced as follows. The client PC takes blocks of monaural audio data either as audio input from the audio card or from a PCM audio file stored on the PC. The current 3D coordinates of the 3D joystick are then attached to each audio block, and this data is packetized and transferred over the Ethernet network to the server machine. There, each block of audio data and its corresponding 3D data is encoded into a B-Format audio block (see Equation x), which consists of W, X, Y and Z components, each of which is equal in size to the original monaural audio data block. This B-Format data is returned to the client PC via the network. The B-Format version of the monaural data is then decoded on the PC (see Equation x) producing eight outputs which are fed to the speaker array via the audio card.

When recording or playing back spatialised tracks, the behaviour of the system is similar to monitoring, with some additions. These are: the storage of the B-Format spatialised track being recorded (if there is one), the retrieval of previously spatialised B-Format tracks, and the mixing of all the required tracks resulting in a composite B-Format audio track that is sent back to the client for rendering. This may be seen in more detail in Figure 5.6.

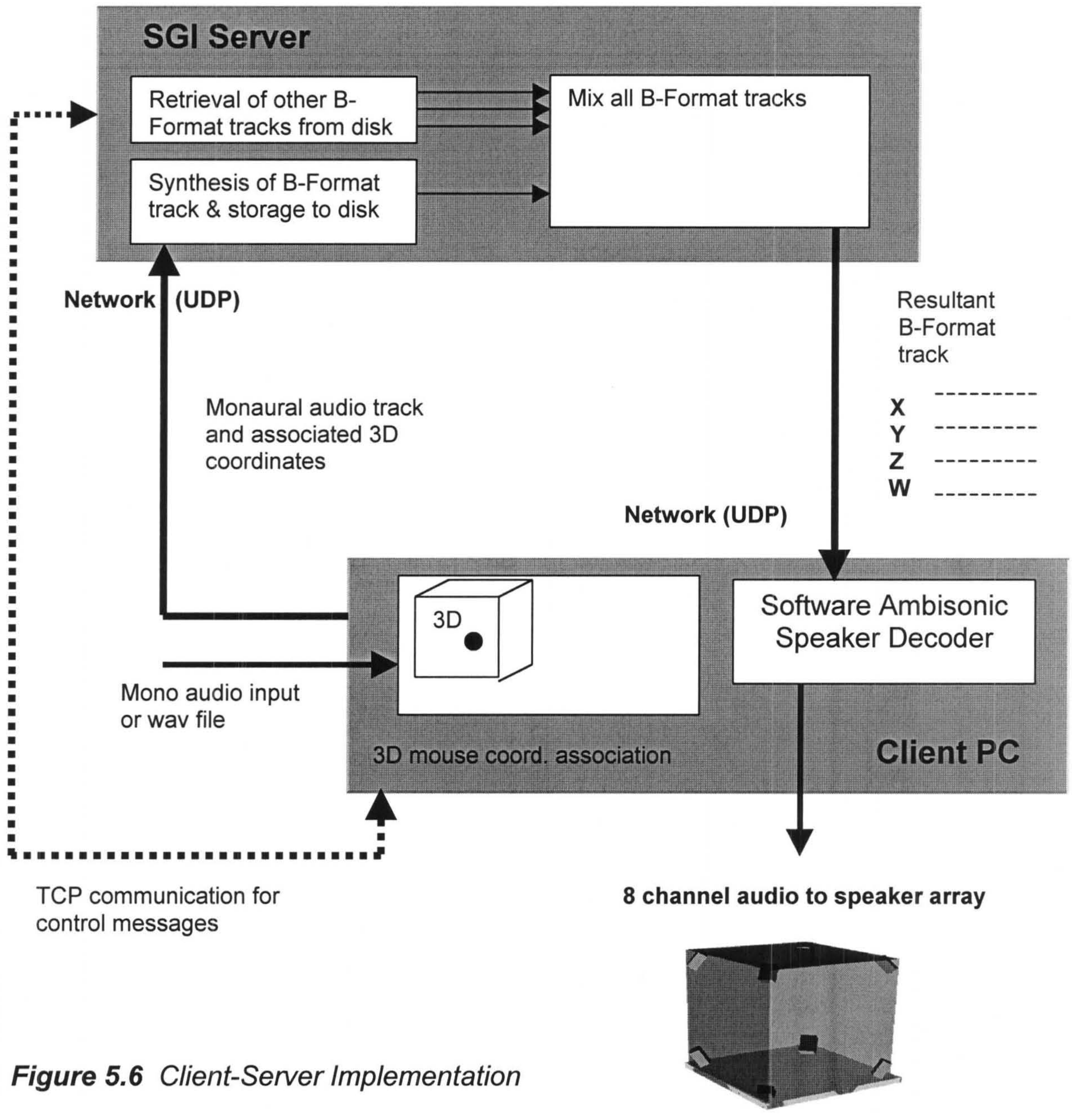

SGI Server
Retrieval of other B-Format tracks from disk
Mix all B-Format tracks
Synthesis of B-Format track & storage to disk
Network (UDP)
Resultant B-Format track
Monaural audio track and associated 3D coordinates
X
Y
Z
W
Network (UDP)
3D
Software Ambisonic Speaker Decoder
Mono audio input or wav file
3D mouse coord. association
Client PC
TCP communication for control messages
8 channel audio to speaker array

Figure 5.6 *Client-Server Implementation*

The client has two main responsibilities. The first is to capture monaural audio from the soundcard or audio file, and associated 3D coordinates from the 3D mouse. This information is sent over the network to the server for spatial processing. The other major task is the receipt of spatialized B-Format data from the network and the decoding of this spatialized audio information for reproduction over the speaker array. A behavioural model of the capture and transmission of 3D coordinates and audio data is given in figure 5.7.

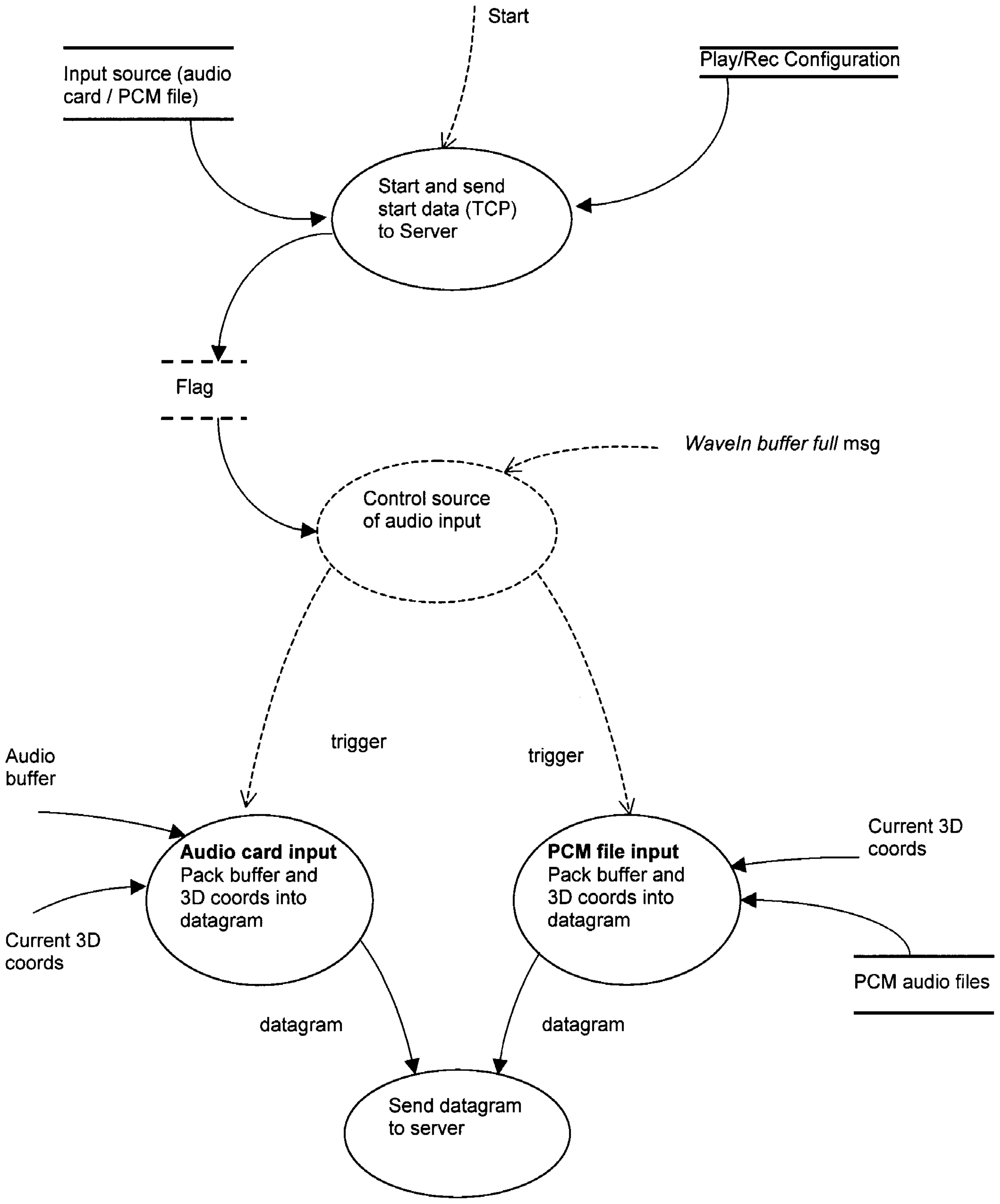
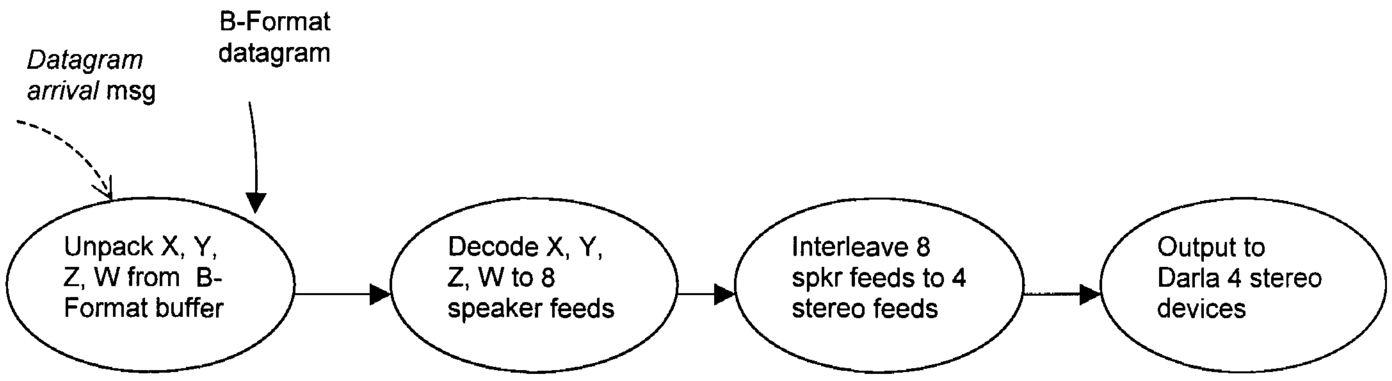

Start
Input source (audio card / PCM file)
Play/Rec Configuration
Start and send start data (TCP) to Server
Flag
WaveIn buffer full msg
Control source of audio input
trigger
trigger
Audio buffer
Audio card input Pack buffer and 3D coords into datagram
PCM file input Pack buffer and 3D coords into datagram
Current 3D coords
Current 3D coords
PCM audio files
datagram
datagram
Send datagram to server

Figure 5.7 *Client Software Behavioural Model – Capture and Transmission of Data*

When the spatialization process is started on the client side, the nature of the audio input source and the play/record configuration are determined from the controls on the sequencer dialog box. Information about the spatialization process is sent to the server, and a flag denoting the audio source will exist for the entire spatialization process. Whenever a full wave input buffer is available from the input device, the audio source flag is checked, and if the audio source is the soundcard, the audio data in that wave input buffer is sent with the current 3D mouse coordinates to the server. If the audio source is a file, a buffer is read from the file and the audio data in that buffer is packaged with the current 3D mouse coordinates and sent to the server for spatialization. In the latter case, it should be noted that the reception of a full wave input buffer is used as a trigger to retrieve an audio buffer from the audio file. The audio information contained in the wave input buffer is not used.



Datagram arrival msg
B-Format datagram
Unpack X, Y, Z, W from B-Format buffer
Decode X, Y, Z, W to 8 speaker feeds
Interleave 8 spkr feeds to 4 stereo feeds
Output to Darla 4 stereo devices

Figure 5.8 *Client Software Behavioural Model – Receipt and Reproduction of B-Format*

Figure 5.8 shows a behavioural model of the client implementation for the receipt and decoding of B-Format data for reproduction over a speaker array. Each B-Format datagram that arrives at the client is unpacked and processed into a speaker feed format suitable for output to the Event Darla audio card.

The client interface also provides knob controls for manipulation of the soundfield as a whole, such as rotation or tilting. The data generated by these controls is packaged with the audio data and 3d coordinates, and used for processing the soundfield data on the server side. This capability may be extended to allow for mirroring or distorting the soundfield.

5.4 Software Implementation

The functions of the client software have been covered broadly in the System Behavioural Specification. Here, a more detailed explanation of the workings of the server and client software is presented.

Connection-oriented protocols such as TCP are not suitable for continuous types of multimedia transmission (Fluckiger, 1995). The connectionless User Datagram Protocol (UDP) is therefore used for the streaming of audio data, and the connection-oriented Transmission Control Protocol (TCP) is used for communication between the client and server workstations.

5.4.1 Server Software

The software on the Silicon Graphics Industries (SGI) Octane was written in C++ and compiled on the Delta C++ compiler running under the SGI version of Unix, IRIX 6.4. Because the server software required two separate network protocols to handle audio data and control messages, and because of the easy implementation of parallel processes under IRIX, the server software was approached with a multi-processing solution in mind.

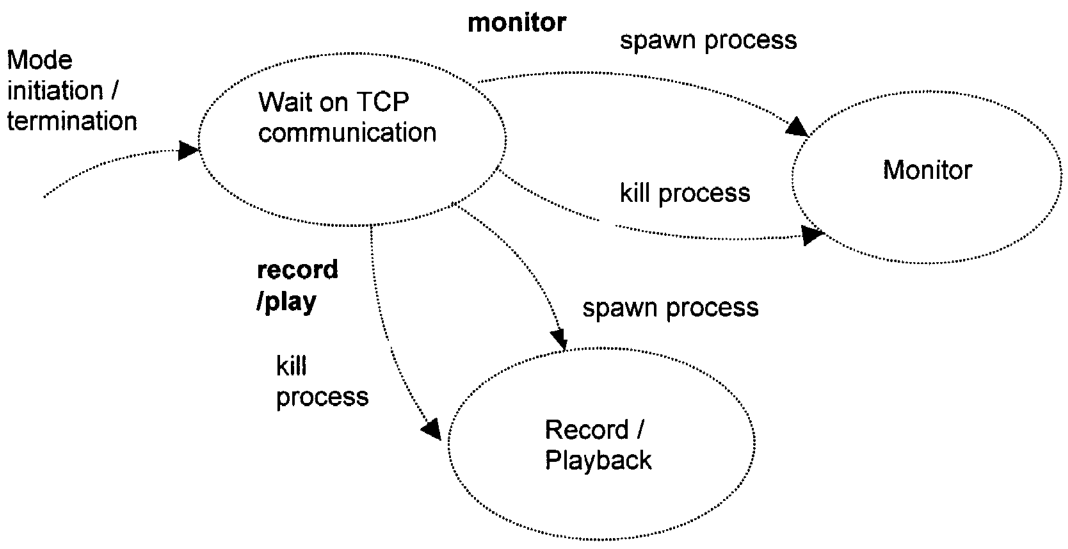


Figure 5.9 Server processes

Essentially, there are two main functions of the server software. The first is to receive control information from the client (TCP), specifying what mode of operation the user is initiating or terminating, and the configuration of that operation (in the case of initiation for example, which track to record and which ones to play back).

The second is to receive monaural audio data streaming from the client workstation (UDP), and process it in a manner which is dependent on the parameters of the TCP control messages. The process handling the TCP communication with the client is responsible for the creation of a parallel process that handles the reception, processing and transmission of UDP audio data.

The record and playback functions are combined into one routine because of various similarities such as the need to retrieve and mix previously recorded B-Format files. Once a TCP connection is established between the client and server, the server will wait for initiation of either the monitor mode or the record/playback mode. The main process spawns a child process for the appropriate mode required by the client. The child process will then accept UDP data from the network and process it according to the mode and configuration specified. When a termination message is received from the client, the parent process kills the currently active child process. In the case of the record/playback mode, files for recording and playback are opened before the child process is spawned, and are closed after the child process has been killed.

A flowchart of the monitoring process is given in detail below.

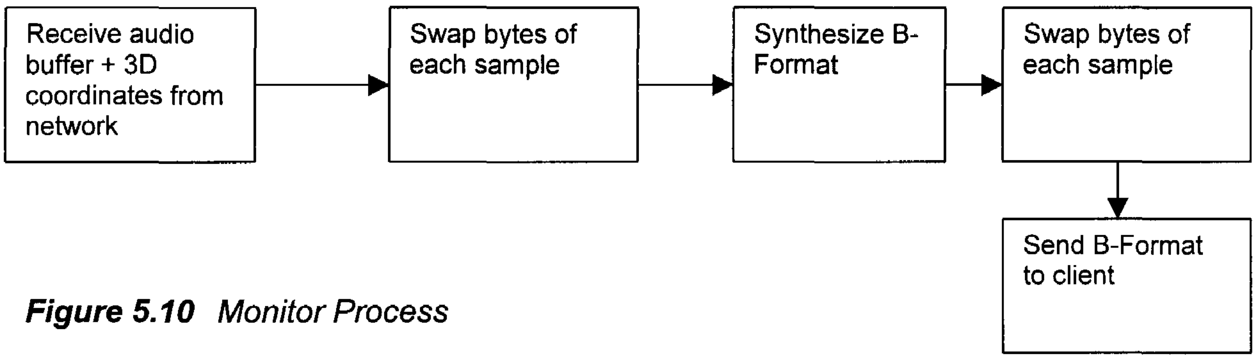

Receive audio buffer + 3D coordinates from network
Swap bytes of each sample
Synthesize B-Format
Swap bytes of each sample
Send B-Format to client

Figure 5.10 *Monitor Process*

Intel-based computers store their data in 'little-Endian' byte order, meaning that the most significant byte is stored as the right-hand byte of a word (Comer & Stevens, 1991). This is the other way around to the network standard 'big-Endian' byte order, which is also the byte order used by SGI machines. It is for this reason that when an audio buffer is transferred from PC to an SGI machine over a network, the two bytes of every 16-bit sample contained in the buffer need to be swapped. This work is done on the powerful CPU of the SGI machine.

The B-Format synthesis process currently implemented is a simple one, only applying the equations suggested by Malham (1995) in section 3.4. More realistic B-Format synthesis would require the inclusion of early echoes, high frequency air absorption simulation, and Doppler effect simulation for rapidly moving sound sources.

In recording mode, a B-Format version of the incoming monaural audio data is synthesized, and stored on the server disk. Any playback tracks that have been specified in the recording configuration are retrieved and mixed with this new spatialized track in real-time, and the composite B-Format data returned to the client for reproduction. In playback mode, the server also receives audio data blocks from the client, but instead of using these for spatialization, the interval at which they arrive is used as timing information to trigger the retrieval of audio data buffers from playback tracks already stored on the server disk. These tracks are mixed into a composite B-Format track and sent back to the client. All B-Format tracks are stored as raw four-channel PCM (Pulse Code Modulation) files. Figure 5.11 shows a flowchart of the record and playback processes.

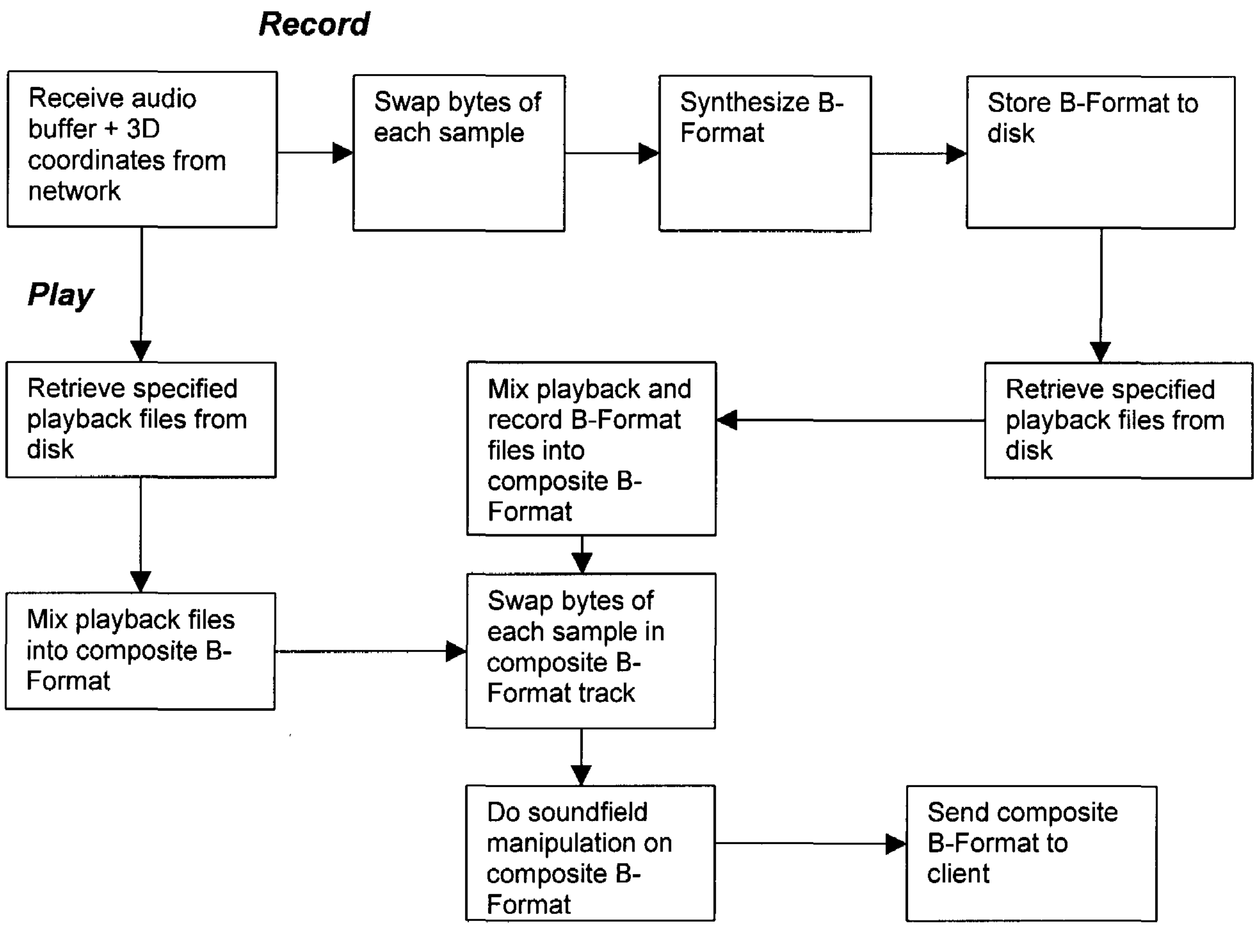

Record
Receive audio buffer + 3D coordinates from network
Swap bytes of each sample
Synthesize B-Format
Store B-Format to disk
Play
Retrieve specified playback files from disk
Mix playback and record B-Format files into composite B-Format
Retrieve specified playback files from disk
Mix playback files into composite B-Format
Swap bytes of each sample in composite B-Format track
Do soundfield manipulation on composite B-Format
Send composite B-Format to client

Figure 5.11 *Record/Playback Processes*

The mixing algorithm used is the one suggested by Cook [9].

```
void Mix (short* a, short* b, short* mix, int bufsize)
{
        float scale_a = 0.5, scale_b=0.5, result;

        for (int i=0; i < bufsize/2; i++)
        {
                result = scale_a * float(A[i])  + scale_b * float(B[i]);
                // limit the sum before going from Fp to 16bit
                if (result < -32768.0f) result = -32768.0f;
                else if (result > 32767.0f) result = 32767.0f;
                mixedbuf[i] = short(result);
        }
}
```

If two 16-bit signals, **a** and **b** of type 'short' are mixed by simply summing them, overflowing the 16 bits is a possibility, resulting in significant discontinuities in the signal. For example, if **a** = 32767 and **b** = 1, then **a** + **b** = 32768, which in a two's-complement short data representation would wrap around to –32768. In the above algorithm, overflow as a result of the sum of two signals going out of range is guaranteed not to occur, by scaling each input signal and limiting the result of the summation. Floating-point calculations are used because floating-point multiply operations are generally much faster than integer multiply operations on the SGI Octane CPU [9].

Once the composite B-Format data has been calculated, manipulation on the complete soundfield can be done. Examples of these transformations may be found in Chapter 3. Extra data from the knob controls in the client human interface is transmitted with the audio data and 3D coordinates to the server and allows the rotation, tilting and tumbling of the complete soundfield. In the current implementation this information is not recorded.

5.4.2 Client Software

The client software was written in Microsoft Visual C++ version 5.0 under Microsoft Windows 95. The implementation used the framework generated by the Microsoft Foundation Class Application Wizard in the Visual C++ environment. This framework provides the means to facilitate programming using Windows event-response mechanisms by structuring the message-handling process (Kruglinski, 1996).

The Win32 low-level audio functions were used in programming waveform audio input and output. The Win32 low-level audio functions require that each audio buffer (for wave input or output) is first prepared and then sent to the wave device driver. When the device driver returns the buffer to the application, the buffer must be unprepared. If it is to be used again, the buffer must be re-prepared.

Before any spatialization can take place, the input from the 3D mouse and the corresponding 3D GUI are enabled, and a TCP connection is established between the client and the server. The user is then able to commence operation in any one of the three spatialization modes using the controls provided in the user interface dialog boxes. The initiation of spatialization in monitoring, playback or recording mode causes a TCP message containing the mode of spatialization and the configuration data (in the case of record and playback modes) to be sent to the server. This notifies the server to prepare to receive and process datagrams containing 3D coordinates and monaural audio data. The source of the monaural audio data can either be the input of the audio card, or a PCM audio file residing on the disk of the client.

A message handler function exists for handling wave input. In this implementation, the only wave input message of interest is the **MM_WIM_DATA** message, which is posted whenever a wave input buffer has been filled. When this message is posted, the filled wave input buffer is also returned. If a **MM_WIM_DATA** message is posted, the audio from the wave input or PCM file (depending on the status of the audio source checkbox) is packaged together with the current coordinates of the 3D mouse into a datagram buffer. Any other data need by the server for processing, such as soundfield rotation angles, is included in this datagram buffer, and the datagram is then sent to the server.

If a PCM file is specified as the input source, the occurrence of **MM_WIM_DATA** messages is used as timing information to trigger the retrieval of an audio buffer from the PCM file. It is the audio information in this buffer that is put into the datagram, and the audio information contained in the buffer that is returned to the application with the **MM_WIM_DATA** message is discarded.

The other main section of the client software will receive data from the server and reproduce it over the speaker array. The receiving UDP socket is configured in a non-blocking mode. When a B-Format datagram arrives at the socket, a user-defined *Datagram arrival* message is posted to the application. The datagram is disassembled into the four components making up the B-Format signal and these are used to derive eight audio signals to feed each of the eight speakers in the reproduction array (see equations in Chapter 3). The device driver for the Event Darla audio card makes the audio hardware visible to Windows 95 as four stereo devices. The eight speaker feeds have to be interleaved into four stereo speaker feeds, and these are output to the four stereo devices.

Multiple buffering is required on both waveform input and output to ensure an uninterrupted stream of audio. It is necessary to use multiple buffering on the input in order to avoid audio data being lost during capture. If only one buffer were used, a loss of audio information would occur during the time that the buffer is returned to the application, unprepared, re-prepared, and sent back to the input device.

When multiple buffering the input stream, a number of empty buffers are sent to the wave device before the start of wave input. The input is started, and when a buffer is filled, it is returned to the application, and one of the other buffers replaces the first immediately. The first buffer is processed, and an empty buffer is sent to the wave input device. In this way, there is a constant pool of empty buffers ready for filling by the wave input device.

The procedure is similar with waveform output. Because of the nature of small but unknown delay times throughout the spatialization system, multiple buffering of the waveform output is of paramount importance in achieving a continuous audio output stream. The multiple buffering on the wave output works in the following way: A number of buffers are accumulated before starting to write to the wave out device, the wave output is started, and all of the buffers will be sent to the device driver for output. As more buffers arrive for output, they are written to the wave out device driver. The device driver will therefore always have at least one buffer to output, and an uninterrupted audio output stream will result.

The audio format used throughout the implementation is 16-bit PCM-encoded audio sampled at 44.1kHz. An audio buffer size of 4K bytes was used, and multiple buffering of three such buffers was found to produce reliable uninterrupted audio input and output streams.

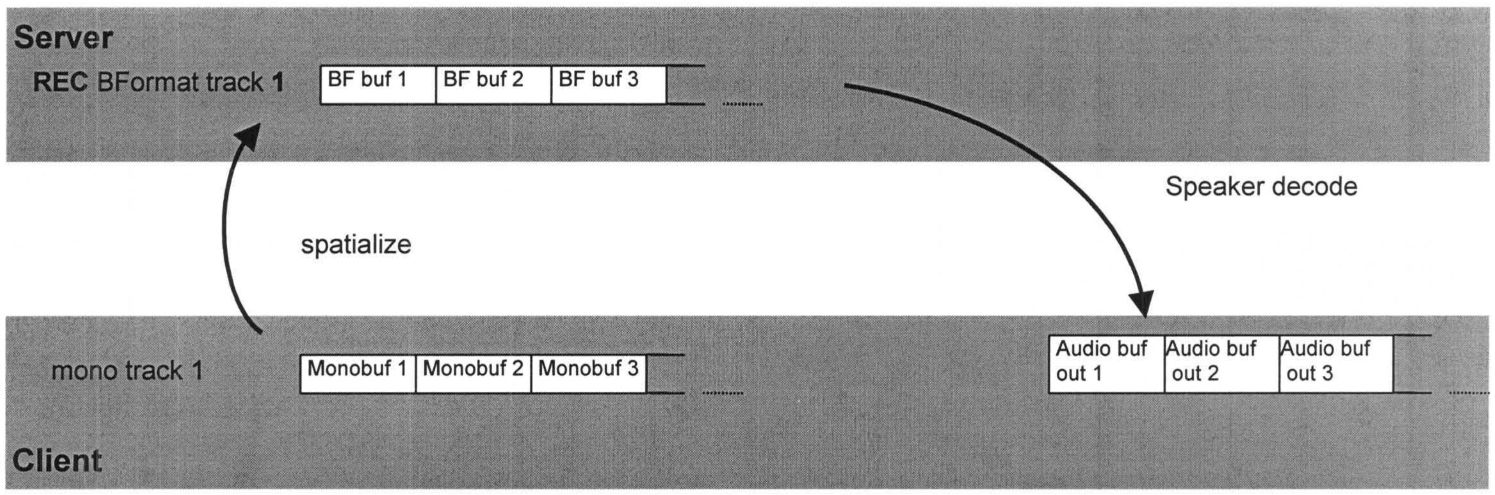
5.5 Creating a Surround Sound Composition

A spatialized composition may be created by deriving spatialized versions of individual audio sources, and layering them. The optimal method of achieving this with the system is to start with a composition that already exists as a number of monaural audio files on the client disk. In this implementation, the files would have to be in 'raw' 16-bit monaural PCM format. PCM files are similar to wave files, but contain only audio data, and no header information, such as the sampling rate or sample size of the audio data.

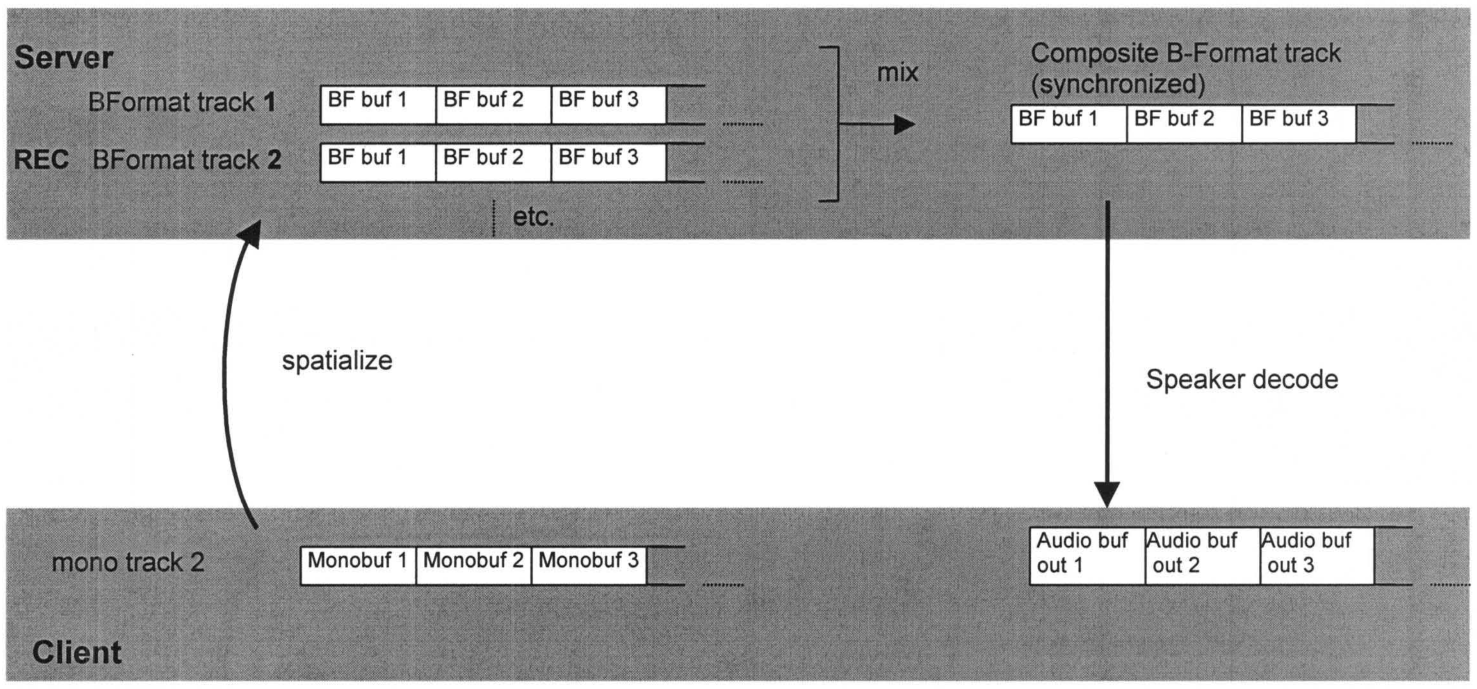
As many computer music studios use MIDI sequencers with MIDI instruments such as sound modules and samplers for the production of music, the outputs of such devices must be mixed and stored as audio files on hard disk. This is possible with modern commercial multi-channel digital audio cards such as the Event Darla and Layla cards [10]. All MIDI instrument outputs can be input into a multi-channel audio device on a music workstation, and digitally recorded to hard disk using a program like Steinberg's Cubase VST [11] or Cool Edit Pro from Syntrillium [12].

Once four monaural PCM tracks exist on the client, the spatialization of each one in turn can be recorded on the server disk. As mentioned earlier, while one is recording a spatialization, one can optionally hear any of the previously recorded spatialized tracks simultaneously, thus providing the 'sound-on-sound' capability found on multi-channel tape recorders and in MIDI sequencers.

Synchronization of spatialized tracks is maintained when recording and/or playing provided that the original monaural audio tracks are first synchronized with each other. This is illustrated in the following diagram.



Recording spatialisation of track 1



Recording spatialisation of track 2

Figure 5.12 *Synchronization of spatialized tracks*

The synchronization of the original monaural tracks on the client can be achieved in a multi-track wave editor such as Cool Edit Pro. Occasionally a datagram may be lost in the transfer of audio across the network. If this happens during playback only, datagrams audio buffers can only be lost on the path from the server to client. Because these buffers are the mixed result of several tracks, the same amount of information would effectively be lost from all tracks. A glitch in the audio stream is noticeable, but the synchronization between spatialized tracks is still retained. A problem will occur, though, if datagrams are lost in the process of *recording* spatializations. For example, while recording track 1, the 58th datagram might be lost en route to the server, and so the 58th audio buffer will be lost. Then while recording track 2, no datagrams (therefore no audio information) may be lost. In this case, during the playback of these two tracks, track 1 will have buffer 58 missing, and the audio buffers 59 onward of track 1 will be out of sync with those of track 2, by the length of one audio buffer. If we use 16-bit audio buffers of length 4K sampled at 44.1kHz, this would mean a discrepancy of 2048 samples, or 46 ms, a noticeable difference, especially as this effect would be cumulative if more than one datagram were lost.

It would be preferable in future to implement a robust synchronization scheme, perhaps using a time-stamping system such as SMPTE (Society for Motion Picture and Television Engineers) time code. This code consists of time stamps in the format **hrs:min:sec:frames**, and is used widely in music and video production and post-production. Each track would have SMPTE time codes throughout their length at various intervals, and the system would be responsible for generating a master SMPTE clock for reference, as well as conforming the recording or playback of each track to that clock.

It is usually necessary to plan spatial movements for each track of a composition before the spatialization is done. With most compositions, the paths of spatialization will be complex enough to forget the original path halfway through the spatialization process. A hand-drawn storyboard depicting the main parts of a musical piece with the desired spatial positions of each track is extremely useful when spatializing a multi-track piece.

5.6 Delivery of Surround Sound Compositions

The ambisonic surround sound delivery format (B-Format) used in this research was chosen for the reasons mentioned in chapters 2 and 3. However this route was taken in the context of the surround sound compositional environment that the system provides.

In a computer music studio environment it is assumed that bandwidth between workstations and disk space on those workstations are abundant, and so using B-Format as a studio quality surround sound format for the transferal and storage of ambisonic audio is a viable option. The benefits of retaining surround sound audio in ambisonic B-Format covered in chapter 3 are also evident. However if high resolution audio is required, and as many lower-end systems have bandwidth or disk space restrictions, the handling of files of this nature would soon become cumbersome. The lossless compression of B-Format (explained in section 3.6) would be an ideal solution to this problem. Lossy compression techniques are employed in many multimedia delivery mechanisms. The effect of lossy compression on B-Format is not known, and in studio quality applications is an undesirable loss in absolute sound quality.

If reproduction of surround sound without decoding is required, the most obvious solution is to deliver to the destination the actual speaker feeds derived from a surround format such as B-Format. The disadvantages of this approach are that the configuration of the end-user reproduction environment must be assumed. This involves pre-determining the number of speakers present, and exactly where the speakers will be placed.

Several delivery methods designed specifically for ambisonic material have been mentioned in chapter 3, which included UHJ and G-Format. UHJ encoding has many advantages such as excellent mono and stereo compatibility, but has the disadvantage of needing decoding for extraction of the surround information, and only being able to hold surround information for planar surround sound. Many surround sound systems are home theatre systems using 5.1 surround technology. To make ambisonic material available in this format, Ambisonic G-Format was developed. In G-Format the speaker feeds are calculated according to an assumption of the typical placement of 5.1 systems' speakers, and encoded into a digital surround format such as Dolby AC-3.

The decoding of ambisonic signals upon delivery in the home theatre environment requires extra hardware to realize the complete soundfield. This hardware decoder would be, for example, a Dolby 5.1 surround decoder, or a UHJ decoder. In software environments, the decoding of ambisonic signals, for example B-Format or UHJ, is less problematic, due to the existence of adequate software decoding methods and DSP power as is exhibited in this thesis.

When the need arises to integrate ambisonic sound into other media such as digital video, especially in a distributed multimedia environment, several options are available. One possibility could be the inclusion of 3 or 4-channel B-Format in a commonly used multimedia file type such as Microsoft's AVI (Audio/Video Interleaved) or MPEG file format. The Microsoft AVI multimedia file format caters for one video stream and as many audio streams as are needed. This provision allows even 4-channel 'with-height' ambisonic sound to be incorporated into an AVI file containing video information. Unfortunately AVI files do not contain time-stamping information, and in a production environment where many types of media are integrated and edited digitally, this is a serious flaw. This deficiency would warrant the use of a more suitable file type, for example, Microsoft's new Active Streaming Format (ASF) [13] which resembles the AVI format, but has the capacity to handle time-stamping.

Summary

The implementation of the distributed spatialization system proposed in Chapter 4 has been detailed. An intuitive human interface for spatializing sound was developed on the client workstation using the inexpensive Pegasus FreeD 3D mouse with an accompanying simple 3D GUI. The behaviour of both client and server workstations is illustrated using the Ward and Mellor modeling tools (Ward & Mellor, 1985), and the implementation details of the system are then explored.

Chapter 6

System Performance Issues

During the development of the system the primary focus was on real-time interactivity, which concerns the latency introduced in the various sections of the audio path. The implementation of a real-time audio system looks towards minimizing any delays that may occur. These include the delays caused by processing digital audio, the transmission of audio data over a network, the peripheral hardware of the machine (for example the audio card) and delays inherent in the operating system under which the application is running. The entire distributed spatialization application described in the previous chapter has been carefully investigated in order to locate the nature, magnitude, and the sources of any network or processing delays occurring throughout the system.

The system performance is also investigated in qualitative terms, such as the user's perception of the near real-time response of the spatialization application. The realism of the soundfield recreated by the reproduction of B-Format soundfields over a speaker array is evaluated, and the musical composition implications arising out of 3D compositional environments are explored.

6.1 Audio Processing and Network Performance

If the spatialization system were analyzed in its fully developed form, which allows layering of tracks and the real-time mixing of these tracks, the task of isolating bottlenecks would be arduous. In a complex application the effect of simultaneous reading and writing of multiple audio files on the server disk is difficult to measure, and influences the network and processing performance of the system in an unpredictable manner. The in-depth analysis of either the operating systems, the hardware of the SGI Octane, or the Pentium PC used in the implementation, and any subsequent evaluation, is beyond the scope of this thesis. The measurements presented in this section are made solely for the evaluation of the performance of the spatialization system.

The system performance experiments were therefore conducted with a simplified version of the spatialization system. This version captures monaural audio and associated 3D coordinates at the client workstation and sends this data to the server. A B-Format version is created from this data and the four resulting audio channels are sent back to the client where they are decoded into eight speaker feeds, and reproduced.

6.1.1 Audio Processing Performance

Figure 6.1 shows a breakdown of the audio path through the system, highlighting the main functional sections. The sections of the system that are expected to cause delays in the audio path due to audio processing (including the multiple buffering on the PC) are numbered from 1 to 7. The performance of the network between the client and server is evaluated by the delays and the variation in the delays measured between point A and point B in Figure 6.1.

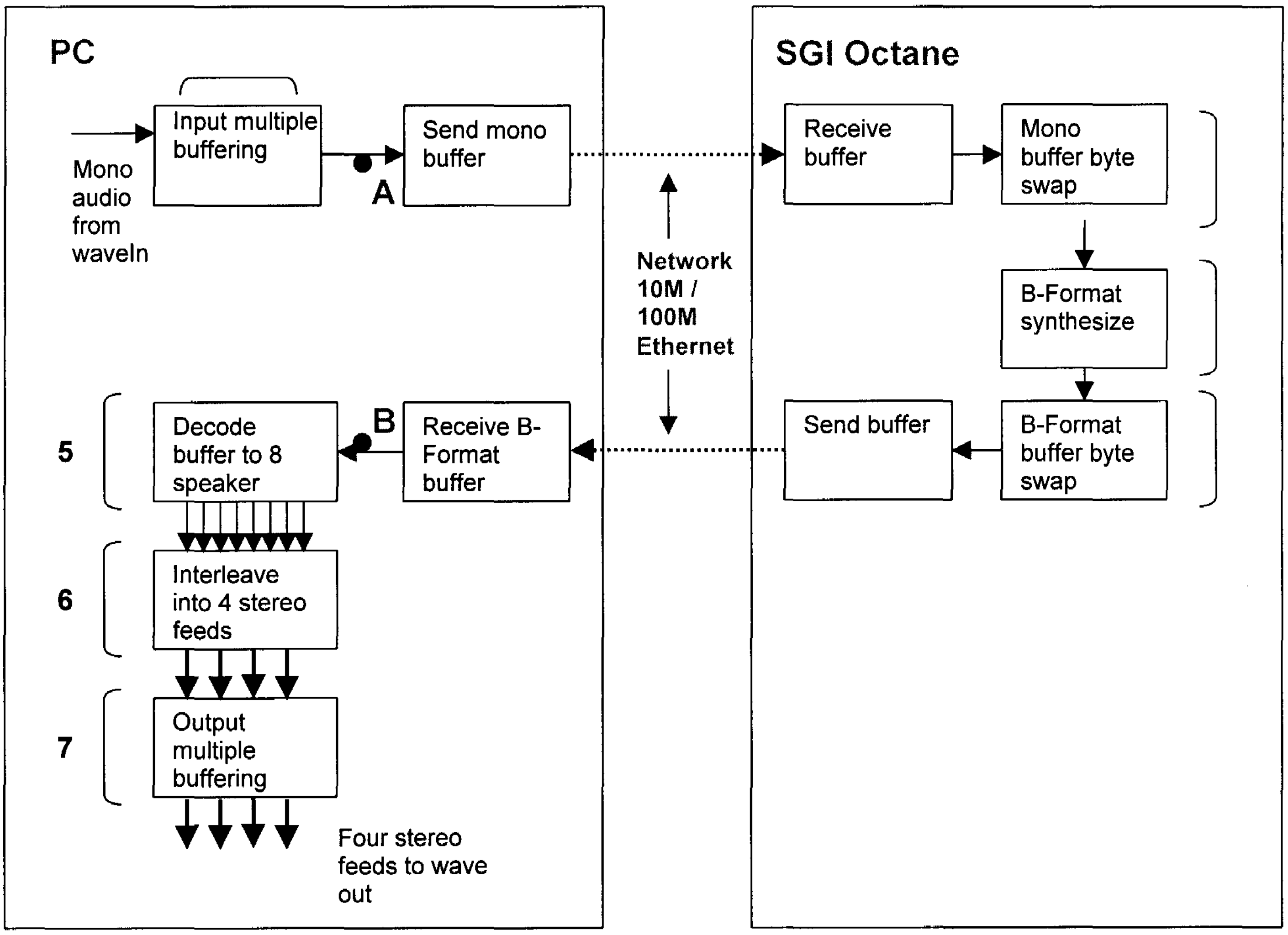


Figure 6.1 *Audio path in a simplified spatialization implementation*

The path of audio in the system is as follows: A 4K monaural audio buffer is captured with a set of 3D coordinates, and this data is packaged and sent over the network to the server. The words in this buffer have their bytes swapped into the server's byte order. A 16K B-Format buffer is created from the mono buffer and the 3D coordinates, and this 16K buffer has all its words changed back into the PC's byte order. This 16K buffer is sent back to the client, where it is first decoded into 8 4K speaker feeds. Then the speaker feeds are interleaved to create 4 8K stereo buffers, which are sent to the output wave device.

Processing times for sections 2, 3 and 4 in Figure 6.1 were measured on the server machine, and those for sections 5 and 6 were measured on the client platform. In order to measure the execution time of a certain section of code, a simple method was used where the code is looped a large number of times, typically 10000 or 100000 times. The time taken to loop the section of code can be timed manually, and the total number of seconds elapsed for the looped code to execute is divided by the number of iterations of the loop. This provides a fairly accurate measurement of the time that would elapse for one iteration of the loop, and hence a measurement of the execution time of the section of code in question. It should be noted that these times have been measured in isolation from any other overheads on either machine, for example, the interference of Graphical User Interfaces and continuous disk access. The results of these timings are shown below.

Silicon Graphics Audio Processing Functions	**Measured Time (ms)**
Mono buffer byte swap	0.30
Synthesis of B-Format	0.84
B-Format buffer byte swap	0.93
Total	2.07
Pentium 200 MHz PC Audio Processing Functions	**Measured Time (ms)**
Decode B-Format buffer to 8 speaker feeds	12.20
Interleaving speaker feed buffers into 4 stereo buffers	1.50
Total	13.70

Figure 6.2 *Audio Processing Calculation Times*

These results are fairly insignificant if viewed in isolation. If they are considered in addition to delays caused by the network and wave input and output buffering (explained in the following sections) the total delay becomes significant.

6.1.2 Network Performance

The transportation of the audio over the network between the client and server also causes latency. The two capacities of networks employed were 10 Mbit/s and 100Mbit/s Ethernet. The physical connection was Unshielded Twisted Pair (UTP) cable, and the two workstations were connected via a Cisco Catalyst 1700 switch. The experiments were conducted with a relatively small amount of other network traffic being handled by the switch. The connection between the server and client workstations was effectively a dedicated link.

The delay incurred in the system between points A and B in figure 6.1 represents the total network delay including the processing time on the server. The latency in the system due to the network between points A and B was investigated by measuring the elapsed time between the sending of a 4K audio buffer from the client to the server, and the return of the resulting 16K spatialized buffer. In effect, there are always five monaural audio streams existing on the network at any time.

The windows multimedia timer function timeGetTime() was used, which returns the system time of the PC to a resolution of approximately 1 millisecond. A 'buffer round trip time' is the delay between sending an audio buffer to the server, and the reception of the spatialized version of that buffer at the client. Each buffer was tagged with a unique integer identifier, and this identifier was copied into the spatialized buffer on the server. A returning buffer could then be recognized, and buffer round-trip time calculated from the difference in the PC system time between the sending and receiving of buffer. 1000 consecutive buffer round trips times were measured per experiment. Graph plots of this data were generated to visualize the magnitude and variation of the buffer round-trip delays. The round trip times measured here are done so during the execution of the simplified spatialization application. They will therefore not necessarily reflect completely accurately the actual delay over the network, but the delay caused by the transfer of data over the network as part of the application. This is because other internal workings of the application will affect the measured round-trip times.

6.1.2.1 10Mbit/s Ethernet Experiments

An example of the data collected from one of the 10 Mbit/s experiments is shown below.

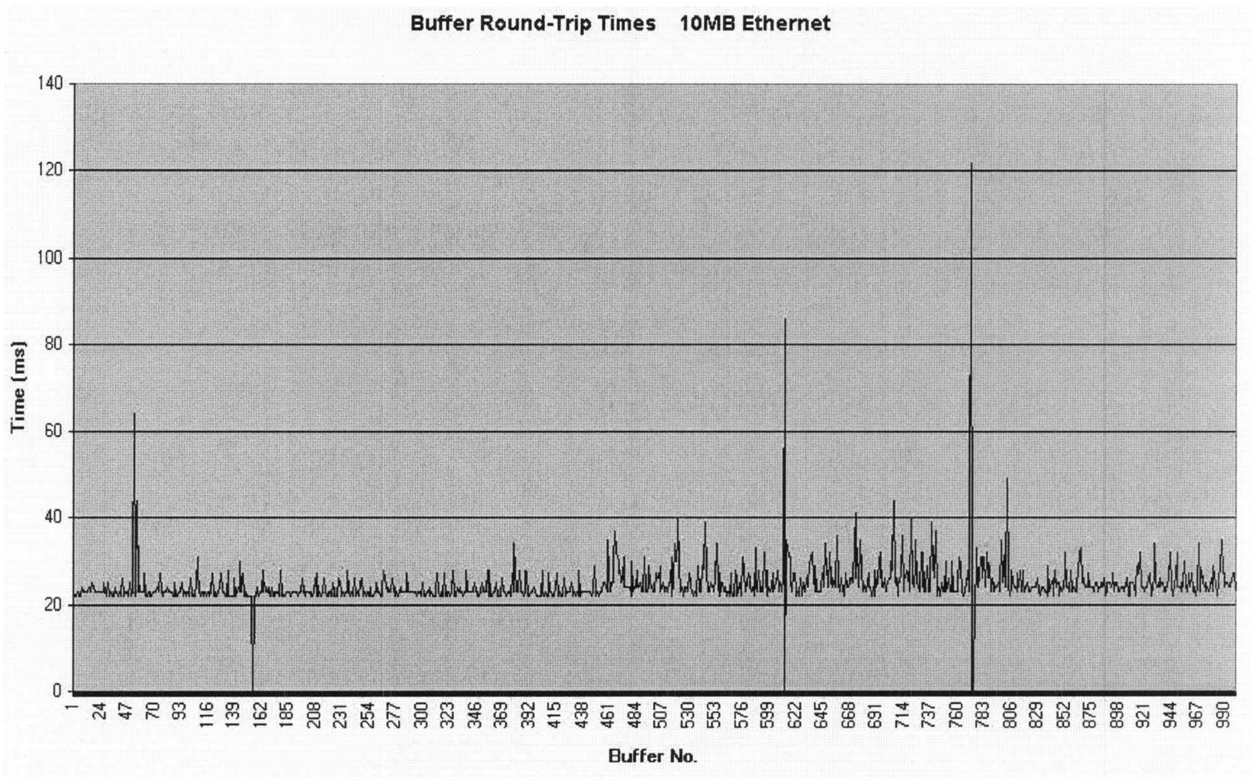


Figure 6.3 *Graph of Buffer Round-Trip Times for 10M Ethernet with no server processing*

Figure 6.3 shows the plot of 1000 consecutive buffer round-trip times, measured on the Ethernet network, and with the server audio processing disabled. The zero values in the graph represent lost datagrams (datagrams that were sent, but which never arrived). The cause of these lost datagrams is discussed in section 6.1.2.2, which describes the results of the 100 Mbit/s Ethernet experiments.

The average times for a number of these experiments are shown in the table below.

Experiment	1	2	3	4	5	6	7	8	9	10	**Average**
Average Round-trip Time (ms)	24.83	24.13	24.71	25.39	24.65	24.71	24.49	25.27	25.61	24.64	**24.84**

Figure 6.4 *Average Buffer round-trip times using 10Mbit/s Ethernet*

An Ethernet network is a shared network medium. A workstation will only proceed to transmit a data packet over the network if the cable is not occupied, otherwise it will wait until the cable is free. If more than one workstation transmits simultaneously, a collision occurs, and both workstations stop transmission, each waiting a random time before attempting to transmit again (Tanenbaum, 1988). Because all workstations have an equal opportunity to access the medium at any time, there is quite a lot of variation in the buffer round trip times, and this will increase with an increase in the bandwidth utilization. The data rate that is required for one monaural track of 16-bit audio sampled at 44100 Hz is approximately 5 Mbytes per minute. For 5 monaural tracks the data rate equates to 2.5 Mbytes per minute, or about 3.3 Mbit/s. This would need the 10Mbit/s Ethernet network to operate at around 30% utilization.

The irregularity in the delay of audio buffers is not problematic because the audio output multiple buffering will 'absorb' the occasional unusually long delay. However, if a datagram is lost, the audio information contained in that datagram is unrecoverable. The loss of one datagram would result in missing audio information for 1/20th of a second, which is noticeable in the audio stream.

The data shown in the graph for the experiment above was collected while spatializing sound with the 3D mouse. This means that GUI overheads and operating system delays play a part in delaying the precise measurement of buffer round-trip times. With all the experiments, the 3D mouse was placed out of scope of its receiver unit for certain periods of time during the collection of data of the 1000 buffers round-trip times. From buffer 0 to approximately buffer 400, the 3D mouse had no contact with the system. From about buffer 400 through to buffer 800, the 3D mouse was brought into range of its receiver, and the mouse moved continuously for that period of time. From buffer 800 to buffer 1000 , the mouse was again removed from range of its receiver, and the system operated with no 3D mouse input. It may be observed in the Figure 6.4 that between buffer 400 and buffer 800 the round-trip times are more erratic and generally higher than the other two sections outside of these buffers. These irregularities correspond to the movement of the 3D mouse and occur because the 3D mouse coordinate update messages are queued with other notification messages in the application, and are handled one at a time.

6.1.2.2 100 Mbit/s Ethernet Experiments

In order to reduce the delay caused by the network, the 10 Mbit/s link was replaced with 100 Mbit/s Ethernet. The experiments were otherwise identical. The diagram below shows a graph plot of 1000 consecutive buffer round-trip times measured with the 100 Mbit/s Ethernet in place and the audio processing on the server disabled.

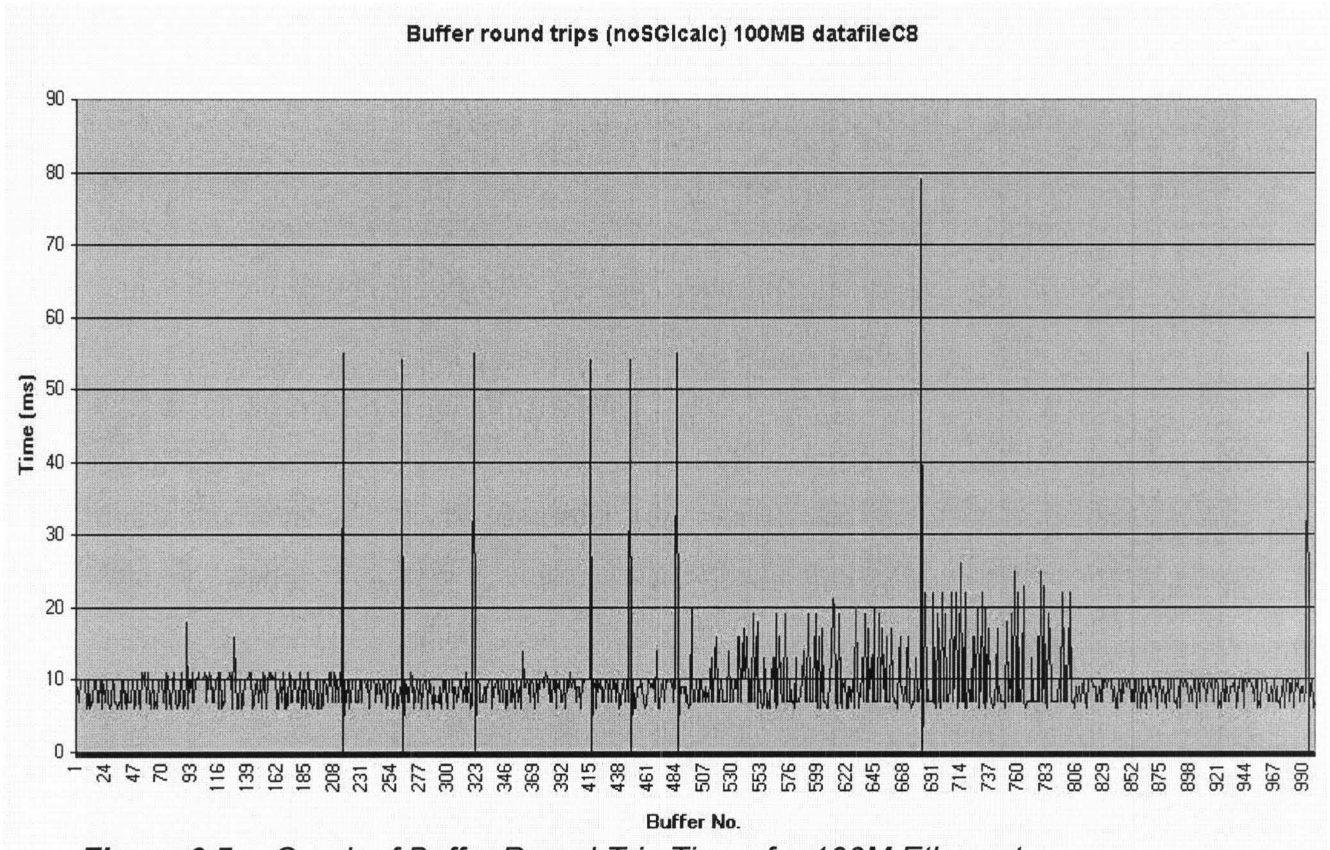


Figure 6.5 *Graph of Buffer Round-Trip Times for 100M Ethernet with no server processing*

The use of a 100 Mbit/s Ethernet link exhibits a large improvement in the buffer round-trip delays. This improvement cuts the latency due to the network by about 60% from about 25ms to under 10ms. In figure 6.5, one can observe that the variation in the round-trip times is small and usually remains between about 8 and 10ms. The experiment was repeated several times and the results are shown below.

Experiment	1	2	3	4	5	6	7	8	9	10	**Average**
Average Round-trip Time (ms)	9.90	10.09	9.92	9.39	9.40	9.35	9.18	9.39	9.59	9.28	**9.55**

Figure 6.6 *Average Buffer round-trip times using 10Mbit/s Ethernet*

The 3D mouse movement was restricted to the period between the 400th and 800th buffer round-trips as in the first experiments with 10Mbit/s Ethernet. The effect on the measured buffer round-trip times are clear in Figure 6.5.

Datagrams are lost in both 10 Mbit/s and 100 Mbit/s experiments. With a required data rate of 3 Mbit/s over a 100Mbit/s network, it is assumed that the network is not the chief cause for the discontinuities seen in these results. Assuming the network is not responsible for dropping datagrams, the problem must lie in the application software. At certain times, the client application fails to pick up a datagram from the network. This often happens after a peak in the buffer round trip delay, as can be seen in Figure 6.5. A possible cause of this would be that when a buffer is retrieved from the network by the client, and for some reason, the system is delayed for a short period, the next datagram arriving over the network is not detected.

The following section describes some modifications to the software used for the experiment that attempt to reduce some of the variation in the buffer round trip times measured in the above experiments.

6.1.3 Latency due to Multiple Buffering

The time delays caused by the multiple buffering of the input and output wave streams are represented by numbers 1 and 7 in figure 6.1. The need for multiple buffering in this system has been discussed in Chapter 4. With a standard audio buffer size of 4096 bytes (2048 samples) the input and output wave audio streams are implemented using 3 buffers for both wave input and wave output.

6.1.3.1 Wave Input Multiple Buffering

When capturing audio from the wave input device, a finite time is taken for the wave input device to fill an audio buffer with recorded audio. With a buffer size of 4096 bytes using 16-bit audio sampled at 44100 Hz, the time taken to capture one complete audio buffer is approximately 46 milliseconds. After this time, the buffer is returned to the application. Multiple buffering of the wave input is implemented by ensuring that the wave input device always has as least one fresh buffer available in which to put any captured audio data. This is done by supplying the device driver with several audio buffers before the start of wave input. As soon as a buffer is filled, it is returned to the application for processing and another buffer is sent to the wave input device driver.

6.1.3.2 Wave Output Multiple Buffering

The delays in the audio path that occur throughout the system before the output of audio to the speaker array, are small, but irregular enough to produce an audio output stream that contains many gaps and jitters. An audible gap will occur when the wave output device driver has finished outputting a buffer, and the next buffer to be played has not arrived. The output stream is smoothed out by introducing multiple buffering of the arriving audio buffers. Once the system has started receiving audio buffers arriving from the server machine, a number of buffers are accumulated before writing any buffers to the wave output device. When the specified number of buffers for multiple buffering have arrived, they are written to the wave out device, and the audio output starts. Any buffers that arrive after this point are simply written straight to the output device. A smooth output stream is ensured as there will always be a few buffers waiting to be output by the device driver, and this has the effect of eliminating the unpredictability of the buffer stream received from the server. The delay caused by multiple buffering the output will be the time taken for the first three audio buffers to arrive. This delay overshadows all other delays in the system, and as explained in the next section, is significant enough to determine the total latency in the system.

There is no easy experimental way to measure these buffering delays, but they can be calculated from the size and number of buffers used in multiple buffering. Because there are a number of concurrent processes in the system, the effective delay introduced by multiple buffering the wave input and output streams is more complex to calculate. Figure 6.7 illustrates broadly the delays in the overall system, and the total latency is calculated thereafter.

6.1.4 Total Latency of Audio in the Distributed System

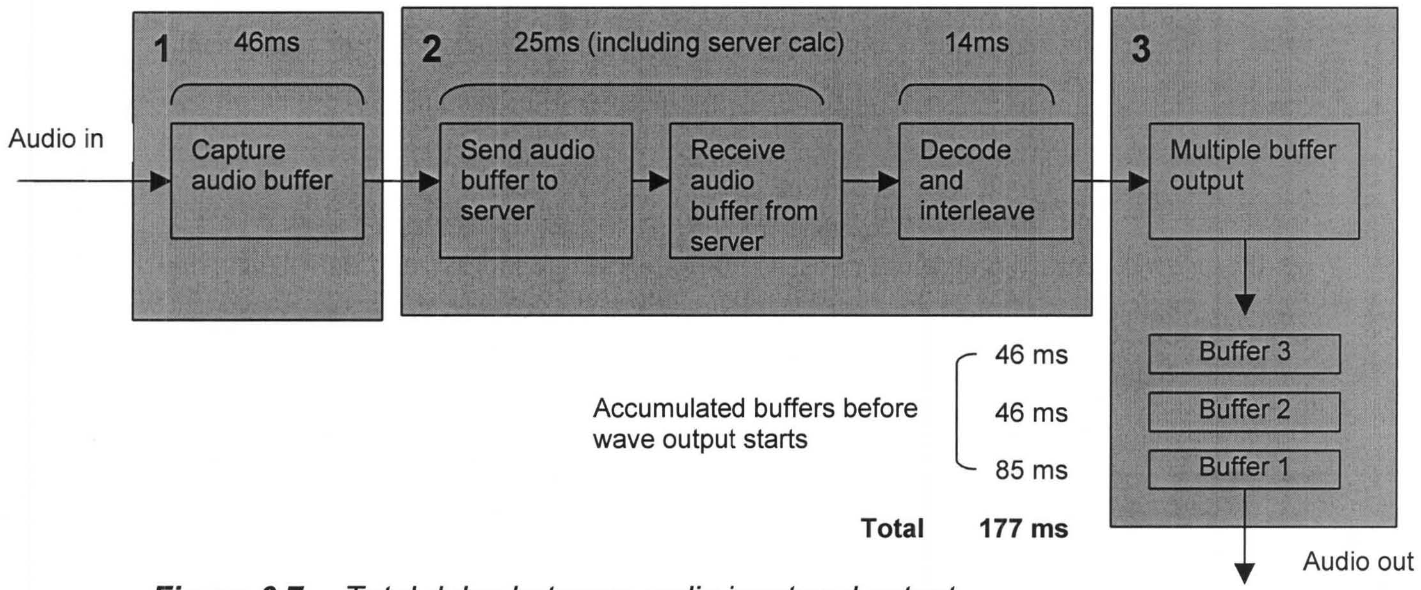


Figure 6.7 *Total delay between audio input and output*

The total delay between the input of an audio buffer and the output of a spatialized version of that audio buffer is the sum of the delays of the input buffering, the network delay and server processing, the speaker decoding and stereo interleaving and the output multiple buffering. Sections 1 and 2 in Figure 6.7 are effectively parallel processes in the system. After the first buffer is captured (46ms from start), it is sent through section 2. In the meantime, the second audio buffer is being captured. By the time the first buffer has reached the output multiple buffering stage (section 3) 39 ms later, the second audio buffer is still being captured. After another 7 ms this buffer will be filled completely and sent, and capture of the third buffer will start. When the second buffer reaches section 3 (total time for second buffer = 46ms), the third buffer will have been filled for 39ms. After another 7ms the third buffer will be filled and sent, and will take 39ms to reach section 3 (total time for third buffer = 46ms). Only at this time is wave output started. A Ward and Mellor model showing the concurrent processes in the system is given below (Ward & Mellor, 1985).

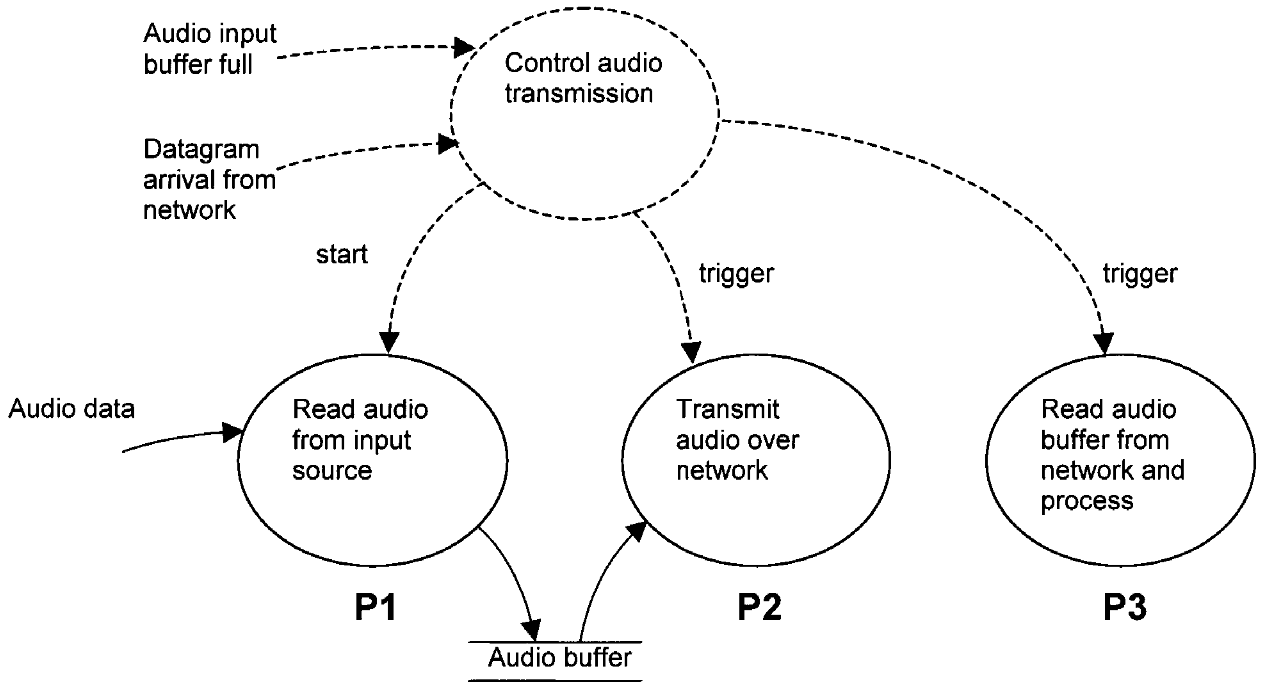

Audio input buffer full
Datagram arrival from network
Control audio transmission
start
trigger
trigger
Audio data
Read audio from input source
Transmit audio over network
Read audio buffer from network and process
P1
P2
P3
Audio buffer

Figure 6.8 *A Model of the Concurrent Processes in the System*

The concurrent processes in the system are shown in Figure 6.8 as P1, P2 and P3. P1 captures audio data, P2 transmits this audio data over the network to the server, and P3 reads the incoming spatialized buffers from the network and processes them for output.

A timeline outlining the total delay is shown below.

Time (ms)	**0**	**46**	**85**	**92**	**131**	**138**	**177**
	Wave input starts	1st buffer capture completed and buffer sent. 2nd buffer capture starts.	1st buffer arrives at output multiple buffering stage	2nd buffer capture completed and buffer sent. 3rd buffer capture starts.	2nd buffer arrives at output multiple buffer stage	3rd buffer capture completed and buffer is sent. 4th buffer capture starts etc...	3rd buffer arrives at output multiple buffer stage **Wave output started**

total delay of audio = **177 ms**

Figure 6.9 *Timeline for output multiple buffer accumulation*

An audio signal will be delayed by 177 ms before it is reproduced in a spatialized form over the speaker array. A basic experiment was set up to confirm this delay in practice using a microphone as the audio input source. A tap on the microphone produces a delayed spatialized tap over the speaker array, and both sounds were recorded by a second microphone connected to the soundcard of another computer. The difference between the two audio events was measured and this figure averaged out to approximately 183 ms.

The user would observe a delay of 177 ms when using an audio input source that allows interaction with the user. An example of this would be the use of a microphone as the audio source. The user can identify both an audio input event and its corresponding output spatialized version, and audibly detect the difference between the two. The delay perceived by the user will be the full 177 ms, as was confirmed in the audio experiment conducted above. The delay between two audio events should be at most about 30ms else the human auditory system would recognize them as separate events (Foss & Mosala, 1996). An example of this usage of the spatialization system would be a user spatializing his voice in real-time. There will be a delay of 177 ms between speaking into the microphone, and the spatialized version appearing over the speaker array. This long delay will cause a separation between the input and output audio events and the system will not be perceived as operating in real or near real-time. The effect produced in this example would be much like speaking into a public address system that has speakers spread over a large distance, and would produce an undesirable echo-like effect.

Because the system would normally be used with a non-interactive audio input source for example a CD line-input or wave file (and not a microphone) the perceived latency in the system is not as high as 177 milliseconds. The delay that is noticed by the user in this case will not be between audio input and output, but rather between the movement of the 3D mouse and the movement of the spatialized sound. The start of the measurement would be at the time of attaching the 3D coordinates to the buffer. This delay would be the same as the total delay of an audio buffer minus the time taken to fill the audio buffer at the input. Therefore the resultant latency observed by the user is 131ms. This latency in a real-time audio system would perhaps seem excessive. However in the context of this system, the user will be less sensitive to this latency, and the requirement for a very small delay is not as urgent as for example, a MIDI application. This is due to the fact that the two events between which the perceived delay occurs are the movement of the 3D mouse, and the corresponding movement of the spatialized sound through the speaker array. Both of these movements are continuous, and the lag of the spatialized sound behind the movement of the 3D mouse will only be noticed if the 3D mouse is moved rapidly. For example, the user might be moving the 3D mouse from the right-hand side of the coordinate space toward the left-hand side at a steady pace. The spatialized sound will move across the room following the 3D mouse gesture. If there is a delay that causes the spatialized sound movement to appear one or two metres behind where it should appear, the difference in position is hardly noticeable as long as the spatialized sound continues to 'catch up' with the mouse movement. A difference will be noticed if the movement of the 3D mouse from the right-hand side to the left-hand side is very quick. The delay of the appearance of the spatialized sound is consistent, and if the 3D mouse movement is rapid enough, the time taken for the 3D mouse transition would near the time delay of the appearance of the spatialized sound. The result would be that the 3D mouse coordinates would dictate the final left-hand side coordinates while the spatialized sound is still appearing at the right-hand side of the speaker array.

6.1.5 Optimization for Latency Reduction

The delays introduced by the operating system are often irregular and can inhibit the accurate measurement of buffer round-trip times. These delays will also affect the spatialization system latency adversely. In an effort to reduce the system latency, and provide more accurate network measurements, several modifications to the software on the client were implemented.

6.1.5.1 Optimizing the 3D Coordinate Capture and Audio I/O

The Pegasus FreeD 3D mouse has two ways of interacting with a Windows 95 application. The first is a mouse driver, which allows the data to be captured as one would capture ordinary mouse positional data (X and Y coordinates). An extra information variable provided with standard mouse messages carries the Z coordinate. In order to optimize an application running under Windows 95, an understanding of the basic way in which the operating system works is necessary. Windows 95 is a multi-tasking event-response type system. A message loop runs constantly during the life of an application. Messages are posted to the application signifying the occurrence of the events that the application may be interested in. For example, a button press may send a message to the application, which has a specific handling function for that event, which in turn contains the required actions for a behavioural response to that event. Mouse messages are another example of messages that may be posted to an application due to the occurrence of an event, for example the WM_MOUSEMOVE message, which is posted to any interested application if any mouse movement takes place. All windows messages are placed in a queue, and handled on a first-come first-served basis. The initial implementation of the experimental spatialization application used the Pegasus Win95 mouse driver to capture 3D coordinates. Whenever the Pegasus 3D mouse is moved, a standard WM_MOUSEMOVE message is generated and posted to the application, and a handler function is called to deal with that event. The resolution of the 3D mouse is very fine, and a large number of mouse messages are generated and placed in the Windows message queue. Other messages also need to be posted to the application to provide notification of external events. These include message from the GUI, the wave device drivers, and the network sockets. A large number of messages in the Windows message queue which are waiting to be processed causes interference with the sending and capture of packets to and from the network.

The coordinates of the 3D mouse can also be captured programmatically by using the Dynamic Link Library (DLL) supplied with the device. This interfaces directly with the application and does not make use of the mouse driver. Every time the 3D mouse is moved, the DLL sends a message to the application specifying the new coordinates of the device. These messages also appear in the Windows message queue. One of the main optimizations we wished to implement was to avoid the message posting behaviour of the 3D mouse completely, and thereby reduce the number of messages in the Windows message queue. This would allow more efficient handling of other messages such as datagram arrival messages and wave input buffer full messages. However, if the message queue is to be avoided, the 3D mouse should be able to make a direct call to a callback function, which would handle the event without needing to make use of the message queue. The DLL for the 3D mouse unfortunately does not cater for this type of communication with the application. When the 3D mouse is moved, the DLL sends a message to the Windows message queue, which then triggers a handler function. Even though message posting is still used as an update method for the 3D mouse, there are improvements in the performance of the software. This result suggests that even though both techniques use the message queue, the latter has less overhead than implementing the system with the Windows mouse driver.

The waveform audio input and output device driver can also communicate with the application in several ways. Messages are sent from the driver to the application for notifying the system about certain events. For example, when a wave buffer has been filled, a MM_WIM_DATA message is sent to the application, so a suitable response may be triggered. This can be done by sending the message to the application callback window, in which case the message gets placed into the Windows message queue and waits for any other messages to be processed before it is handled. A better option is to use a callback function to handle all wave input and output messages. This function is called directly by the wave device driver, and the Windows message queue is therefore bypassed. When using a callback function in the simple spatialization system used for the experiments detailed earlier, there is a significant improvement in the performance of the system. The use of callback functions for multimedia programming is suggested for a quicker response to multimedia event messages (Microsoft Multimedia Development Kit Programmer's Workbook, 1991). However, the performance gain is not as much as may be thought, and a recent study of real-time multimedia application performance under common operating systems, including Windows 95, suggests that callback functions actually cause more significant delays than expected (Brandt & Dannenberg, 1998). Figure 6.10 shows a graph of an experiment to measure the buffer round-trip times over 10 Mbit/s experiments. In this experiment however, the 3D mouse capture was implemented using the DLL instead of the Windows mouse driver, and the messages from the wave device drivers are received via a callback function instead of a callback window.

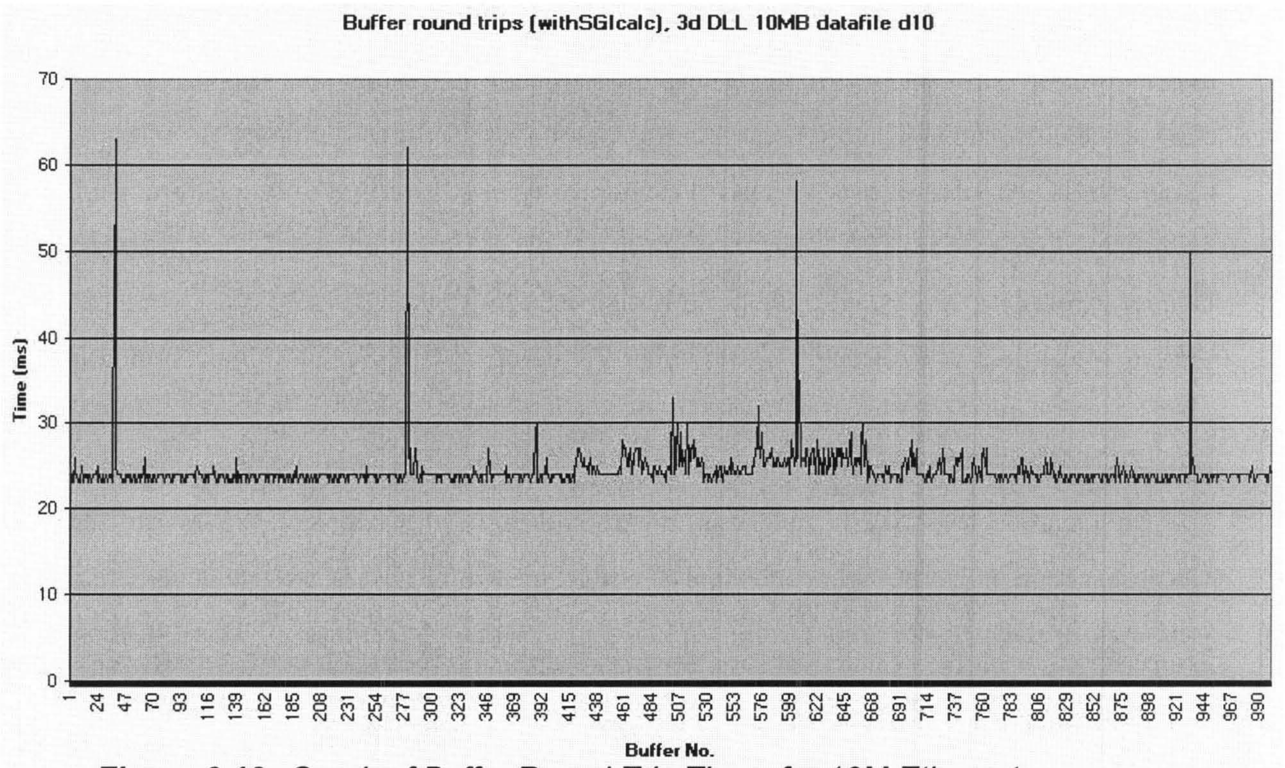


Figure 6.10 *Graph of Buffer Round-Trip Times for 10M Ethernet with 3D mouse DLL and wave I/O callbacks implemented*

The variation of round-trip times throughout the experiment remains fairly constant with less deviation than before. The mouse movement period between buffers 400 and 800 is still discernible, however the variation is not as erratic as in the previous experiments. In general there are also a lower number of lost datagrams.

6.1.5.2 Reduction of the Audio Buffer Size

It has been found that the major delays in the audio path of the spatialization system have been caused by the multiple buffering of wave input and output to ensure the uninterrupted audio streams. Fluckiger (Fluckiger, 1995) confirms that multiple buffering to compensate for any variation during the network transmission of sound is a significant delay factor in this type of system. If this delay can be reduced, then the overall performance of the system would be improved significantly. The only way to achieve this is to reduce the size of the buffers used in the audio stream. If a 4K buffer takes 46 ms to fill, a smaller 2K buffer will only take half as long to fill. The total delay calculated previously would be reduced from 173 ms to 100 ms including the input multiple buffering. The delay between the movement of the 3D mouse and the spatialized sound would be about 77 ms, and due to the non-precise character of 3D sound movement, this would allow performance of the system to be perceived as almost real-time. There is a problem in reducing the audio buffer size, as the output multiple buffering will provide less of a cushion for the variation existing in the audio path. For example, three 4K buffers might work adequately, but 3 1K buffers would perhaps be too small in total to soak up all variation. The number of buffers may therefore need to be increased with a decrease in buffer size, which will again increase the overall delay, thereby undoing the initial benefit of a reduced buffer size.

There are also other limitations when reducing the buffer size in the distributed spatialization application. Some of these were demonstrated in the following buffer size experiment. The experimental application was modified to use various sizes of audio buffers and varying numbers of these buffers. The network performance was noted, as well as the number of lost datagrams and the spatialized audio quality. The results are presented below.

Size of buffer	4k	3k	2k	2k	1k	1k
No. of buffers	3	3	3	6	3	9
Average round trip time	24.9	22.7	25.1	19.9	22.7	22.7
Datagrams lost (out of 1000)	1	1	7	11	51	68
Graph appearance	Normal	Normal	Very erratic	Fairly erratic	Very erratic during 3d mouse mov. GUI struggles	Very erratic during 3d mouse mov. GUI struggles
Audio quality	Good	Good	Degraded	Degraded	Bad	Bad

Figure 6.11 *Buffer Size Variation Results*

The results show that a reduction of the audio buffer size to 3K instead of 4K works with a corresponding improvement in the network latency and without the requirement of an increased number of these buffers. As well as cutting down the network transport time, the input buffering is reduced also, resulting in an overall latency of 138 ms. The reduction of the buffer size to 2K and 1K was generally unsuccessful, with many datagrams being dropped, and a large decrease in performance of the GUI environment.

It is also possible to only reduce the number of buffers used in the multiple buffering. For example, the use of two 4K buffers in multiple buffering resulted in good sound quality, and the latency was reduced to about 125 ms.

6.1.5.3 Optimizing Network Links and Processing

The most significant contributor to latency is the multiple buffering of the input and output audio streams. The real-time performance of both distributed and stand-alone spatialization applications will suffer from this added overhead. One can reduce this latency with a reduction in buffer size and the number of buffers in the multiple buffering stages, but only within limits before the optimization becomes counter-constructive. However, the overall performance can be improved by lessening the other delays that occur in the system.

The concept of distributing the system by placing some of the more intensive processing on a server was formed in an attempt to increase the efficiency of processing time, with some sacrifice to network latency. The network performance on 10 Mbit/s link is fairly constant and reliable, but it does make up a significant proportion of the total latency. The decision to transfer B-format, and not speaker feeds over the network has ensured that the network will always behave the same way despite the number of speakers in the reproduction array. The use of 100 Mbit/s Ethernet however, increases the network performance and can introduce a network latency (both to and from the server) of as little as 9 ms per 4K audio buffer. Coupled with the saving in processing time gained by spatialization processing, mixing and soundfield manipulation on the faster server machine, a near real-time spatialization application can be realized.

The network links used in the system are effectively dedicated paths for the audio transmission needed to distribute some of the processing in such an application. If these networks are shared LANs, the reliability of real-time audio data is seriously threatened. The 100 Mbit/s Ethernet will perform far better than the 10 Mbit/s Ethernet, but this is granted only if other traffic on the network segment is kept to a minimum. The Ethernet protocol works using a shared network medium. Only one workstation at a time can transmit an Ethernet packet over the network. If a workstation wants to transmit a packet, it can detect if the medium is occupied, and if this is the case, it will wait for a short period before attempting transmission again. This is a major disadvantage of using a distributed system as a dedicated network is needed between the server and client workstations. Fluckiger suggests the use of ATM (Asynchronous Transfer Mode) for bandwidth-intensive, real-time multimedia applications (Fluckiger, 1995). Bandwidth can be guaranteed with ATM unlike Ethernet. This type of approach would allow a shareable network to be used, but also would provide a reliable network link for multimedia traffic.

6.1.6 Conclusions Regarding Distributing Processing in the Spatialization System.

The capture of audio, the decoding of B-Format to speaker feeds, and the output of these speaker feeds are the tasks performed by the client in the distributed spatialization system. In addition, all control of the distributed system is located on the client workstation. The server has the responsibility of synthesizing B-Format spatialized audio from monaural audio, and the mixing and manipulation of that synthesized B-Format.

In distributing the system, a network transfer delay is introduced. The decisions to place both the synthesis and further processing of the B-Format signals sought to minimize the total network load, and thereby ensure the most efficient network transport of audio possible. The synthesis and processing could also be applied more efficiently on the server than on the PC. The network always has five monaural audio streams at any given time. This amount of data consumes approximately a third of the 10 Mbit/s network capacity. A 100 Mbit/s network provides a faster and less varied buffer transit time, and is the preferred network transport protocol for this application.

In order to keep the network load at a minimum, the B-Format decoding was placed on the PC. Another reason for this placement was to place the control of the output speaker array configuration on the client. The latency of the system would be increased by placing this processing on the client, but is accepted in order to maintain a constant, useable network load. For example, if eight speaker feeds were decoded on the server and transferred to the client, the network would be heavily utilized, possibly to the extent of degrading the audio quality to an unacceptable level. The overall perceived latency has been found to be useable, and this matter is discussed in section 6.2.

Much of the CPU intensive audio processing is placed on the server in order to reduce the latency in the system. However, this might be viewed as a poor solution, because the network carrying the audio between the client and server introduces its own delays into the system. Modern desktop processors would now be able to handle a fully functional spatialization application without the need for distribution of processing. However, even with much more powerful desktop machines becoming available, the extensions of the spatialization application to improve all audio processing aspects in the system would require more processing power. These extensions would include more complex soundfield synthesis, and an upgrade of the audio quality of the system to 24-bit audio sampled at 96 kHz. The application would then have to be ported to a bigger server-type machine, or coded onto dedicated DSP hardware. The distributed system presented in this thesis provides a framework that allows the extension of the spatialization system in this manner, while providing a user interface on a standard desktop PC.

There are disk space concerns when dealing with multitrack audio systems, and particularly multi-channel systems where several audio channels exist in each track, and a number of tracks make up a composition. Monaural 16-bit audio sampled at 44100Hz has a data rate of approximately 5 Mbytes per minute. In the case of a four-minute four-track B-Format composition, the disk space needed is sixteen times this figure, which results in a disk usage of 320 Mbytes for one average length piece. Server machines are often fitted with fairly large disks, and this is another advantage of the distribution of the system.

6.2 Qualitative Issues

6.2.1 Perceived Real-Time Response of the System

An important goal of the system described in Chapter 5 was to achieve near real-time movement of the virtual sound source in response to the user's movement of the 3D input device. The interactivity of the system is greatly enhanced if the delay between the movement of the user and the movement of the sound is reduced sufficiently. The total latency expected in the use of a simplified version of the spatialization application has been explored in the previous section. The fully developed system would show a greater overall latency because of the distributed server processing and disk access involved.

The actual delay between the movement of the 3D mouse and the movement of the virtual sound source is approximately 133ms. While this delay between two sounds would be distinct, the movement of sound with hand gestures is more forgiving. In practice the perception that the virtual sound source is moving in real-time with the 3D mouse remains up to a delay of about 150ms between the mouse movement and the sound movement. Only very fast gestures cause the lag in the spatialized sound movement to become noticeable.

6.2.2 Ambisonic Reproduction Performance

The optimal listening position in a regular speaker array is in the sweet-spot at the centre of the array (as was mentioned in Chapter 3). However in practice it is noted by many of the Ambisonics enthusiasts that this sweet spot is a fairly broad area around the centre of the array. The positioning of virtual sources in the soundfield can often be perceived even if one is standing outside of the speaker array (Elen, 1998). These observations have been confirmed during the auditioning of our reproduction array, provided that the listener is not too close to any individual speaker. The speakers used in the array should have a flat-frequency response and generally all the speakers should be identical. The individual speakers were placed in a geometrically accurate cuboid, but good results were obtained if the speakers were offset from those positions by a small distance. The main concern is to ensure that all speakers are directed toward the centre of the array. When setting up the array in an auditioning room, an anechoic environment is optimal. Sound reflections of speaker outputs off room walls create incorrect cues in the reproduced soundfield, and will alter the accuracy of localization. Our speaker array was set up in a carpeted room with concrete walls, and the soundfield reproduction was good, even though the room acoustic was not completely dry.

The shape of the speaker array is completely flexible if the B-Format is decoded appropriately for the required speaker layout. Humans perceive sound emanating from the left and right directions much more sensitively than from the front and back due to the position of our ears. If a B-Format soundfield is decoded to a square array, for example, an increased perception of sound localization in front of and behind the audience may be rendered by retaining the same square speaker layout decoding, but physically stretching the array in the forward-backward direction. Likewise, the auditory localization in the up-down direction is significantly weaker than both the forward-backward and left-right directions. If a cuboid array is used, the performance of the array when reproducing 'with-height' surround sound (the B-Format signal contains W, X, Y and Z) would be greatly improved in a fairly tall room, with the speakers placed in all eight corners, and the audience platform placed mounted halfway up the room.

The Ambisonic B-Format decoder implemented in software is a simple one. Nevertheless the reproduced soundfield is quite good. Sound source images are placed accurately and are very stable. A major feature of Ambisonic reproduction systems is their ability to provide the illusion of an almost completely seamless soundfield. If a virtual sound source is moving around the perimeter of the array the sound does not move from speaker to speaker, but the image will carry between the speakers. If one closes one's eyes, the speaker directions often become somewhat ambiguous. This is an enormous improvement over the 5.1 systems where 'holes' are found between speakers, for example the right front and right surround speakers. There is a tendency for high frequency sounds to 'pull toward' the speakers with Ambisonic decoders, and this effect is discernible in our reproduction array. The sounds that cause this effect are sounds with sharp attacks such as high-hats. This effect is not helpful for creating virtual sound images between speakers. It is preferable to use softer attacking sounds such as strings and pads if the sound imaging between the speakers is to be fully beneficial.

6.2.3 Surround Sound Music Production and Compositional Aspects

The availability of a 360-degree 'with-height' surround sound stage in which to place and pan audio has many implications for music composition. Several of these issues are identified and explained here, and are sourced mostly from experience with the spatialization system described in Chapter 5.

The spatialization system uses multiple audio tracks as input in order to build a spatialized composition. These tracks can originate from any audio source, though with computer music composition, sequencers and sound modules are often used. The way in which the system would typically be used would be to create the composition with a sequencer and some sound modules, and then record that composition into say, 4 audio tracks. It is important to monitor the mix of the 4 audio tracks in mono before they are spatialized in order to equalize the individual levels of the tracks. A disadvantage of monitoring in mono is that when the tracks are separated into spatialized tracks in a surround mix, the sounds won't interact with each other in the same way as with the mono mix. A better method of monitoring before spatialization may be to monitor the four tracks in a stereo soundfield placing the tracks as far apart as possible. This would help to project an idea of how the tracks will sound when they are spatialized.

Sounds can be panned throughout the surround soundfield in a composition. This effect is useful for rendering a soundfield that is 'alive' and that attracts the listener's attention. However, one should be careful not to use this new capability as a gimmick for rapidly moving many sources around at the same time. The human ear is not well enough trained to follow more than two or three moving sources. The sources should also not move too quickly or they will not be followed accurately by the ear.

As in stereo soundfields, the surround soundfield should exhibit a balanced nature. In most cases this works better than monopolizing one side of the soundfield. However, this may be a compositional strategy to surprise the listener with a sound source from a previously unused location in the soundfield. While composing a piece to be spatialized, much thought has to be given to the placement of certain sections of the composition in the soundfield, and in many cases the piece can be tailor-made for surround sound reproduction. There is a large difference between working with a stereo soundfield and a surround soundfield. In a stereo soundfield it is all too easy to 'clog up' a mix by using too many sound sources with different timbres, and a muddy effect can result if the instruments and effects are not chosen carefully. With a 360-degree soundfield at one's disposal, the soundstage is very large, and much more 'sound' can be used in a mix, while preserving some separation and clarity. The disadvantage of spatializing a composition after it has been composed is that a mix needs to be done in mono before spatialization, and the timbral effects in the mono mix do not usually present themselves in the same way in a surround soundfield. For example, in a stereo mix, one might place similar string sounds in the same place for a warmer sound. If these two sounds exist on separate tracks that are to be spatialized, the warmth will be destroyed if the tracks are not placed together in a surround sound mix. The advantage to this is that more sound separation can be achieved in a surround mix than a stereo (or mono) mix because of the widened soundstage.

Often it is useful to have stationary sources in the soundfield against which a moving source will be effective. This is evident in stereo mixes, where too much panning of sources bores the listener and has a tiring effect. One panning source against a background of stationary sound is a more effective option. The orientation of the audience has traditionally been toward the performers at the front of the listening area. With a surround sound technology capable of rendering consistent spatial quality from any direction, the composer can now consider the issue of listener orientation. The composer has far greater freedom in deciding how to present music to an audience than before. The spatial aspect of a composition can be viewed as another method in the composer's toolbox. There is still much to be learned in this area, and any further discussion on this matter is reserved for a more in-depth study of the compositional implications.

Summary

The performance of the network and processing aspects of the spatialization system proposed in Chapter 4 and described in Chapter 5 have been explored. Experiments to measure the network delays and the processing delays were carried out, and some optimizations to reduce the overall latency in the system were discussed. The decisions made in Chapter 4 over the distribution of components of the system have been reviewed, and we have found the distribution of the system to be appropriate for realizing a near real-time spatialization system. Chapter 7 follows to conclude the findings in this study.

Chapter 7

Conclusion

7.1 Thesis Overview

The goal of this thesis was to investigate the practicality of a distributed surround sound system for the creation and performance of surround sound music compositions. Essentially, there are two ways in which to deliver a spatial sound experience to a listener. The first is binaural 3D sound, which is reproduced over headphones. Although capable of rendering very realistic spatial sound, there are disadvantages associated with this approach to surround sound music production, such as fixation of the soundfield orientation to the listeners head orientation if head-tracking is not used. The other option for producing surround sound is to use many speakers located around the listener, and this approach was followed instead due to the impracticality of implementing binaural sound for a large audience.

Several surround sound technologies have been developed, and the dominant systems are motion picture theatre systems like Dolby Digital, which provide 5-channel surround sound and a subwoofer channel for low frequency effects. A surround sound music reproduction system should render an equal imaging quality from all directions, and these images should be stable. As theatre systems rely mostly on level-only localization, which is only one of several psychoacoustic criteria for our perception of directional sound, their spatial imaging is poor particularly at the back and sides of the 5.1 speaker layout. We therefore decided to use Ambisonic methods for creating and reproducing music since they offer the accurate and stable sound source imaging necessary for surround sound music compositions.

B-Format is the 4-channel professional studio format used for the representation of ambisonic surround sound. Ambisonic technology offers techniques for encoding monaural sound into B-Format, and for decoding B-Format to a variety of speaker layouts. B-Format signals can also be mixed to form composite soundfields, and manipulations such as soundfield rotations can be performed on the soundfield. The delivery of B-Format as a consumer surround sound format has been impractical until recently because of a lack of 4-channel delivery mechanisms. Another ambisonic format, UHJ, provides a 2-channel audio stream that can be decoded to produce surround sound, but is also compatible with mono and stereo systems. For surround sound reproduction, both B-Format and UHJ require decoding, and most home systems are 5.1 surround systems with no ambisonic capabilities. The introduction of the High Quality Audio Disc format within the DVD specification has provided a platform for 5.1 systems and ambisonic multi-channel systems to coexist on the same disc. Furthermore, formats have been developed for rendering ambisonic surround sound over conventional 5.1 systems. These emerging ambisonic formats aim to eliminate the need for an ambisonic decoder in order to experience ambisonic sound.

The ambisonic spatialisation of a monaural audio source necessitates the synthesizing of a B-Format signal from the source, performing additional processing such as mixing this signal with other B-Format soundfields, and decoding the resultant B-Format to a speaker array. In the realization of a real-time spatialisation system, the implications of processing ambisonic audio in software for basic spatialisation on a Pentium PC were explored. We proposed distributing the system using a client-server approach with the client (a Pentium 200 MH PC) and server (a Silicon Graphics Octane) machines linked with a fast network. This strategy would minimize the processing delays typically found in real-time audio applications by performing much of the audio processing on the fast server.

Experiments were conducted to investigate the computational costs of the essential audio spatialisation processes. The B-Format synthesis stage and the additional processing on that B-Format signal are potentially expensive operations, and were placed on the server. The B-Format decoding to a set of speaker feeds takes place on the client. This is also a relatively costly operation, but there are certain benefits for placing this processing on the PC that are outlined below.

The processing distribution choices made are advantageous in two ways. The first is that the client PC has control over the configuration of the speaker array. The B-Format signal created on the server is sent to the client and can be decoded to speaker layouts of different shapes and numbers of speakers by modifying the decoding equations. If the speaker decoding is done on the server, then the server must know about the speaker layout, decode the corresponding speaker feeds, and send them to the client for reproduction. The other advantage is that the network will always carry a 4-channel stream (B-Format) back to the client regardless of how many speakers are used in the reproduction array. This will result in a constant network load. The latency introduced by the network would also remain the same if improvements were made to the spatialisation processing operations performed on the server.

A surround sound spatialisation application requires an intuitive user interface for the movement of spatialized sound, particularly with surround sound that includes a height component. A simple 3D graphical user interface was developed for the client workstation that displays a visual representation of the position of the 3D mouse in a virtual room. This corresponds to the spatialized sound source position in the soundfield generated via the speaker array. The essential system allows the user to move the 3D mouse and hear a corresponding movement in the spatialized audio source within the speaker array in near real-time. The implemented system provides a basic spatialisation sequencer that allows the recording and playback of spatialized sound. A multi-track spatialized composition can be built by overlaying up to four spatialized tracks.

The performance of the multi-track spatialisation system implementation was closely examined in order to locate the main causes of latency in the system. The processing delays were measured on the client and server platforms to determine the delay caused by these operations. The network performance was analyzed by measuring the round trip delays of audio buffers in the distributed system using both 10 Mbit/s and 100 Mbit/s Ethernet. Both the audio processing performance and the network performance have a significant bearing on the total latency in the system. However, there are much larger delays in this type of real-time audio application. These delays are caused by multiple buffering of the audio input and output streams on the client PC. When audio is captured the system is delayed by the time taken to fill an audio input buffer with audio data. In addition to this delay, several buffers must be accumulated at the audio output. This has the effect of smoothing out the various delays occurring in the system and producing an uninterrupted audio output stream. The only way to reduce these delays is to reduce the audio buffer size. However, this reduction is only effective up to the point where the operating system's overhead of handling many small buffers starts to have a detrimental effect on the system.

This analysis of the spatialisation system in Chapter 6 shows a number of significant factors contributing to the latency in the system. However the usability of the spatialisation environment requires only that the perception of the spatialisation be in real-time. As discussed in Chapter 6, the actual latency of audio within the system can be significantly higher than the normal real-time audio latency allowance, and in practice, this has been confirmed. With a total audio latency of approximately 180 ms, the system is perceived to spatialize monaural audio in real-time.

7.2 Future Directions

The distributed approach to implementing the system sought to reduce the processing delays with the compromise of introducing network audio transfer. With a high-speed network, the transfer times can be diminished sufficiently to make the compromise insignificant. As the number of features enhancing the spatialisation system increases (with an increase in the associated audio processing), this approach will become more beneficial. A distributed system can be significantly enhanced, as any extra processing of B-Format would reside on the server machine. The extensions to the B-Format decoder would have to be implemented on the PC side, but with the PC dedicated to only the user interface, and the capture and output of audio, the extra processing overhead in improving the B-Format decoding process would be handled adequately by the PC.

The distributed system as it stands can be extensively enhanced. These improvements include an increase in the number of Spatialized tracks that may be laid, from four tracks to eight or perhaps even sixteen tracks. The audio quality may also be upgraded to the

24-bit 96kHz quality now preferred in professional studio work. The implications of these extensions are not trivial concerning the increase in audio processing and disk space. The disk space required for a surround sound composition which is already large, would be doubled with an increase to an eight track system, and tripled if the audio quality is upgraded. This would result in six times more disk space being required than with the present system. Likewise, the audio processing will increase accordingly. There is obviously a limit as to how far one should go to enhance the system, especially as computing resources are finite, and will continue to be for now. The improvements mentioned above are possible with a fast server machine and a large hard disk.

The audio processing tasks responsible for the software synthesis and reproduction of B-Format can also be refined. There is ample processing power on the server machine for more digital audio processing that would introduce effects like reverberation and early reflection synthesis to improve the realism of the soundfields that are generated. The decoding of B-Format can be enhanced to cater for different speaker layouts, and to appropriately alter the decoding formulas for low and high frequency sounds. Although these require software DSP, these features could possibly be incorporated without too heavy an effect on the PC performance.

It has been reasoned that the distribution of processing in this type of application is a good idea. However, there are still many advantages to a standalone PC spatialisation system. Firstly, no network or server hardware is required. This makes the application completely portable, and would be able to run on a standard PC. If the features of the system need to be limited to be able to handle only three or four Spatialized tracks, with fairly simple ambisonic surround encoding and decoding, the system would still be a powerful tool. Accessibility of the system to other users, who do not have the network or server machine facilities, is another advantage. Rewriteable CD-ROM technology is being released into the consumer market and with rewriteable, removable media of this type, the disk space issues become less troublesome.

There is an increase in the number of manufacturers that are producing affordable multi-channel audio cards that also have hardware DSP capabilities. Microsoft's DirectX programming interface allows a direct path from the programmer's application to the hardware residing on the audio card of the computer. These audio cards are now becoming DirectX compliant, and in cases like the mixing of digital audio signals, the card's DSP capabilities can be used for efficient processing. In this light, with the release of faster PC processors, the implementation of an enhanced real-time PC-based spatialisation system is not far away.

The creation of surround sound music is an undeveloped skill. The availability of a 3D soundfield in which to place and pan sound provides a host of possibilities for surround sound compositions. The implications of surround sound compositional systems have been touched upon due to the limited experience gained in this area through working with the distributed spatialisation system.

References

Bamford, J.	An Analysis of Ambisonic Sound Systems of First and Second Order, M.Sc. Thesis, Department of Physics, University of Waterloo, Ontario, Canada, 1995.
Begault, D.R.	*3-D Sound for Virtual Reality and Multimedia*, Academic Press, Inc., Cambridge, Massachusetts, 1994.
Borwick, J (Ed).	*Sound Recording Practice* (Association of Professional Recording Studios), Oxford University Press, 1987.
Brandt, E. & Dannenberg, R.	Low-Latency Music Software Using Off-the-Shelf Operating Systems, *Proceedings of the 1998 International Computer Music Conference, Michigan,* 1998.
Burgess, D.A.	Real-Time Audio Spatialisation with Inexpensive Hardware, Graphics and Visualisation and Usability Center, Georgia Institute of Technology, 1992[a], *(unpublished paper).*
Burgess, D.A.	Techniques for Low Cost Spatial Audio, Graphics and Visualisation and Usability Center, Georgia Institute of Technology, 1992[b], *(unpublished paper).*
Comer, D. & Stevens, D.	Internetworking with TCP/IP Volume II, Prentice-hall, Inc., New Jersey, 1991.
Davis, M.	The AC-3 Multichannel Coder, *Proceedings of the 95 Audio Engineering Society Convention, 1993.*
Elen, R.	Ambisonic Mixing – An Introduction, Studio Sound Magazine, September 1983.
Elen, R.	Whatever Happened to Ambisonics, AudioMedia Magazine, November 1991.
Elen, R.	Ambisonic Surround Sound in the Age of DVD, Audio Media Magazine, April 1998.
Farina, A.	Software Implementation of B-Format Encoding and Decoding, *Proceedings of the 104th Audio Engineering Society Convention, 1998.*
Foss, R. & Mosala, T.	Routing MIDI Messages Over Ethernet, *Journal of the Audio Engineering Society, Vol. 44 No. 5, May, 1996*

Fluckiger, F.	*Understanding Networked Multimedia – Applications and Technology*, Prentice Hall, Hertfordshire, England, 1995.
Furlong, D.	Comparative Study of Effective Soundfield Reconstruction, *Proceedings of the 87th Audio Engineering Society Convention, 1989.*
Fox, B.	Digital Dueling, *Studio Sound Magazine*, September, 1998.
Gerzon, M.	The Design of Precisely Coincident Microphone Arrays for Stereo and Surround Sound, *Proceedings of the 50th Audio Engineering Society Convention, 1975.*
Gerzon. M.	Practical Periphony: The Reproduction of Full-Sphere Sound, *Proceedings of the 65th Audio Engineering Society Convention, 1980.*
Gerzon, M.	Ambisonics in Multichannel Broadcasting and Video, *Proceedings of the 74th Audio Engineering Society Convention, 1983.*
Gerzon, M.	Ambisonic Surround-Sound Mixing for Multitrack Studios, *Proceedings of the 2nd Audio Engineering Society Conference, 1984.*
Gerzon, M.	Optimal Reproduction Matrices for Multispeaker Stereo, *Proceedings of the 91st Audio Engineering Society Convention, 1991.*
Gerzon, M.	General Metatheory of Auditory Localisation, *Proceedings of the 92nd Audio Engineering Society Convention, 1992[a].*
Gerzon, M.	Signal Processing for Simulating Realistic Stereo Images, *Proceedings of the 93rd Audio Engineering Society Convention, 1992[b].*
Gerzon, M.	Psychoacoustic Decoders for Multispeaker Stereo and Surround Sound, *Proceedings of the 93rd Audio Engineering Society Convention, 1992[c].*
Griesinger, D.	Multichannel Matrix Surround Decoders for Two-Eared Listeners, *Proceedings of the 101st Audio Engineering Society Convention, 1996.*
Kruglinski, D.	*Inside Visual C++, Version 4*, Microsoft Press, Redmond, Washington, 1996.

MacCabe, C. Pestel, C. & Furlong, D.	Virtual Imaging Capabilities of Surround Sound Systems, *Proceedings of the 93rd Audio Engineering Society Convention, 1992.*
Malham, D.	Computer Control of Ambisonic Soundfields, *Proceedings of the 82nd Audio Engineering Society Convention, 1987.*
Malham, D.	3-D Sound for Virtual Reality Systems using Ambisonic Techniques, *VR93 Conference, London, 1993.*
Malham, D.	Ambisonics – A Technique for Low Cost, High Precision, Three Dimensional Sound Diffusion, *Proceedings of the International Computer Music Conference, Glasgow, 1990.*
Malham, D.	Progress in the Application of 3-Dimensional Ambisonic Sound Systems to Computer Music, *Proceedings of the International Computer Music Conference, Montreal, 1991.*
Priest, R.	A Windows Surround Sound System, Honours Thesis, Rhodes University Computer Science Department, 1995.
Reilly, A. & McGrath, D.	Using Auralisation for Creating Animated 3D Sound Fields Across Multiple Speakers, *Proceedings of the 99th Audio Engineering Society Conference, 1995(a).*
Reilly, A. & McGrath, D.	Convolution Processing for Realistic Reverberation, *Proceedings of the 98th Engineering Society Conference, 1995(b).*
Roads, C.	*The Computer Music Tutorial*, The Massachusetts Institute of Technology Press, Cambridge, Massachusetts, 1996.
O'Modhrain, M.	SurroundSound – A B-Format Soundfield Processing Program for the Composers' Desktop Project Soundfile System, *Proceedings of the International Computer Music Conference, Glasgow, 1990.*
Smith, A. & Foss, R.	Network Delivery of Multi-Channel Audio for Surround Sound, *Proceedings of the Teletraffic '97 Conference, Grahamstown, South Africa, 1997.*
Smith, A. & Foss, R.	A Distributed System for the Composition and Delivery of Ambisonic Surround Sound Audio, *Proceeding of the South African Telecommunications, Networks and Applications Conference, Cape Town, South Africa, 1998.*
Tanenbaum, A.S	*Computer Networks*, Prentice-Hall, Inc. 1988.

Vennonen, K. A Practical System for Three-Dimensional Sound Projection, *Proceeedings of the Synaesthetica '94 Conference,* Australian Centre for Arts and Technology (ACAT), , Canberra, 1994.

Ward P. & Mellor, S. *Structured Development for Real-Time Systems*, Prentice Hall, Inc., New Jersey, 1985.

Technical References

Microsoft Multimedia Development Kit Programmer's Workbook	Microsoft Corporation, 1991.
Pegasus FreeD User Manual	Pegasus Technologies Ltd. Israel, 1996.
Event Darla User Manual	Echo Corporation, Carpinteria, CA 93013, 1997.

World Wide Web References

[1] LakeDSP Home Page	http://www.lakedsp.com/
[2] Meridian Audio Home Page	http://www.meridian-audio.com/
[3] Dylan Menzies-Gow's LAmb site	http://www.york.ac.uk/inst/mustech/3d_audio/lamb.htm
[4] Richard Furse's Ambisonic Csound Page	http://www.muse.demon.co.uk/csound.html
[5] The Leeds Csound Front Page	http://www.leeds.ac.uk/music/Man/c_front.html
[6] Angelo Farina's Aurora Page	http://pcfarina.eng.unipr.it/aurora/
[7] Gerzon, M.A. U.S. Patent No. 3997725 located by the IBM Patent Server	http://www.patents.ibm.com/
[8] The Surround Sound Discography	http://personal.riverusers.com/~manderso/
[9] Doug Cook's SGI Audio Mixing Page	http://reality.sgi.com/employees/cook/audio.apps/dev/mixing
[10] Event ElectronicsHome Page	http://www.event1.com/
[11] Steinberg (Cubase)	http://www.steinberg.net/
[12] Syntrillium (Cool Edit Pro)	http://www.syntrillium.com/
[13] The AVI File Format Overview	http://www.rahul.net/jfm/avi.html

Other Relevant World Wide Web Pages

The Ambisonic Home Page	http://www.ambisonic.net
The Ambisonic Index	http://www.apogeedigital.com/ambisonics/
Kimmo's Ambisonic Links	http://www.anu.edu.au:80/ITA/ACAT/Ambisonic/Ambisonicslinks.html
York University Ambisonic Page	http://www.york.ac.uk/inst/mustech/3d_audio/ambison.htm
The Ambisonic FAQ	http://www.york.ac.uk:80/inst/mustech/3d_audio/ambsfaq.htm
Dolby Laboratories – Dolgy Digital	http://www.dolby.com/digital/
Digital Theatre Systems	http://www.dtstech.com/

Appendix A

Operating Manual

A1 Compilation of the Source Code

Spatserver - The Spatialization Server Application

The server application source file **spatserver.cpp** is written in C++ and is compiled with the MIPS C++ compiler running under SGI IRIX 6.4. The executable object file **spatserver** is created with the command:

```
cc spatserver.cpp -lm -o spatserver
```

The mathlib option **–lm** is needed for the cos and sin functions in the spatserver.cpp file. For each

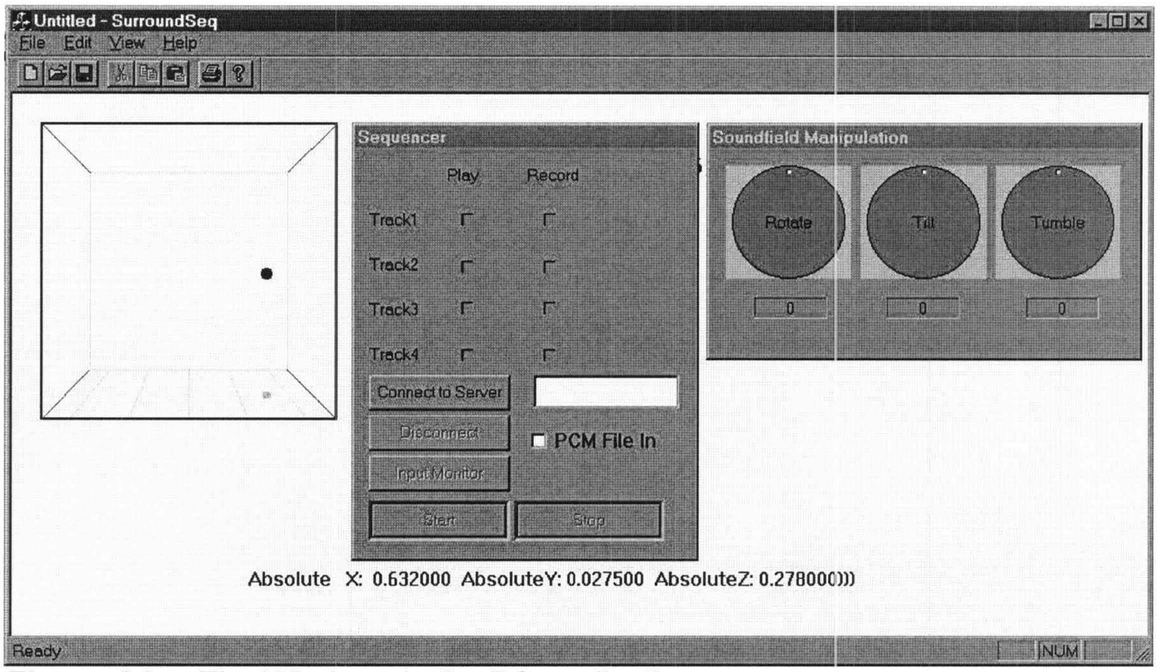
SurroundSeq - The Spatialization Client Application

The client application should be compiled with Visual C++ version 5.0, under Windows 95. The executable file is **SurroundSeq.exe**.

A3 Getting Started

The **spatserver** application can be run from a Telnet client on the PC. It is necessary to start this program to initiate the TCP server role before starting the client SurroundSeq application. Each spatialized track is a four-channel raw PCM track and the tracks are named track1.pcm, track2.pcm, etc… For each different multitrack spatialization, it is necessary to copy the executable to a new directory in order to prevent recorded audio files from being overwritten.

The client application, **SurroundSeq**, can then be executed on the PC. Several message boxes will appear confirming the number wave input and output devices and the successful preparation of the 3D mouse and the network socket. The main application window will then appear, with two floating windows, a Sequencer window and a Soundfield Manipulation Window. The screen display is shown below.



Absolute X: 0.632000 AbsoluteY: 0.027500 AbsoluteZ: 0.278000)))

Figure A1 *The Windows in the PC application*

The box on the left side of the application window is the virtual room in which sound will be placed. When the 3D mouse is moved in range of the Pegasus FreeD receiver, a small blue circle will appear in the virtual room. This object portrays the position of the virtual sound source in the room, and it will follow the 3D mouse movements. The virtual room is a cube centred about the origin (0,0,0) and extends one arbitrary unit in each of the X, Y and Z directions (X = left-right, Y = forward-backward, Z = up-down). It is necessary to configure the speakers to correspond to this convention, and to ensure that the orientation of the speaker array matches that of the application's virtual room. For example, when facing the computer, the X axis runs left to right, so the speakers that are decoded with X in phase should be setup on the right-hand side of the room. The absolute coordinates of the 3D mouse position are shown at the bottom of the application window.

Connecting to the Server

Clicking on the **Connect** button will initiate TCP communication with the server and enable the distributed spatialization system to operate in any of the three modes. These are Monitoring, Recording, and Playback of spatialized sound. The controls in the Sequencer window will also be enabled.

The Audio Input Source

The audio input source is taken either from the Darla audio card analog input, or from a PCM file residing in the directory of the **SurroundSeq** executable. The default source is the Darla input, but if a PCM file is to be used, the **PCM File checkbox** must be checked, and the full name of the PCM file entered into the edit box in the Sequencer window.

Spatializing the Source in Monitoring Mode

Once the source has been selected (if necessary), the monitor mode can be started by click the **Monitor** button. Any audio coming from the audio source will be spatialized by the **spatserver** program on the server, and the result reproduced through the eight Darla outputs to the speaker array. The spatialized audio source will be reproduced in the listening array at positions corresponding to the coordinates of the 3D mouse in the virtual room. To terminate monitoring, click on the **Stop** button.

Recording and Playing Back a Spatialized Track

A spatialized track may be recorded by checking the appropriate **Record** checkbox for that track. When the **Start** button is clicked the audio source will be spatialized, and the spatialized version will be recorded. The recording can be terminated by clicking the **Stop** button. The track that has been recorded can be played back by checking the **Play** checkbox for that track and then clicking the **Start** button. The spatialized track that has been recorded will now be played back over the speaker array.

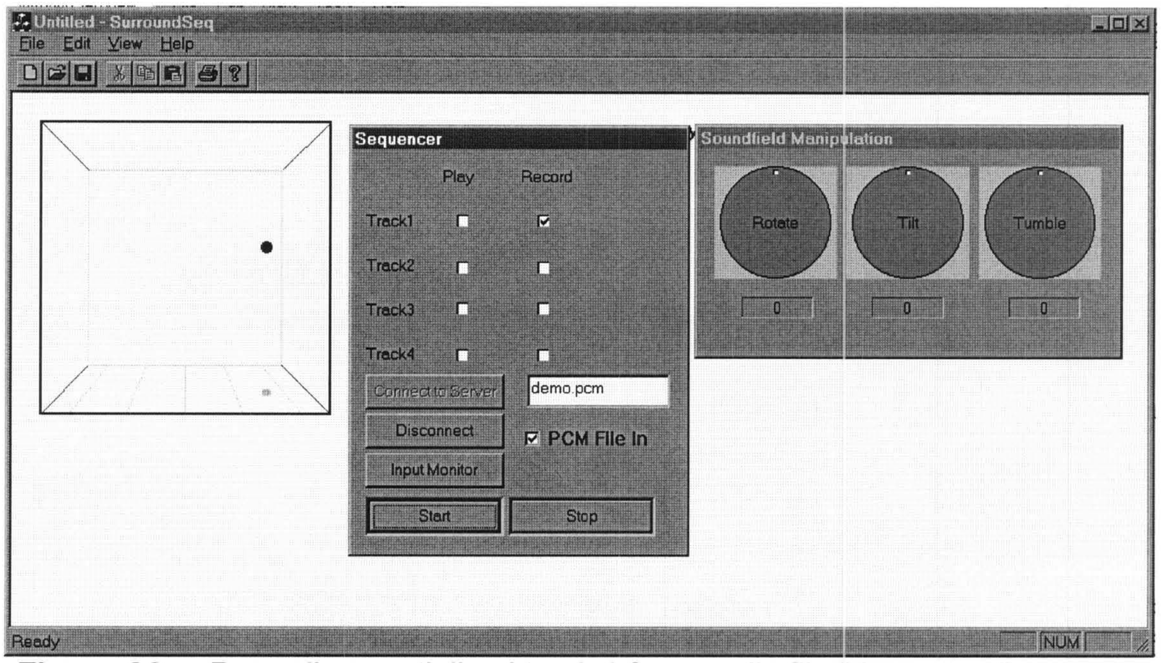


Figure A2 *Recording spatialized track 1 from audio file "demo.pcm"*

Figure A2 shows an example of recording a track with the record box for track 1 checked, and the PCM audio file source selected. At playback, the **Play** box is checked and the **Record** box is automatically unchecked.

Deleting a Track

A track is automatically deleted if that track number is recorded again. There is no other method of deleting a track except to record over it.

Recording and Layering Many Spatialized Tracks

After a track has been recorded, it can be played back by enabling the **Play** checkbox of that track while another track is being recorded. The tracks that have been previously recorded can then be heard while spatializing the currently recording track. After all required tracks have been recorded, the complete piece can be rendered by enabling the **Play** checkboxes of all the required tracks.

Manipulation of the Overall Soundfield

The three knobs in the **Soundfield Manipulation** window enable the composite soundfield to be manipulated by rotation, tilting and tumbling. Currently only the rotation functionality is available, and this manipulation information is not recorded.

A4 Terminating the Client and Server Applications

The Client Application is terminated with the standard Windows 95 'Close Window' button located at the top right of the main application window. The server application must be terminated with a hard break at present. In the telnet client a **Ctrl-Z** will suspend the application and show the IRIX command prompt. A process list must be called up with **ps** and the **spatserver** process killed with the **kill** function. An example is shown below.

```
Ctrl-Z
romeo%  ps
PID   TTY    TIME  CMD
543   ttyq2  0:00  tcsh
561   ttyq2  0:00  ps
594   ttyq2  0:00  spatserver

romeo% kill 594
[1]    Terminated   spatserver
```

The process number 594 is killed and the spatserver application is terminated.

A5 Bugs and Other Information

Several bugs exist in the spatialization system.

Firstly, if the audio card input is selected as the audio source, and only playback is happening, this source will appear over the speaker array following the 3D mouse.

Secondly, sometimes it is necessary to restart both the client and server applications if the system fails to work, and occasionally rebooting of the PC workstation is required.

The Internet Protocol numbers are hard-wired into the system. These need to be changed in both server and client applications according to the machines used to run the system. The wave device numbers in the client application are also hard coded. The configuration works for a system on which the only audio device installed is the Event Darla card. Raw PCM format files (these have no header) are also used as audio sources in the client application, and four-track raw PCM files are read and written in the server application. The audio quality is hard-coded for 16-bit sound sampled at 44.1 kHz.

The FreeD mouse driver should NOT be installed when using the system otherwise no 3D input is possible.