MSc Distributed Computing Systems Engineering

Department of Electronic & Computer

Engineering

Brunel University

Audio Server
for
Virtual Reality Applications

Marc Schreier

May 2002

A dissertation submitted in partial fulfilment of the requirements
for the degree of Master of Science

MSc Distributed Computing Systems Engineering

Department of Electronic & Computer

Engineering

Brunel University

Audio Server

for

Virtual Reality Applications

Marc Schreier

Ivor Brown

May 2002

A dissertation submitted in partial fulfilment of the requirements
for the degree of Master of Science

Acknowledgements

First of all, I want to thank my supervisor Mr. Ivor Brown for his advise and support. I also want to thank Mr. Roger Prowse for his support during the courses. Then I would like to thank Mr. Jürgen Schulze-Döbold for his useful recommendations during the practical work and furthermore Dr. Ulrich Lang and Mr. Uwe Wössner who allowed me to test the audio server at the Visualisation Department of the Stuttgart Supercomputing Centre.

Abstract

This document is about a system that provides surround sound to computer graphic visualisation systems. The Audio Server project proposes a hardware and software system based on products available on the consumer market. The hardware consists of a computer equipped with a network interface and a multi-channel sound card, which is connected to a multi-channel audio amplifier. An application software handles remote requests and manages the sound generation.

Conceived as a separate and mostly independent system, the Audio Server can easily be integrated with different VR systems. This provides a large field of applications.

A Prototype implementations of the Audio Server was built using OpenAL, a recent open audio library. An evaluation of the Audio Server has shown how the perceived sound direction depends on the position of the listener.

Table of Contents

1 Introduction

Various audio systems for virtual reality (VR) applications already exist, but in most cases they are experimental and use specially designed hardware and software components, or state-of-the-art equipment that is very expensive. Visualisation systems are mostly designed for high visualisation performance and therefore lack convincing audio properties. In the past years multi-channel audio for home entertainment has become very popular and affordable. Recent PC soundcards produce multi-channel and surround sound of a quality worth being investigated for use with VR.

First this document takes a look at sound generation and psycho-acoustic effects which affect human hearing. Then it describes some common surround sound formats and the use of surround sound for VR applications. The Audio Server concept will be presented and finally a prototype will be tested.

The following sections give an overview of acoustics and psycho-acoustics, surround sound and 3D sound, virtual reality and finally computer networks.

1.1 Acoustics

Acoustics is the physical term for sound related effects.

1.1.1 Sound waves

Sound is a physical phenomenon caused by temporal changes of media properties. If the contraction and release of a medium happens repeatedly, it will vibrate with a certain frequency and emit sound waves. The propagation speed of these sound waves mostly

depends on the density and temperature of the medium. The propagation speed of sound is 340 m/s in air at a temperature of 21°C. In water, the propagation speed is 1450 m/s because of the higher density.

Sound waves behave similarly to light waves. Generally a sound source emits sound waves equally into all directions. Special sound sources like loudspeakers can have a primary direction while no sound will be emitted to the other side.

Sound waves are affected by interference, reflection, diffraction, refraction, convection, etc. Sound waves can interact with other media, e.g., they can be transmitted from solids into fluids.

Each medium has a specific frequency at which it amplifies very well, which is the resonance frequency. Resonance is volitional with oscillators and music instruments but can have destructive effects with machines.

1.1.2 Room acoustics

Each room has its own acoustical characteristics that determine how sound is affected by the room itself. They depend on many parameters like room size, the materials used for the floor, walls and ceiling, location and type of windows, curtains, furniture, plants, etc. Anything in a room modifies the sound waves.

Sound engineers can measure the acoustical properties of a room, e.g., its resonance frequency, reverberations and echoes. Special rooms where all sound is absorbed by special walls are used for acoustical research and music recording. This is necessary to listen to or record the unmodified sound of a musician or instrument.

Effect processors can simulate many room types or acoustical environments. Any sound can be enhanced to sound like being emitted in a bathroom or in a cathedral.

1.1.3 Psycho-acoustics

Psycho-acoustics describe the effects specific to human hearing.

Human hearing is affected by a lot of parameters like frequency, direction, loudness and many others which make it difficult to achieve an exact acoustical representation. Furthermore, every person has a different perception of sound.

Mathematical models of the ear help to understand how hearing works and how sound must be modified to give the best possible reproduction for binaural hearing. This can be achieved by recording sound with different properties with a dummy head microphone. The resulting ear print is a set of functions depending on the frequency and

the direction of the sound. These functions are called Head Related Transfer Functions (HRTF) and are specific for each ear of the test person. A complete set of HTRF is optimally suited for a specific person. A subset of the HRTF is interchangeable and can be used with most other people, and therefore included in HRTF processors for 3D sound for use with headphones.

1.2 Surround sound

Surround sound is the general term for reproducing sound that comes or seems to come from different directions around the listener. Only a few surround sound systems can reproduce a 3D sound environment, mostly the sounds are located on a horizontal plane.

Various systems exist which can reproduce surround sound using different technologies, the most important and most common are listed in Table 1.1. A one channel audio signal, also called mono, cannot carry surround sound information. At least two channels are needed for spatialised sound.

Designation	**Discrete audio channels**	**Transmission format**
Mono	1	analogue or digital
Stereo	2	analogue or digital
Dolby Surround, Dolby Pro Logic	4	encoded into a stereo signal
Dolby Digital AC-3, DTS	1 - 8	digitally encoded
MPEG	1 - 8	digitally encoded
SDDS	6, 8	digitally encoded
Ambisonics	4-∞	mathematical functions

Table 1.1 Common audio formats

1.2.1 Stereo

Common surround systems are based on stereo sound with two audio channels used in audio tapes, compact discs, and radio and television broadcasts. The best acoustical impression using two loudspeakers for a listener is on a position in front and in the middle of them. By varying the gain of a sound related to both channels, its position can be changed to come from anywhere on a line between the left and the right speaker. Some devices create a pseudo-surround effect by adding a phase inverted signal of both channels to each other which let music sound wider than without this effect, but the surround effect is very limited.

A stereo signal can reproduce the recorded sounds of a room, but cannot reproduce the impression of being in that room. Only by utilisation of appropriate psycho-acoustical techniques and HRTF, room acoustics can be reproduced sufficiently. Although stereo based surround sound works well with headphones, it is difficult to achieve the same effect with two loudspeakers.

1.2.2 Dolby Surround and Dolby Pro Logic

The Dolby Surround system (1982) uses phase shifting of +/-90° to add information of a surround channel to a stereo signal. The decoder sends all signal that has a phase difference of 180° between the left and the right channel through a filter and time delay to the centre channel.

With the Dolby Surround Pro Logic system (1987), 4 channels (left, centre, right and surround) are encoded into a stereo audio signal. Matrix decoders detect the different

channels and send them to the amplifies for front left, centre, front right, and rear speakers. All sound with the exact same phasing in the stereo signal is sent to the centre speaker, everything with a phase difference of 180 degrees is sent to the rear speakers. A stereo only amplifier can still reasonably playback Dolby Surround Pro Logic encoded signals. The Dolby Pro Logic system has an active adaptive matrix decoder that enhances the channel separation. But it is preferable to have a principal sound direction with such a surround system. Too many different sounds from different directions will cause artefacts and lower the surround impression.

Although Dolby Surround and Pro Logic both offer 4 channels, they cannot be used very well for spatialised sound because 3 channels (left, centre and right) are located at the front, and the remaining channel, played by two speakers for better dispersion, provides all the sound from the back.

1.2.3 Ambisonics

A very sophisticated audio format is Ambisonics, originated by the mathematician M. Gerzon in the late 1970. The Ambisonics format is not a regular audio signal. It is a mathematical representation of the recorded surround sound and therefore not compatible to the usual stereo signal. Using an array of 4 microphones, acoustical environments can be recorded and later encoded to the Ambisonics B-Format. It consists of four component signal: a mono sound signal W, and three difference signals X (front-back), Y (left-right) and Z (up-down). The decoder reproduces surround sound with at least 4 speakers located in a quad array on a circular plane or more speakers located on a sphere around the listener.

With Ambisonics sounds can be reproduced in 3D. The Ambisonics B-Format can be transcoded to produce common stereo or multi-channel compatible media.

1.2.4 Digital surround sound

Current home cinema surround sound systems have digital surround sound decoders for 5.1 channels. Digital Multi-channel surround sound was proposed 1987 by the Society of Motion Picture and Television Engineers (SMPTE). *5* is the number of directional channels front left, centre, front right, left surround and right surround. The *.1* stands for a low frequency enhancement channel (LFE), which is to be played by a subwoofer, a loudspeaker for very low frequencies e.g. from special sound effects, explosions, earthquakes, etc. Bass management can be used to route the low frequency signals to a subwoofer or to the front speakers depending on the available speaker configuration.

The sound information is digitally encoded using perceptual coding for data compression. The most common formats are Dolby Digital - AC-3 (1992), Digital Theatre Systems - DTS (1993), Moving Picture Experts Group audio layer 2 - MPEG-2 (1994) and Sony Digital Dynamic Sound SDDS (1993).

While MPEG-2 video was chosen as the common video compression format for all DVDs, the MPEG-2 audio layer was only defined for PAL-DVD while the mandatory audio format for NTSC-DVD is Dolby Digital AC-3. In practice, MPEG-2 decoders have not been very successful for home entertainment. Nowadays PAL-DVDs also have only AC-3 and DTS tracks plus stereo PCM tracks. Movie theatres are usually outfitted with AC-3, DTS or SDDS systems.

Modern digital decoders are backwards compatible to stereo and Pro Logic and they are often enhanced with digital signal processors to simulate a variety of room acoustics. More sophisticated devices can be adapted to the configuration and the position of the speakers and to the room parameters.

With modern digital surround, 4-channel and 5.1-channel based spatialised sound is feasible, but the elevation of sounds still cannot be reproduced.

1.2.5 MPEG

MPEG (Moving Pictures Experts Group) defines standards for multimedia applications. The MPEG audio layers rather define audio compression standards than 3D sound. The latest developments describe the compression of multimedia contents for delayed reproduction, for instance by streaming. Real-time encoding is currently only possible with lossy compression that produces artefacts and therefore decreases the original image and sound quality. With a high communication bandwith the compression ratio can be reduced which gives a better quality.

1.2.6 Speaker setup

In case of single screen applications with graphics workstations or home entertainment, the user always sits at the same place in front of the screen. Navigation in virtual worlds only happens remotely with the usual input devices. The window to the virtual world is constrained by the screen dimensions.

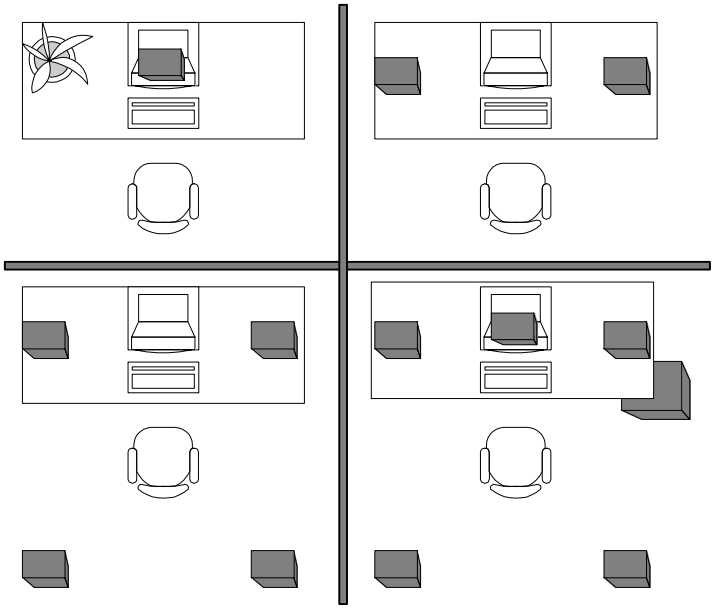

Figure 1.1 Common speaker set-ups: mono, stereo, quad, 5.1

Speaker systems can be positioned around the screen and around the user. The most common speaker positions for different numbers of speakers are shown in Figure 1.1.

Modern digital surround amplifiers have settings to compensate speaker positions that cannot be mounted at optimal positions. Additionally, they add room reverb effects or virtual loudspeakers.

When it comes to precise spatial sound positioning, the common surround sound systems with 4 or 5 main loudspeakers do not compete well with other solutions by principle.

The International Telecommunications Union proposed a speaker setup for 5.1-channel sound systems (ITU Recommendation 775), which is displayed in Figure 1.2. All speakers (except the subwoofer) are located horizontally at ear height on a circle around the listener. The centre speaker C is exactly in front of the listener at 0°. The left and right front speakers FL and FR are at +/-30° from the centre. The surround speakers RL and RR are at +/-110° of the centre speaker position. The subwoofer SUB used for bass and LFE can be placed anywhere, e.g., in a corner due to the fact that lower frequencies are perceived without noticing their direction.

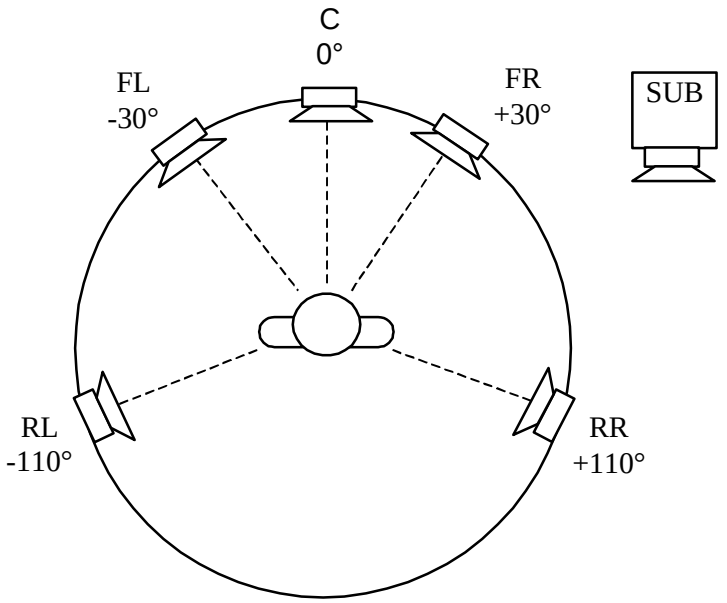

Figure 1.2 ITU recommendation 775 for 5.1 speaker setup

1.3 Sound with computers

In the earlier days of computers, the only noise they made was generated by ventilation and clicking of electromechanical components.

Later on, sound beeps were introduced, mostly to confirm input or show errors. With computer games and multimedia technologies, computers became able to generate digitised sounds like spoken text or music. The current generation of sound cards can generate, modify and reproduce multi-channel audio with digital signal processors (DSP) or simply using the CPU of the host computer and special software. Current consumer sound cards can reproduce 5.1 surround sound and make use of a DSP for spatialised sound in computer games.

Sound cards for professional music production have a large number of audio inputs and outputs and guarantee a low latency but do not have multi-channel 3D sound processing. A sound API helps the programmer to access the sound card with higher level functions. The operating system provides only base audio functions or none at all.

Table 1.1 shows some sound APIs that are available for the Microsoft Windows operating system.

API	**3D processing**	**remarks**
Windows Multimedia	no	base OS services for playing sounds
DirectSound	no	playing sound without 3D processing
DirectSound 3D	yes	3D positioning, distance attenuation, Doppler effect, basic reverb and chorus effects, different loudspeaker settings and HRTF
OpenAL	yes	3D positioning, distance attenuation, Doppler effect
EAX	yes	Room acoustic enhancement only, requires DirectSound3D or OpenAL
Sensaura	yes	HTRF and virtual surround only
Aureal 3D	yes	HRTF, extended room characteristics

Table 1.1 Windows Sound API

The Windows Multimedia API provides only simple playing of sound files.

The Microsoft DirectX API is a complete multimedia and game programming system. The DirectSound and DirectSound3D parts are specific for audio.

DirectX can use MIDI through the DirectMusic component. DirectSound3D has much more functionality than OpenAL but is not that portable and more complex to use.

OpenAL is an API for 3D sound intended for portable audio programming. It is available for Apple MacOS, Linux and Microsoft Windows. Unfortunately the development of OpenAL has been suspended. The sources have been made available as Open Source.

Creative Labs, a sound card manufacturer, made an API called Environmental Audio Extensions (EAX) to use the special functions of their 3D sound processing hardware. EAX adds 3D reverberation capabilities with reflection, occlusion, obstruction and other environmental effects to OpenAL or to the DirectSound component of DirectX. EAX can also be used for 5.1 surround sound processing. The most recent release of EAX 3.0 (EAX Advanced High Definition) offers better room simulation and extended parameter settings, but is currently only supported by the latest Creative Labs Audigy cards. The EAX 3.0 SDK is still not available though some games already support the EAX 3.0 effects through OpenAL and DirectSound3D.

Two other sound APIs, Sensaura and Aureal 3D, are specialised for headphone HRTF and virtual surround sound.

1.4 Virtual Reality

The term of virtual reality is generally used for any artificially generated worlds. Most of the time it means the display of rooms and worlds using three-dimensional (3D) computer graphics.

A basic VR system consists of a standard personal computer or graphics workstation. More elaborate systems provide stereographic projection to present objects in three dimensions. Professional systems totally enclose the user with a surrounding projection. This give them a more realistic impression and the possibility to interact in many ways with the objects inside the artificial world.

The quality of all these systems mostly depends on the performance of the computers employed to calculate and to display the data, and of the projection system.

The first graphics systems were only able to draw points and simple vector graphics that had to be calculated tediously before displaying them. Later, images of shaded and textured objects could be rendered with ray tracing programs. Even simple animations required long calculations. Current systems display calculated scenes in almost photo realistic quality in real time. Movie creators often use computers to edit their footage, to create stunning special effects and to mix real scenes with digitally created ones. But there is still no machine which can do photo realistic animated 3D graphics in real time.

1.4.1 Human senses

VR environments try to stimulate as many senses as possible to make the user feel like he is part of the virtual world. In addition to sight, hearing is the second most important

sense to be stimulated in VR environments. Scientific visualisation does not necessarily need acoustic effects, but sound adds a new dimension to a number of other fields. Architects for example like to walk through computer-generated models of their constructions and buildings to get an impression of how they will look when they will be built in reality.

Interaction with elements of the virtual world, e.g., switches, doors, and dynamic scenery, is unnatural without sound effects. Sound adds more life to visual worlds and gives an increased level of reality to the virtual environment. Experiments by Larsson P., Västfjäll D. and Kleiner M. (2001) have shown that audio significantly affects the visual perception.

For the touch force feedback devices for the hand and even entire suits have been developed. In a movie theatre the audience was stimulated by odours emitted according to specific scenes. But increasing the amount of the different stimulated senses also increases the complexity of required technical systems.

1.4.2 The CAVE environment

The CAVE (CAVE Automated Virtual Environment) is a projection-based VR system that surrounds the viewer with three or more screens. It is mostly used for large-scale visualisation of scientific data sets and for architectural design

The screens are arranged perpendicular to each other. To prevent shadows, the screens are usually projected on from the back (rear-projection). The ideal CAVE is a cube

where each side consists of a screen and which totally enclose the user by image projections.

The users of a CAVE stand inside the cubic projection area like shown in Figure 1.1. They wear stereo glasses to get the impression of spatial scenes. The two most common techniques for the stereo effect are active and passive stereo. Active stereo requires shutter glasses which are synchronised with the display of images alternating between the left and the right eye. Passive stereo works with polarised glasses and two projectors for each wall, each polarised differently. With stereoscopic projection, the user gets the impression to be in the virtual world. The glasses of one user are attached to a magnetic head tracking device with six degrees of freedom. As the tracked user moves inside the CAVE, the correct stereoscopic perspective projections are calculated for each wall according to his position and viewing direction. Using another position sensor located in a 3D mouse, the user can interact with the virtual environment, e.g., walk through a

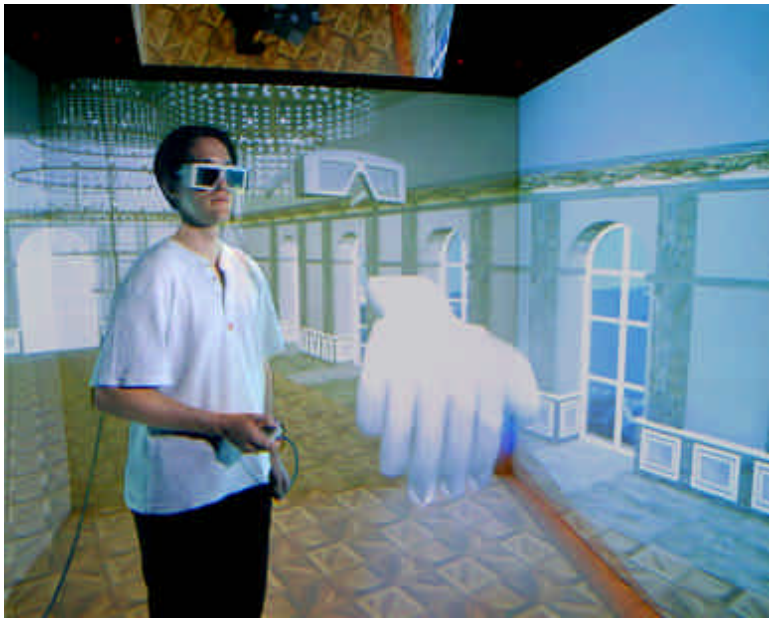

Figure 1.1 User using stereo glasses and magnetic tracking devices

world and open doors.

The most important difference to the usual workstation is that the user not only moves the world on the screen but also can move himself inside the world. He can actually turn his head to look left and right and walk around objects. The user stands in the CAVE with it co-ordinate system which represents a subset of the virtual world co-ordinate system, shown in Figure 1.2.

Three types of movements must be distinguished:

- The user moves physically in the CAVE room. The co-ordinates of the CAVE remain fixed relative to the world co-ordinates.

- The user moves in the virtual world using the pointing device. This translates the CAVE co-ordinate system relative to the world co-ordinates.

- The user moves inside the CAVE and inside the world. This affects both world and CAVE co-ordinate systems.

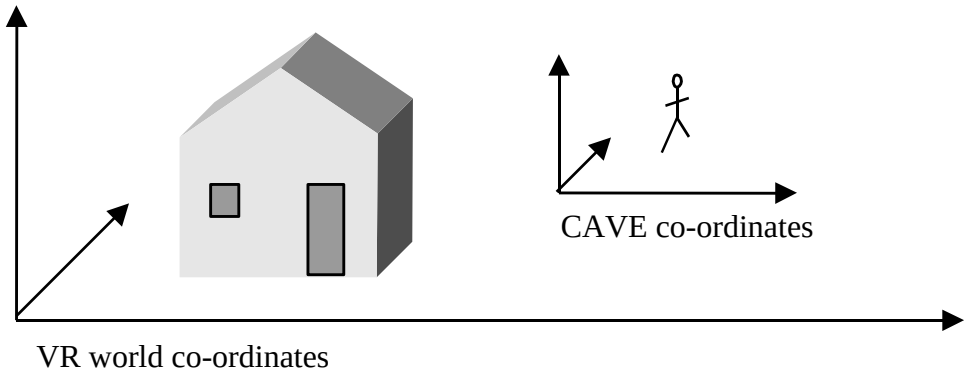

Figure 1.2 World and CAVE co-ordinate systems

1.4.3 Sound and VR

By combining images and sounds, an impressive atmosphere can be obtained, as modern multiplex and IMAX (especially the IMAX 3D) theatres show.

In an immersive virtual environment like a CAVE, the speaker setup is more difficult than for a workstation with just one screen. Where rear projection walls are used, the speakers cannot be placed exactly at the positions recommended by the ITU 775. In a typical 4-screen CAVE, the front speakers cannot be installed at the height of the ears, because if installed behind the walls these would absorb too much sound, and if installed in front, they would disturb the image projection. So they have to be installed on top of the front projection wall. The rear speakers could be installed at ear height at the back if there is no rear projection. But this means the acoustical plane spanned by the speakers would not be horizontal like recommended. Because of the mirror used for the floor projections, a centre speaker cannot be mounted well. Using only 4 speakers would be more suitable.

If magnetic position tracking is used, the speakers require magnetic shielding to avoid interference with the magnetic field.

For real 3D sound perception, additional audio channels would be necessary, e.g. one speaker at each of the eight corner of the CAVE. Current consumer surround amplifiers only have 5.1 channels, some high end devices already have 6.1 channels, using the Pro Logic Matrix technology, to add a surround centre channel to both the surround left and right channels, but this does not improve the sound impression very much, compared to the earlier improvements from stereo to Pro Logic and to 5.1 surround. And the sounds are all still located in a horizontal plane.

The speakers are to produce sounds which simulate a virtual sound source relative to the position of the user in the virtual world and of the direction where he looks (see Figure 1.1). Using a 5.1 system, the sound is correct as long as the listener stays in the centre of the CAVE and looks at the front screen. But in reality the user walks around and looks into different directions. The 5.1 speaker system cannot perfectly reproduce a sound which moves around the listener because, e.g., if the listener is close to a wall, he cannot

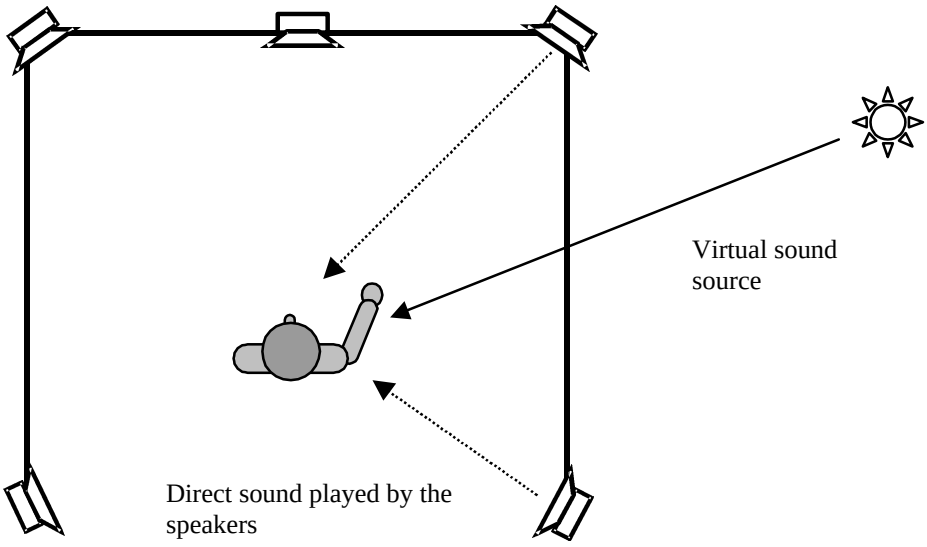

Figure 1.1 Virtual sound source

hear sound from outside of the CAVE, but only from the closest corners.

There is enough room in a CAVE to allow several people to watch the same scene. They always will see the images and hear the sound according to the main user orientation. If everyone in the CAVE would have his position and orientation tracked by the visualisation system, individual sound could be generated and would necessitate each listener to wear headphones. The current computer technology already has wireless pointing devices and headphones, so it would be feasible. Headphone usage with HRTF gives a good 3D impression but is not likely to be used in a CAVE because only one

user is tracked and the sound is generated for him only. In this thesis, the number of tracked users will be limited to one.

With a 5.1 surround system, the height information of a 3D sound cannot be presented. Perraux, Boussard and Lot (1998) have presented a way to use a 5.1 speaker layout to reproduce 3D sound in the horizontal plane and a way to reproduce elevated sounds with Ambisonics and Vector Base Panning. Ambisonics has a better surround reproduction and includes the height but it necessitates special Ambisonics encoders and decoders.

1.4.4 Virtual Reality Modeling Language

The International Standards Organisation (ISO) has set up a branch for the Virtual Reality Modeling Language (VRML) which is widely used to create VR worlds.

VRML is a standardised file format to describe interactive scenes in a three-dimensional space. It is used for many different types of interactive multimedia applications, e.g., for scientific visualisation, multimedia presentations, and demonstrations of architectural design studies.

In VRML, objects are declared as nodes. The relevant nodes for audio are Sound, AudioClip and MovieTexture. Since VRML version 2.0, sounds can have locations in three-space.

The Sound node holds the properties of a sound source in the VR world. These are the position, the orientation and the minimum and maximum listening distances for back and front side of the sound position. In the minimum ellipsoid, the sound has the maximum intensity. Moving from minimum to maximum distance constantly reduces the intensity from its maximum to 0, as shown in Figure 1.1:

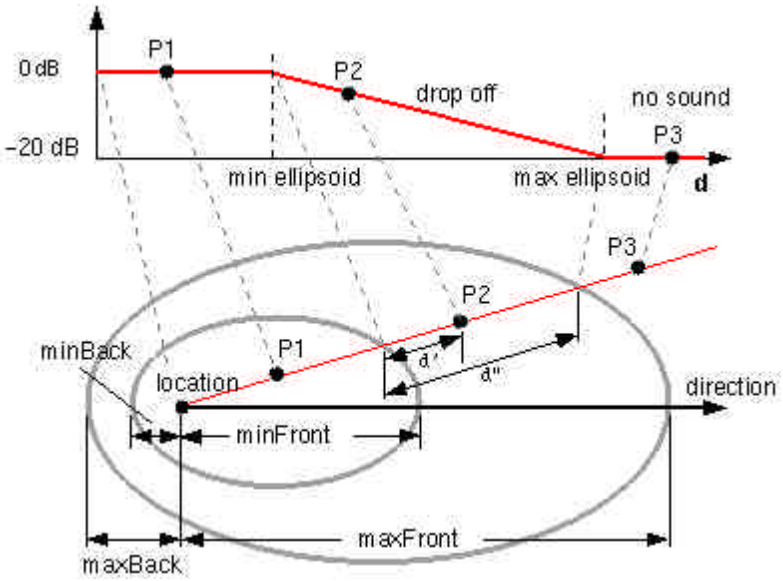

Figure 1.1 VRML sound node geometry

Here is the VRML declaration of a Sound node, the parameter names are bold:

```
Sound {
      exposedField    SFNode      source       NULL     # AudioClip or MovieTexture
      exposedField    SFVec3f     location     0 0 0    # (- ,   )
      exposedField    SFFloat     intensity    1        # [0,1]
      exposedField    SFVec3f     direction    0 0 1    # (- ,  )
      exposedField    SFFloat     priority     0        # [0,1]
      field           SFBool      spatialize   TRUE     # TRUE or FALSE
      exposedField    SFFloat     maxBack      10       # [0,   )
      exposedField    SFFloat     maxFront     10       # [0,   )
      exposedField    SFFloat     minBack      1        # [0,   )
      exposedField    SFFloat     minFront     1        # [0,   )
}
```

The AudioClip node describes a sound file to use with the Sound node. While the

Sound node is the emitter, like a loudspeaker in the virtual world, the AudioClip node is the generator of the sound. Here is the AudioClip declaration:

```
AudioClip {
      exposedField    SFString    description           ""
      exposedField    SFBool      loop                  FALSE
      exposedField    SFFloat     pitch                 1.0       # (0,   )
      exposedField    SFTime      startTime             0         # (- , )
      exposedField    SFTime      stopTime              0         # (- , )
      exposedField    MFString    url                   []
      eventOut        SFTime      duration_changed
      eventOut        SFBool      isActive
}
```

Similar to the AudoClip node, the MovieTexture node describes a movie file. It can be used as sound generator if the movie file contains an audio channel:

```
MovieTexture {
      exposedField    SFBool      loop                  FALSE
      exposedField    SFFloat     speed                 1.0       # (- , )
      exposedField    SFTime      startTime             0         # (- , )
      exposedField    SFTime      stopTime              0         # (- , )
      exposedField    MFString    url                   []
      field           SFBool      repeatS               TRUE
      field           SFBool      repeatT               TRUE
      eventOut        SFTime      duration_changed
      eventOut        SFBool      isActive
}
```

A VRML browser application reads the file and loads all textures and sounds. As the user moves inside the virtual world, the browser displays images and plays the respective sounds. The user can trigger actions by touching elements of the world.

1.5 Computer Networks

Networks are a common way to connect at least two computer systems with each other. Today the most common standard is Ethernet (IEEE 802.2)., which is both a cabling and transmission protocol standard. Ethernet frames can be used to carry data frames from other protocols like TCP/IP. The global internet uses TCP/IP (Transport Control

Protocol/Internet Protocol) for reliable connection oriented services, and UDP (User Datagram Protocol) for connectionless oriented services like audio and video streaming.

Some computers offer specific services to other computers like printing documents or storing files. Such a system is called a server. The users of a server, the clients, connect to the server, make some requests, wait for the answers from the server and finally disconnect.

Sockets, which are service specific connection points, are commonly used for network applications like internet file transfer or browsing web pages.

1.6 Related work

Much work has been done related to multi-channel audio and VR. Some interesting publications are listed in the following:

- AudioFile

  AudioFile is a network-transparent system for distributed audio application done by Levergood, Payne, Gettys et al. (1993) at the Digital Equipment Corporation Cambridge Labs. AudioFile defines a protocol for the communication between clients and the server. The AudioFile system only mixes and plays sounds. There is not support for surround sound or 3D audio.

- CARROUSO

  CARROUSO stands for creating, assessing and rendering in real-time high quality audio-visual environments in MPEG-4 context using wave field synthesis (WFS). The aim of this project is break the constraints of the common surround sound systems with their limited 3D capabilities. It uses special

recording, encoding and decoding techniques and a special array of WFS loudspeakers. See Brix, Sporer and Plogsties (2001) for details

- OpenAL++

  OpenAL++ is a set of C++ classes for a better integration of the OpenAL API with C++ programming. OpenAL++ was made by Hämälä (2002) at the same time the Audio Server project was carried out. OpenAL uses a set of open source libraries like PortAudio which provides a common interface to the sound related functions of various operating systems, and CommonC++ for portable thread and network socket programming. OpenAL++ integrates all the OpenAL functions plus the ability of using microphone and line input. OpenAL++ does not make use of the EAX library, which is now available for use with OpenAL. If OpenAL++ was released earlier, it could possibly have been extended with EAX and be used as part of the Audio Server project.

- VSS

  The Virtual Sound Server was developed by Das S. and Goudeseune C. (1994), at the National Center for Supercomputing Applications (NCSA) and the University of Illinois at Urbana-Champaign VSS is a platform-independent software package for data-driven sound production controlled from interactive applications. VSS was built to control the HTM framework for real time sound synthesis developed at the Center for New Music and Audio Technologies at UC Berkeley. It's main purpose is the sound generation for use as a music instrument. Although it knows different audio channels, it has no real 3D sound

processing and therefore cannot be used very well for VR applications. But it is a good example for a distributed system running on different platforms.

2 The Audio Server

This chapter describes the actual Audio Server project. It is split up in descriptions of the server and the client API, and shows a prototype implementation.

Figure 2.1 shows the basic schema of a setup using the Audio Server. A client, e.g. a visualisation system, sends its requests to the Audio Server across a computer network. The Audio Server processes the requests and sends the requested sounds to the audio amplifier by the soundcard .

2.1 Server

The server represents the actual Audio Server. The hardware consists of a computer equipped with a soundcard, connected to an audio amplifier, and of a network interface. The server software handles network requests and manages the sound generation through the sound hardware. The client communicates with the Audio Server through a protocol presented later.

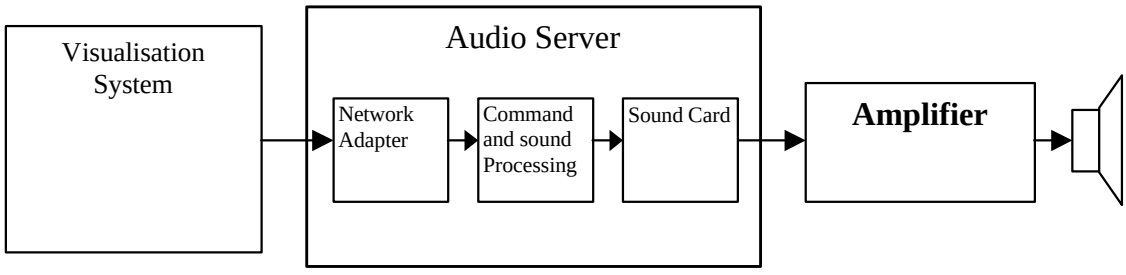

Figure 2.1 Audio Server schema

The Audio Server can be used with different types of application like shown in Figure 2.2. The Audio Server may directly be connected to a graphics workstation or via a visualisation system. If the graphics workstation is powerful enough, the Audio Server could run in parallel to the visualisation application.

2.1.1 Hardware and software requirements

The following defines the requirements to the Audio Server has to fulfil.

- The Audio Server shall be an independent system.

  To avoid restrictions caused by hardware, operation system and software of the client system, e.g., a visualisation device, the Audio Server shall be realised as

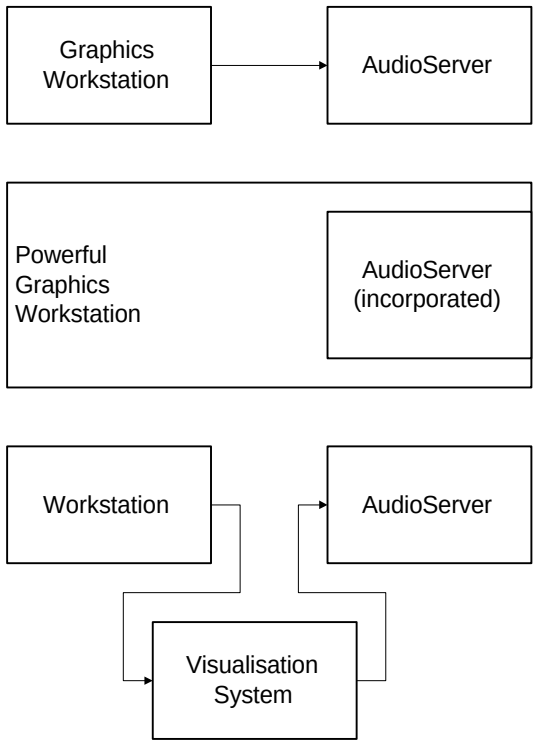

Figure 2.2 Different Audio Server system configurations

separate system. Since most common distributed systems use computer networks, the client-server communication can typically be realised via TCP/IP sockets. Since the socket port is also locally available, a visualisation application could then run on the same machine as the Audio Server if the system used for the Audio Server is powerful enough. Being a network server, the Audio Server could handle multiple client requests. As separate system, the Audio Server requires a network interface adapter or other fast connection to the client.

- The communication protocol shall be simple.

  Especially in experimental environments it is difficult to find out where exactly errors are located. The communication protocol should be simple enough that the Audio Server functions can be tested without programming a complex test application, but, e.g., by a Telnet connection. Instead of using binary numbers, the commands should be represented by readable text messages in the form

  COMMAND [PARAMETER] [PARAMETER] ... [PARAMETER]

- The communication protocol shall be extendable.

  New commands should be added without completely reprogramming the Audio Server. The concept of classes and separate source files for different purposes is very suitable for expandable applications.

- The Audio Server shall provide sound file management.

  To avoid delays during the initialisation by the client, sound files should be stored on the hard disk of the Audio Server as long as they are needed. The files should be stored in a temporary storage that may be deleted on request.

- The Audio Server shall have a soundcard.

  The Audio Server is not only a software project, it requires a minimum of hardware for sound output. To get sound, any computer with an audio device is suitable. But to have minimum support for 3D sound a PC with a stereo sound card is required. The optimum system is a state-of-the art computer and a soundcard with an integrated DSP (Digital Signal Processor) for spatial processing. The card should have multi-channel analogue or digital outputs to connect them to a multimedia speaker set or to a multi-channel amplifier and a 5.1 speaker set. The sound card and the drivers should support spatial sound mixing and processing. While all recent consumer sound cards support DirectSound 3D and OpenAL (Open Audio Library), EAX (Environmental Audio Extensions) is not available for every card.

- The Audio Server shall make use of the resources available on the actual system.

  As a separate system, the Audio Server should make use of the actual hardware, e.g., it could run on a laptop (which today still have poor sound capabilities, typically only stereo output) or on a computer equipped with a good soundcard with 3D sound processing hardware. If a 3D sound card is available, the Audio Server should make use of the 3D sound functions, even if there is an additional stereo card installed.

Regardless of the type of sound card, the system should meet the requirements recommended by the sound card manufacturer. For instance, for the Audigy card, these are a PC with at least a 500 MHz CPU, 128 MB RAM and 1 GB free space on a fast hard disk.

2.1.2 Initialisation and main event loop

The server first queries the sound hardware for its properties and available functions. Depending on the type of the soundcard and of its drivers, different functions may be available, e.g. stereo panning, 3D sound, or even enhanced room acoustics. DirectX, OpenAL or any other sound API used by the actual implementation must have been installed prior to running the Audio Server, otherwise no 3D sound functions will be available. Figure 2.1 shows the initialisation steps and the main event loop.

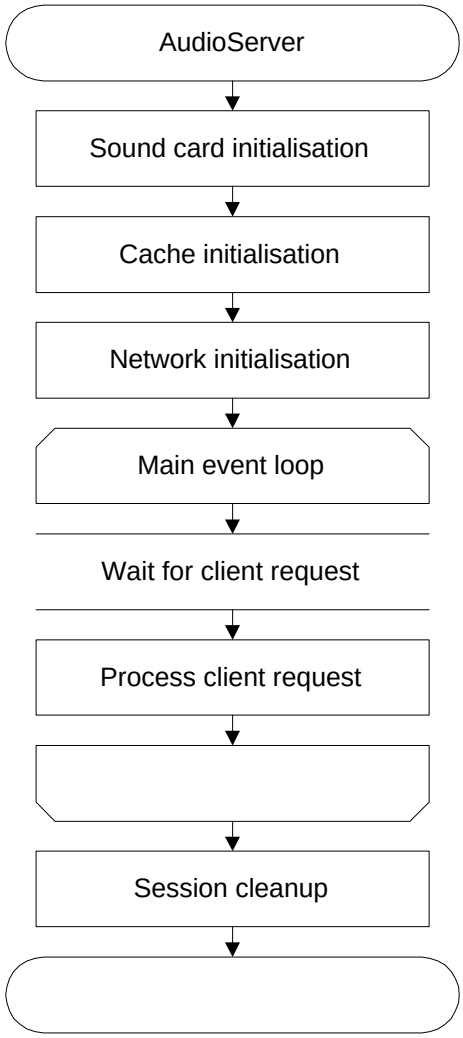

Figure 2.1 Main event loop

Then the sound cache directories for the default sounds and for the dynamic sound cache are tested. If they don't exist, they have to be created. The available disk space is tested. In case the disk is full, some files must be deleted in order to keep free disk space for files later being sent by the client.

Figure 2.2 describes the cache initialisation process.

Larger files take a rather long time to be transmitted, which is especially annoying during the programming and testing phase of a VRML world. Therefore sound files

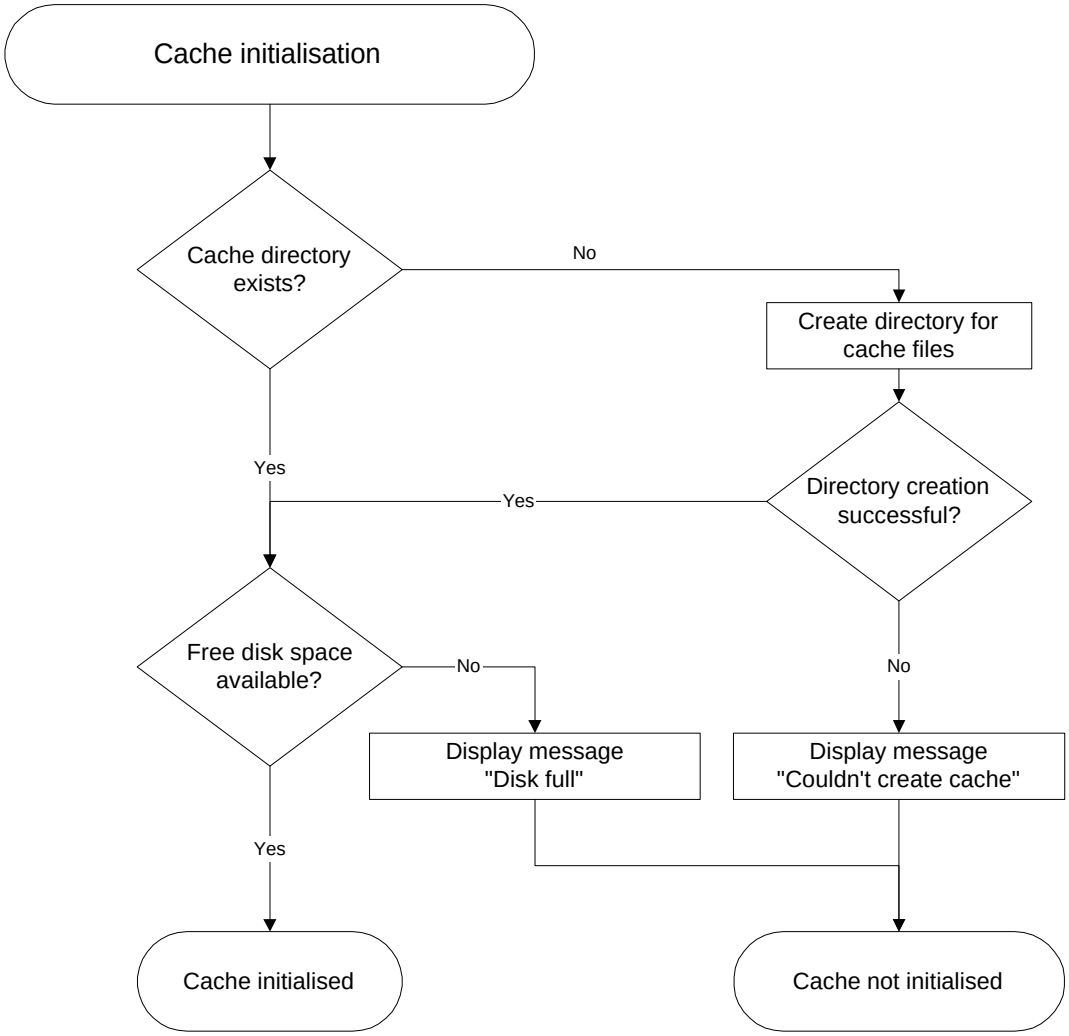

Figure 2.2 Cache initialisation

should be kept on the Audio Server as long as possible. The sound file cache permits to hold a number of recently used files. If a requested file is in the cache, the client will get a handle instantly if the file has previously been uploaded. To ensure a user really gets the sounds he wants, for instance if he changed the contents of a sound file, the cache can be emptied before starting a new session. The Audio Server also keeps a set of commonly used files which are not part of the cache. These files are permanently available in a separate directory and can be used to acoustically enhance frequently occurring user interface actions.

Finally the network interface is initialised and a socket port is opened. The Audio Server then waits for clients to connect.

After all the initialisation steps the Audio Server loops in the main event loop waiting for and processing client requests. The event processing includes network events, sound file management and sound API calls. The basic client operations are the connection, file handle request, operations on the handle like playing and stopping the sound, and disconnection.

2.1.3 Sound file management

Before the client can start playing a sound, it must request a handle for a sound file from the Audio Server. If the handle is invalid, the sound file is not available and must be made available to the Audio Server. This can happen by simply copying the file manually or by transmission over the network connection.

When the client requests a handle for a sound file, the Audio Server first checks if enough free sound buffers are available, which is restricted by available memory.

Then the server tests the cache files like shown Figure 2.1. The server looks into the default sounds directory. If the file is available there, the file is loaded and a positive handle value is sent back to the client. If the file is not in the default sound directory, the

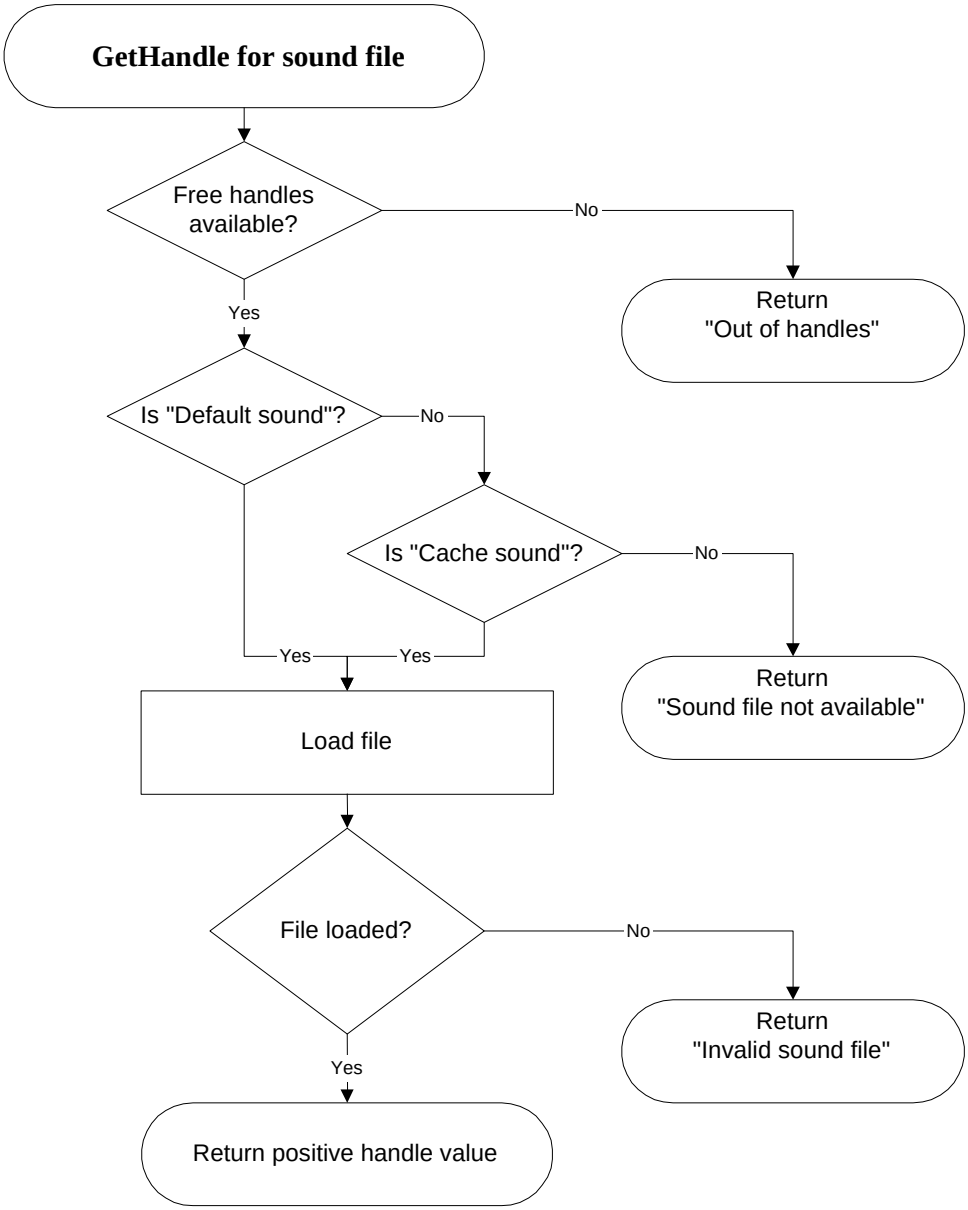

Figure 2.1 Handle request operation

Audio Server looks for it in the cache directory. If it is in there, the Audio Server loads this one and sends a positive handle value. If not, the returned handle will be –1 and the client may send the sound file to the Audio Server. If the file can't be loaded, the Audio Server sends an error message to the client

2.1.4 Sound source management

Because only a limited number of sound sources is available for 3D hardware processing, all the requested sound data buffers cannot immediately be assigned to sources. Basically this must only happen when a sound is to be played. The dynamic creation of sound sources is demonstrated in Figure 2.1.

In OpenAL there is no automatic notification mechanism that is triggered when a buffer was totally played. The status of a source can be queried instead, e.g. for the number of buffers in the queue and for the number of buffers already processed. If the number of processed buffers is equal to the number of buffers in the queue the source has finished playing all of them. Looping buffers always will be marked as queued but never as processed until the source is stopped. in this case the rest of the buffer is played and finally marked as processed. By supervising the source buffer properties, an already used source that has finished playing can be reused instead of deleting and recreating it.

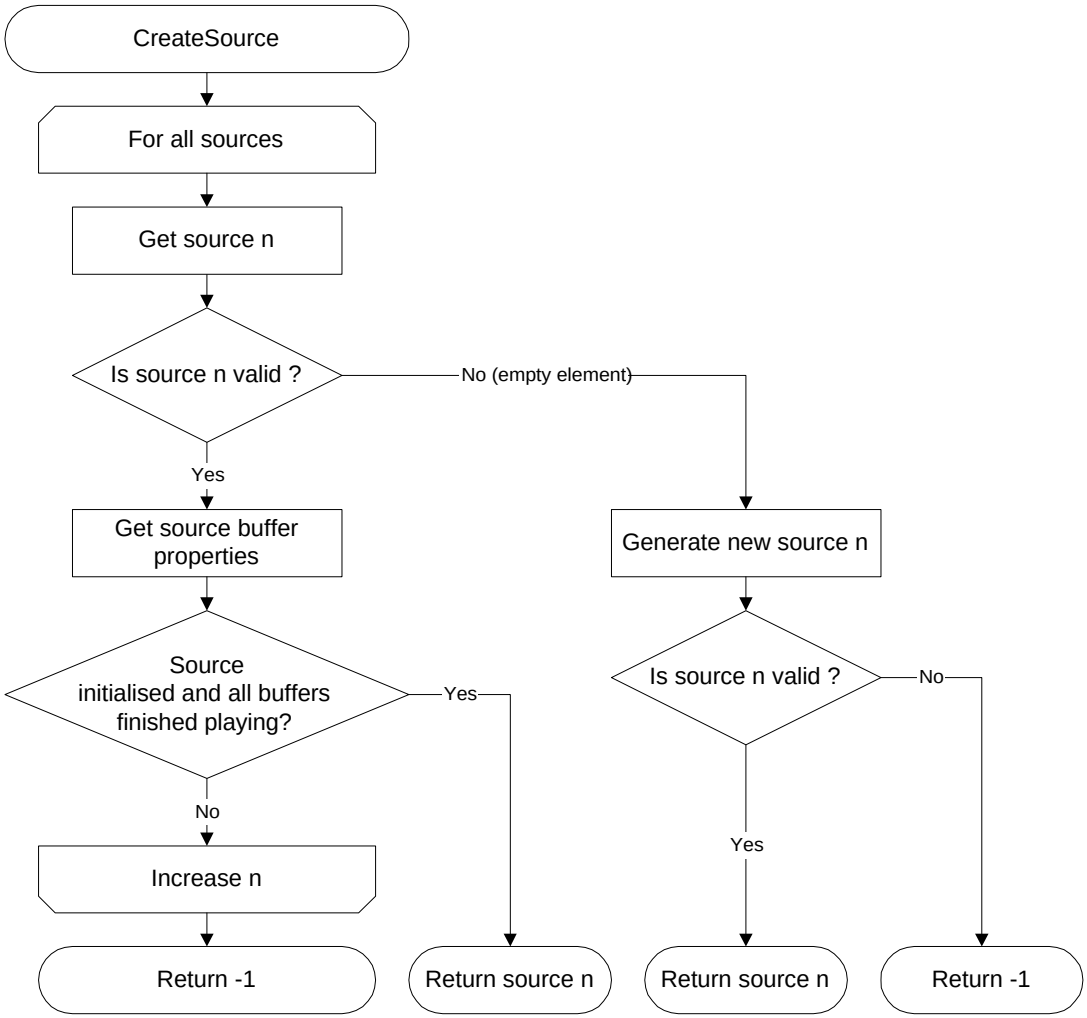

Figure 2.1 Dynamic allocation of sound sources

2.2 Client API

The client application connects to the Audio Server and sends requests which are processed and answered according to their type. Even a Telnet client could be used to do some tests since the protocol consists of simple ASCII commands.

The client API defines a set of structures and functions which can be used by an application to communicate with the Audio Server. Furthermore, the API describes the necessary steps to achieve a successful communication.

The client must first create a socket and connect to the Audio Server socket port. Then it can start the communication. The communication is done by ASCII text commands. Some commands accept parameters, and some are replied to by the Audio Server. And some commands are only available in special cases.

2.2.1 General commands

The general commands of Table 2.1; except the EAX specific commands, are always available regardless of the chosen sound API. Each comment is described in the following.

General commands	**Parameter**
TEST	none or number
PLAYFILE	file name of sound
GETHANDLE	file name of sound
RELEASEHANDLE	handle number
PLAY	handle number
PLAYLOOPED	handle number
PUT_FILE	File name of sound, length of file
SET_VOLUME	volume
SET_SOUND	handle number, Sound specific command and its parameters
SET_SOUND_EAX	handle number, Sound specific EAX command and its parameters
SET_EAX	General EAX setting command and its parameters
STOP	handle number
QUIT	none

Table 2.1 General commands

- TEST (optional)

  Syntax: TEST <number>

  The TEST command provides a simple check if the Audio Server is correctly connected with the amplifier. The TEST command only uses base operating system multimedia services. The Audio Server can play a beep on the internal speaker and system sounds according to the parameter number.

- PLAYFILE (optional)

  Syntax: PLAYFILE <file name>

  The PLAYFILE command is similar to the TEST command. It plays the passed sound file using base operating system multimedia service.

- GETHANDLE

  Syntax: GETHANDLE <file name>

  The GETHANDLE command requests a handle for the current file from the Audio Server. Each handle can be assigned to the same or to different sound files. A handle is a reference number to a sound source on the Audio Server. The parameters can be set separately with the SET_SOUND command. The Audio Server answers with a number. A positive value is a valid handle. A negative value is an error with the following meaning: -1: The requested file is not available on the Audio Server and should be sent with the PUT_FILE command. –2: The requested file could not be loaded.

- RELEASEHANDLE

  Syntax: RELEASEHANDLE <handle number>

  All resources referenced by the passed handle are freed on the Audio Server.

- PLAY

  Syntax: PLAY <handle number>

  The sound referenced by <handle number> will be played with the parameters previously set for this sound.

- PLAYLOOPED

  Syntax: PLAYLOOPED <handle number>

  The sound referenced by <handle number> will be played and looped with the parameters previously set for this sound.

- PUTFILE

  Syntax: PUTFILE <file name> <file length>

  This command tells the Audio Server it has to create a file with the passed name in its cache and it has to expect <file length> data blocks. The file has to be transmitted as a sequence of data buffers over the socket connection immediately after the PUTFILE command. Existing files will be overwritten. A file will stay in the cache until it is full or emptied.

- SET_VOLUME

  Syntax: SET_VOLUME [[<volume>] | [<volume left> <volume right>]]

  Sets the volume of the current audio output. The volume of the main audio channels can be set separately.

- SET_SOUND

  Syntax: SET_SOUND <handle number> <parameter name> <parameter values>

  Sets a parameter for the sound referenced by a handle.

- SET_EAX

  Syntax: SET_EAX <parameter name> <parameter values>

  Sets a global parameter for the EAX extension.

- SET_SOUND_EAX

  Syntax: SET_SOUND_EAX <handle number> <parameter name> <parameter values>

  Sets an EAX specific parameter for the sound referenced by <handle number>.

- QUIT

  Syntax: QUIT

  Signals the Audio Server the client will close the connection. All previously requested handles will be released.

2.2.2 Sound parameter commands

Some sound specific commands set the parameter values of the sound which has been assigned to a handle. All parameter settings have to be sent with the SET_SOUND command. They are listed in Table 2.1 and described in the following:

Syntax for sound parameter commands:

SET_SOUND <handle number> <parameter name> <parameters>

- CONE_INNER_ANGLE

  Syntax: SET_SOUND <handle number> CONE_INNER_ANGLE <angle>

  The CONE_INNER_ANGLE parameter sets the inner angle of the sound cone. A value of 360° makes the source omni-directional, which is the default setting for new sound sources.

- CONE_OUTER_ANGLE

  Syntax: SET_SOUND <handle number> CONE_OUTER_ANGLE <angle>

  The CONE_OUTER_ANGLE parameter sets the outer angle of the sound cone. A value of 360° makes the source have no angle-dependent attenuation zone, which is the default setting for new sound sources.

- DIRECTION

  Syntax: SET_SOUND <handle number> DIRECTION <x value> <y value> <z value>

Sound commands	Parameter
CONE_INNER_ANGLE	Inside angle of sound cone in degrees
CONE_OUTSIDE_ANGLE	Outside angle of sound cone in degrees
DIRECTION	3D direction vector
DIRECTION_RELATIVE	Horizontal angle in radians
LOOP	Sound looping (1 = on, 0 = off)
MAX_DISTANCE	Maximum distance within the sound can be heard
MAX_GAIN	Maximum gain
MIN_GAIN	Minimum gain
PITCH	Pitch (0.5 = half pitch, 2 = double pitch)
POSITION	3D position vector
REL_DIR_VOL	Relative direction angle in radians, sound source gain
VOLUME	Gain
VELOCITY	Velocity

Table 2.1 Sound specific commands

The DIRECTION parameter sets a vector in which direction the 3D sound points. This affects the sound cone and requires the parameters MIN_DISTANCE and MAX_DISTANCE to be set. A listener outside the sound cone will not hear the sound. The default value is a direction of <0,0,0> which means the sound source has no specific direction and no angle dependent attenuation.

- DIRECTION_RELATIVE

  Syntax: SET_SOUND <handle number> DIRECTION_RELATIVE <angle>

  The DIRECTION_RELATIVE parameter sets the 3D sound source position on a horizontal circle of radius 1 around the listener. This parameter is helpful for VRML browsers where the sound source position is not available as a 3D vector.

- POSITION

  Syntax: SET_SOUND <handle number> POSITION <x value> <y value> <z value>

  The POSITION parameter sets the 3D position of a sound source.

- REL_DIR_VOL

  Syntax: SET_SOUND <handle number> REL_DIR_VOL <angle> <gain>

  The REL_DIR_VOL parameter sets the relative direction of a sound source without position and its gain with one message.

2.2.3 EAX related commands

EAX is the Environmental Audio Extensions library from Creative Labs. It adds room processing to the sound output and has several switches and settings. If EAX us

available (if the soundcard can do 3D processing and if EAX drivers are installed) the following command can be used to affect the EAX settings.

EAX commands	Parameter
ENVIRONMENT	Preset number

- ENVIRONMENT

  Syntax: SET_EAX ENVIRONMENT <preset number>

  The ENVIRONMENT parameter sets the current EAX parameters to one of the 26 presets listed in Table 2.1, which are available in all EAX versions. Beyond these presets, individual settings can be made.

2.3 Server and client prototype implementation

The prototype implementations of both server and client have been realised to demonstrate and to test the functionality of the Audio Server in different environments. It is only a matter of time to implement all the functions available by the DirectX,

Nr.	Name
0	Generic
1	Padded Cell
2	Room
3	Bathroom
4	Living Room
5	Stone Room
6	Auditorium
7	Concert Hall
8	Cave
9	Arena
10	Hangar
11	Carpeted Hallway
12	Hallway

Nr.	Name
13	Stone Corridor
14	Alley
15	Forest
16	City
17	Mountains
18	Quarry
19	Plain
20	Parking Lot
21	Sewer Pipe
22	Underwater
23	Drugged
24	Dizzy
25	Psychotic

Table 2.1 EAX presets

OpenAL and EAX API into the Audio Server software. They can be implemented step by step when the functionality is required.

The server prototype is a 32-bit application for Microsoft Windows (NT 4.0 or 2000), using the most important commands of the OpenAL sound API and some EAX functions for room acoustics.

For the testing, a PC running Windows 2000 with a Creative Labs Audigy card was used.

See Chapter 10, Appendix C, for details about implementation of the server and of a client.

2.3.1 Missing sound sources

During the tests it happened that OpenAL only could play 30 sound sources in parallel, although the Audigy card has a hardware processor for 32 sound sources. The missing two sources are probably used by the operating system.

The DirectX Caps Viewer application (see Figure 1.1) from the DirectX SDK says that the Windows operation system reserves one hardware buffer, which is indeed used by the sound card driver itself for primary sound output. The Audigy mixer panel lets the user modify the primary output sound level.

In the screen shot, the value *dwMaxHw3DAllBuffers* shows the maximum number of buffers processed in hardware in total, which is 32 for the Audigy card. The value *dwFreeHw3DallBuffers* is the number of actually available buffers.

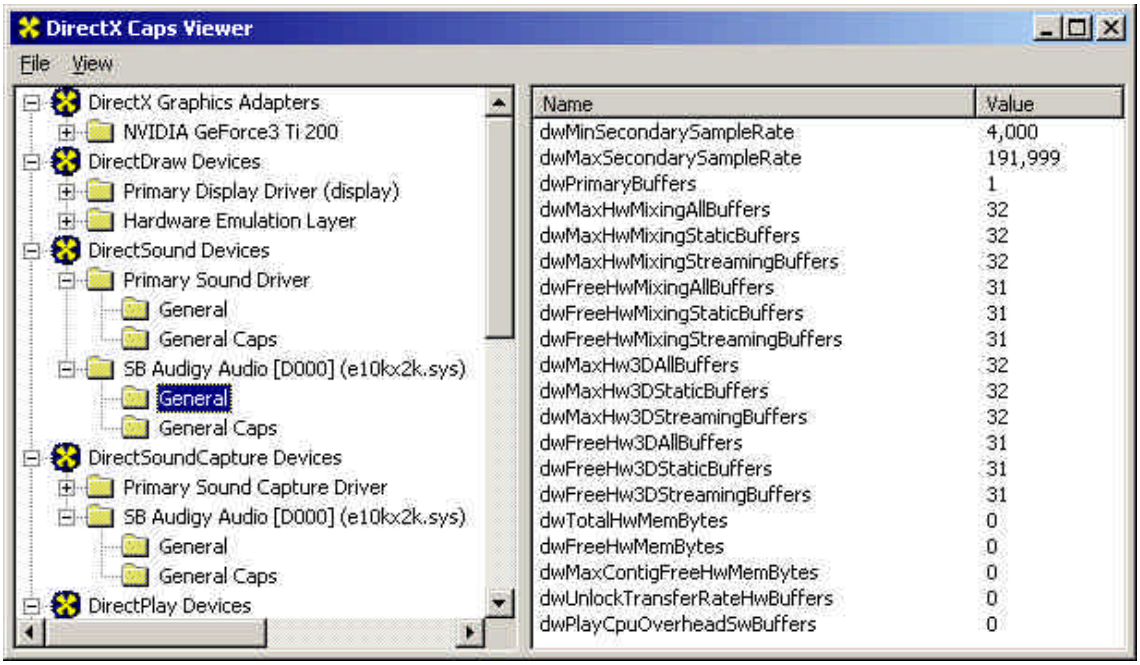

Figure 2.1 DirectX Caps Viewer, Sound card properties

But where was the missing one sound source? It must happened with OpenAL.

By browsing the OpenAL sources, a restriction could be found in the file alc.cpp inside the function alcOpenDevice. There the number of 3D sound buffers available to OpenAL is clamped to a maximum of 31.

The original sources were compiled and linked to a new OpenAL32.DLL, and finally 31 sound sources were revealed. Unfortunately, the function that should return the textual meaning of OpenAL errors then did not work correctly with that file.

2.3.2 Real application with VRML worlds

A client written as a console application was used to test the implemented functions step by step. Additionally, the Audio Server was be tested in a real application environment.

The Stuttgart Supercomputing Centre (HLRS, Höchst-Leistungs-Rechenzentrum Stuttgart) at the University of Stuttgart has a visualisation department which uses a CAVE with 3 projected walls and a projected floor. Their Collaborative Visualisation and Simulation Environment (COVISE) is an extendable distributed software package which integrates supercomputer based simulations, post-processing and visualisation with collaborative work.

The VR functions are realised through the software COVER (COVISE Virtual Environment), which supports VRML2.0/VRML97 data sets. COVER is programmed in C++ and could therefore easily be extended with the Audio Server client functions.

In the HLRS CAVE, the predominant application which uses audio is the visualisation of architectural work created with VRML.

Although both VRML and OpenAL know about the sound sources in their own worlds, combining both is a challenging task. The VRML browser uses different methods to play sound which need adequate solutions.

- Non-spatialised sounds

  These are always played without 3D information and shall not be included in the soundcard 3D processing. This is needed for voice messages and sounds which have to be heard without modification.

- 3D position

  The 3D position of a sound can be relative to the listener position or relative to the world.

- Horizontal direction only

  Here the VRML browser gives a horizontal angle around the centre of the CAVE instead of a sound source position. An angle of 0° means the sound comes from the middle of the front screen, -90° from the middle of the left screen, and +90° from the middle of the right screen. The angle must be converted into a sound source position relative to the Audio Server listener position. This can be done with simple trigonometric functions (angle is in radians):

  ```
  x = (float)sin(angle);
  y = 0.0;
  z = (float)cos(angle);
  ```

  The sound will always have distance of 1 to the listener. Distance can be simulated by setting the gain of the sound, which is available from the VRML browser.

- Velocity and Doppler effects

  The Doppler effect is a phenomenon where the pitch of moving sources varies depending on speed and direction. A source moving towards the listener has a higher pitch, a source moving away from the listener has a lower pitch than the non moving source (see Figure 2.1). This an be experienced well with the siren of a moving police car.

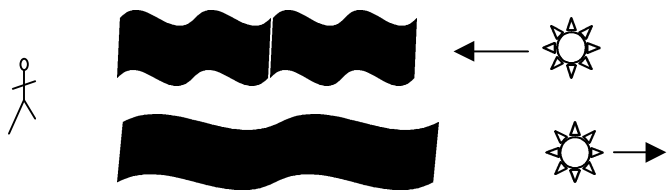

Figure 2.1 Doppler effect of moving sound sources

The Doppler effect can be mathematically represented and is used by OpenAL with the following formula:

$$f' = DF * f * \frac{c - v_l}{c + v_s}, with$$

f	*original sound pitch*
f'	*Doppler effect pitch*
c	*Speed of sound*
v_l	*Velocity of listener*
v_s	*Velocity of sound source*
DF	*Doppler factor*

In OpenAL, the velocity parameter of a sound and of the listener have to be explicitly set to generate Doppler effects. They are 0 by default.

The VRML browser only outputs the absolute velocity in meters per second, but OpenAL expects the velocity as a 3D vector. One could program the VRML browser to output a velocity vector, or the Audio Server could calculate a velocity vector with the last known and new position. Since the pitch of a sound can be set separately of the velocity, the Doppler effect can directly be applied as pitch change:

`pitch = 1 – velocity / c0`

A negative velocity value results in higher pitch for a sound source moving towards the listener, and a positive value results in lower pitch for a sound source moving away from the listener. A stationary sound source has no velocity, its pitch will not change, unless the listener moves himself.

The absolute velocity is set in relation to the speed of sound in air c_0, with c_0 = 343.3 m/s. Valid values for the pitch in OpenAL range from 0.5 (which equals half the original frequency) to 2.0 (double frequency).

3 Evaluation

Beyond the basic functional tests of the Audio Server during the programming phase, an evaluation with 5 different test persons was done to give an impression whether or not the Audio Server works as it is supposed to and to find out if there is still something to improve. 4 loudspeakers were used, 2 in the front corners and 2 on the back side of the CAVE.

3.1 Position test

The test person stays at different points in the CAVE. The sound of a snare drum is played repeated with 1s and without any effect at a random virtual position on a circle with a radius of 10 feet around the listener. The minimum angle interval is 15°. The test person then must point a pink ball to the direction he/she thinks the sound source is positioned and click with a button using the 3D mouse like shown in Figure 3.1. The client application records the real sound source position and the direction in which the

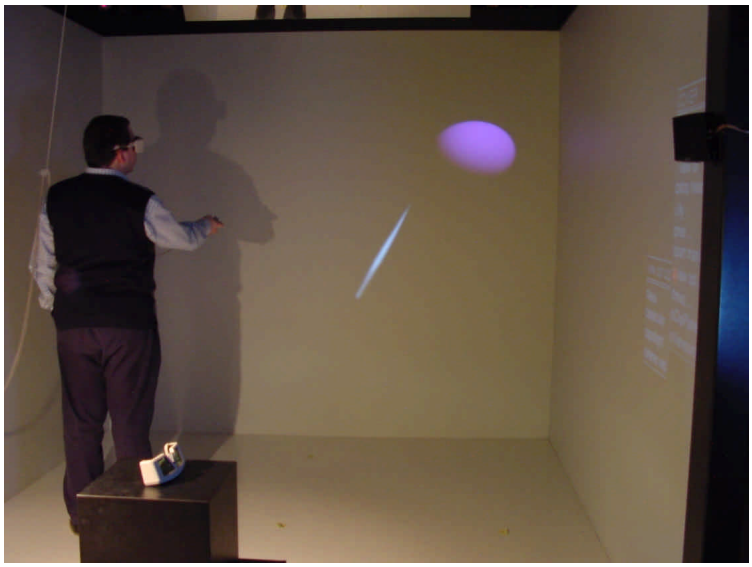

Figure 3.1 Test person at position 2 in the CAVE

test person clicked.

Specific positions have been marked with stickers on the floor of the CAVE where the test persons had to stand during the tests (see Figure 3.2):

- (1) Centre position, this is the optimal listening position for speaker based surround sound and should give the best results.
- (2) 1 step left of the centre position
- (3) Near to the left screen
- (4) 2 steps towards the left front speaker
- (5) Near to the front screen

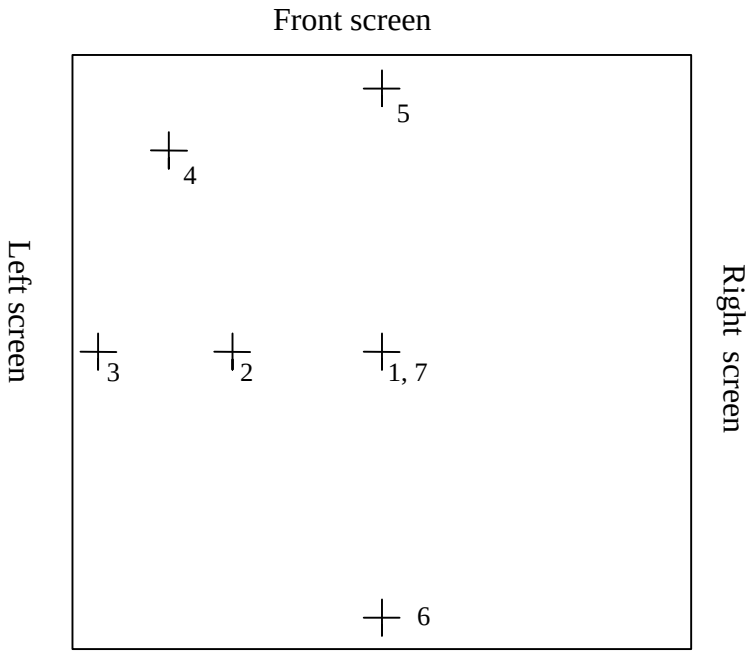

Figure 3.2 User positions in the CAVE

- (6) Near to the open back side

- (7) Centre position. This position again to check if the test person has adapted to the situation and if he detects the position of the sound sources better than at the beginning of the test.

For each test person the following data was recorded into files:

Position (1-7)	**Real position of the sound source**	**Perceived position of the sound source**	**Calculated angle deviation**

Table 3.1 shows the calculated mean angle deviation for each position over all users and the resulting mean angle deviation over all positions.

Angle	Listener Position							Overall Mean
	1	2	3	4	5	6	7	
0	13,21	29,26	22,33	21,44	26,05	14,59	10,85	19,68
15	18,42	16,67	14,31	30,21	20,26	14,42	12,80	18,15
30	12,75	6,01	5,89	9,25	14,60	13,57	6,48	9,79
45	13,28	10,41	7,24	11,81	9,80	20,04	18,21	12,97
60	28,69	27,79	29,96	14,97	9,65	19,94	31,24	23,18
75	56,74	57,51	51,74	39,46	16,45	18,60	31,29	38,83
90	40,00	22,66	43,02	21,51	13,92	5,69	24,69	24,50
105	19,31	35,81	42,11	29,22	23,68	13,15	16,25	25,65
120	15,03	25,42	57,37	13,93	14,71	9,93	14,53	21,56
135	25,16	9,79	14,49	13,07	13,76	18,60	19,00	16,27
150	24,87	5,57	10,95	9,65	7,78	10,43	23,77	13,29
165	36,62	40,06	9,99	14,25	7,74	27,06	30,31	23,72
180	36,86	17,49	16,01	10,38	4,39	5,51	3,39	13,43
195	33,70	11,87	14,27	9,54	6,20	14,26	19,96	15,69
210	29,54	20,26	32,75	10,87	6,28	26,40	27,57	21,95
225	12,83	11,74	16,64	15,90	13,68	23,37	19,26	16,20
240	14,26	22,53	30,73	42,80	25,91	12,88	17,05	23,74
255	16,10	26,45	48,37	31,25	33,27	23,92	19,87	28,46
270	27,79	76,54	65,16	60,06	14,79	10,04	16,86	38,75
285	38,12	48,36	43,42	68,44	24,51	24,70	34,33	40,27
300	31,38	40,76	40,82	24,25	19,35	37,71	35,14	32,77
315	14,50	16,56	18,49	9,70	7,11	16,13	11,99	13,50
330	9,81	16,48	20,88	23,66	21,57	20,24	10,35	17,57

Table 3.1 Mean angle deviation depending on the position

The most common point of all following diagrams is the lower deviation for sounds that come from the same direction as the loudspeakers

In Figure 3.3 the mean of the results of all test persons have been drawn in circular diagrams to show the dependence of the listening position.

It was most difficult to detect a sound direction coming from between the speakers.

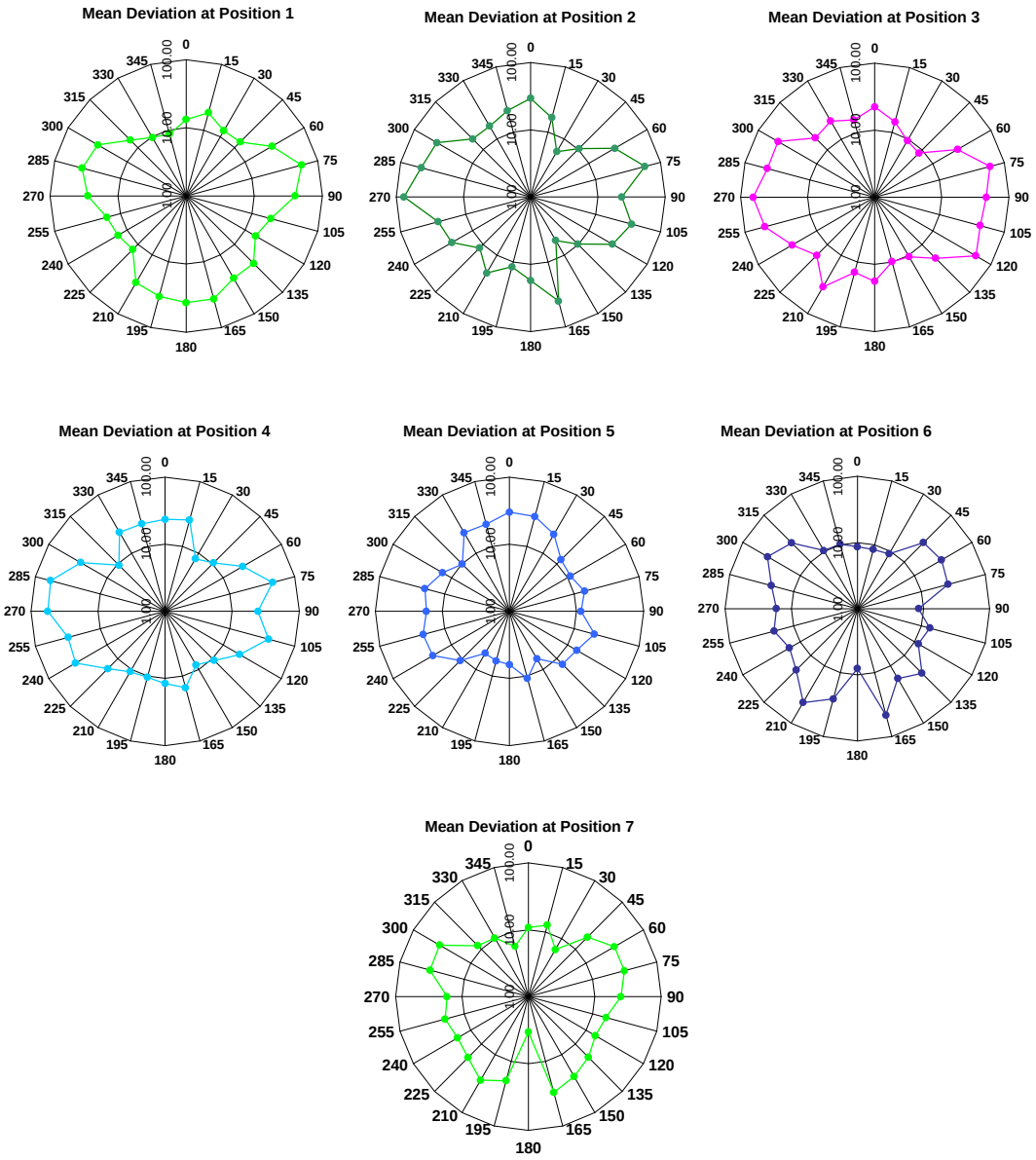

Figure 3.3 Circular diagrams of the mean angle deviation at each position

Because the rear speakers are oriented to the front, their sound was reflected by the front wall which made it difficult to hear the sound direction.

The diagram of position 7 is very similar to that of position 1 except the sound of a direction of 180° from behind has been detected very well likewise at position 6.

The diagrams for the left most positions 2 and 3 and the front left position 4 show it was difficult to detect sounds from both sides left and right due to reflections on the walls.

While at all other positions sound coming from behind was not detected very well, it was the contrary case at position 5 where the front directions were not detected successfully due to reflections on the walls.

The mean value of all positions are also presented in Figure 3.4. along with the average over all positions. The position of the loudspeakers can be seen very well at 45°, 135°,

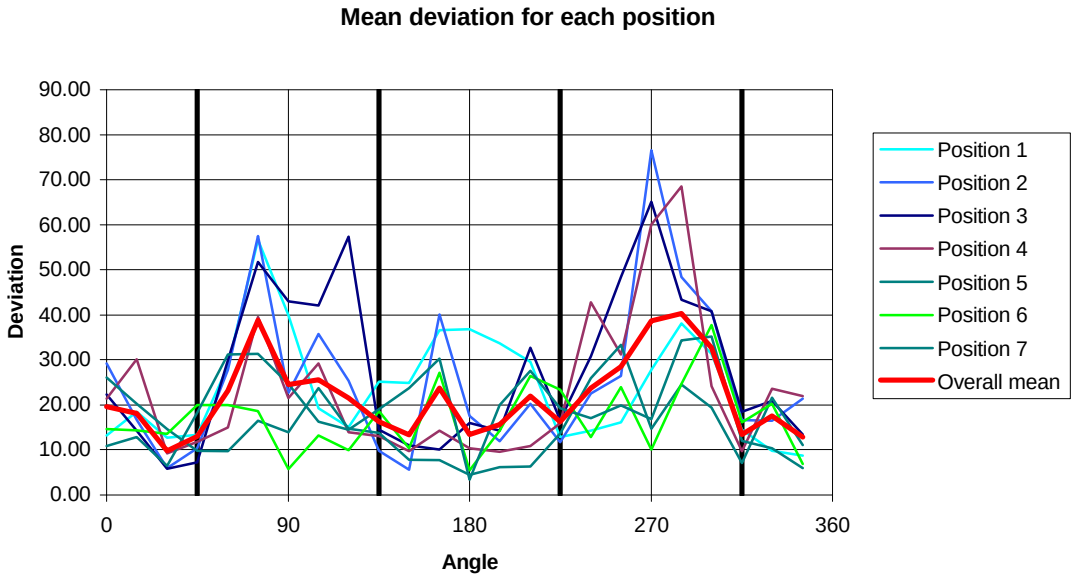

Figure 3.4 Mean angle deviation for each position and overall mean angle deviation

225° and 315°, here the variation of the deviation is lower than for the other angles.

Figure 3.5 represents a circular diagram of the mean angle deviation over all positions and all test persons. Here again sound directions that are similar to the speaker positions were more easily detected. The minimum mean deviation is not lower than 10° that is still a good result compared to the minimum angle interval of 15° during this test. Of course, if the angle step was larger the directions could be better detected. But in a real environment, sounds come from any direction.

The test persons remarked the direction of the sound seemed to change once they turned their head. Even with the looping sound they could not detect the direction of virtual sound source. Then they more often clicked in the direction of the loudest speaker.

While the circular diagrams express the deviation, a histogram shows the frequency of occurrence of the measured values within specific intervals. Peaks represent the angles

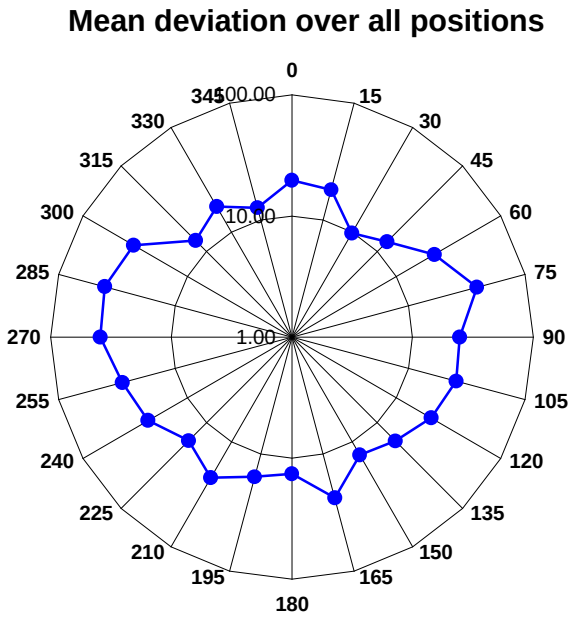

Figure 3.5 Overall mean angle deviation

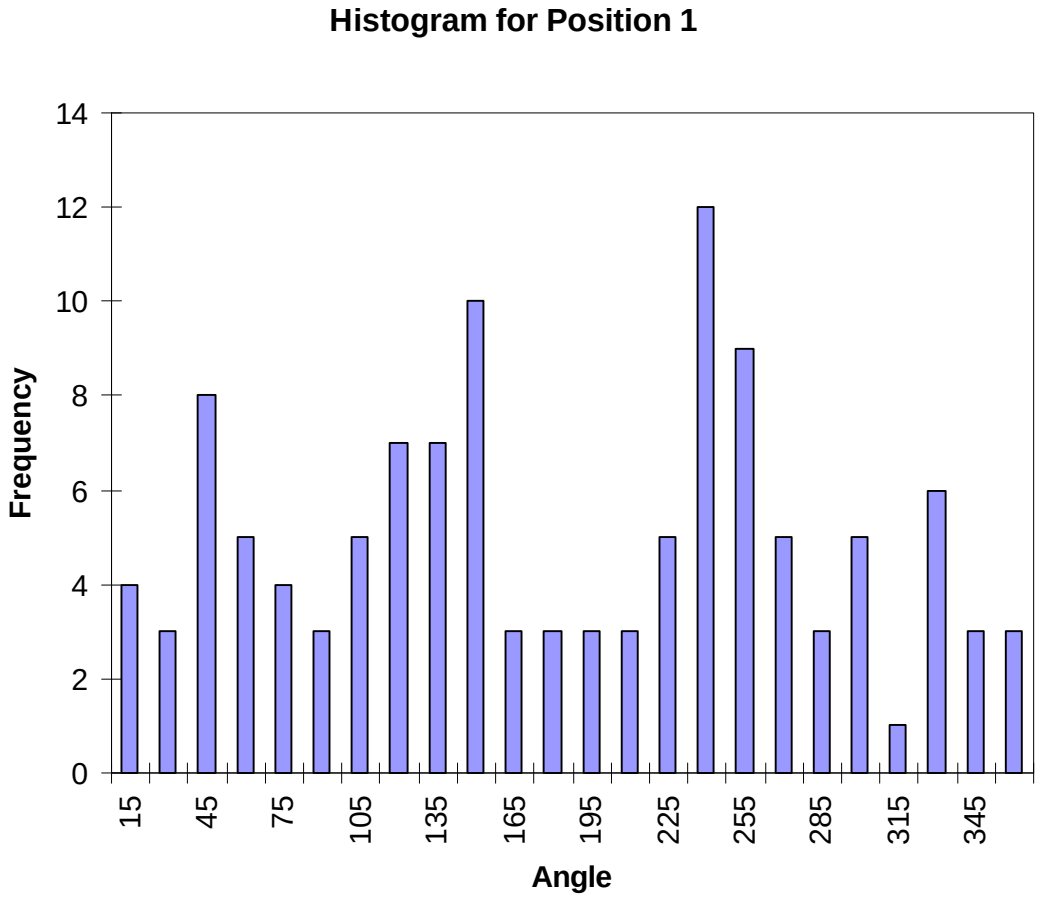

Figure 3.6 Angle histogram at Position 1

that were selected most often. In Figure 3.6 the peaks around 45, 135, 225 and 315 correspond to the location of the loudspeakers.

The current sound API can move the listener inside their own worlds but they can't generate sounds depending on the listener position relative to the loudspeakers.

For all these reasons the position of the listener relative to the loudspeakers should be taken into account for the calculation of the sound source positions *before* generating the sound.

4 Conclusions

This work gave a short overview of acoustics, surround sound and of the use of sound with Virtual Reality.

A concept for the Audio Server was conceived and client application interface was described. A prototype of the Audio Server and of different test applications were implemented. To make a commercial product, more things have to be done, but the present application is working successfully in different cases.

Using OpenAL for programming the Audio Server prototype was less complex than with DirectX, but this limits the functionality just to produce 3D sounds.

Measurements done with the Audio Server and a client application have shown it's possible to detect direction of sounds even with only 4 loudspeakers.

5 Future Work

Since the Audio Server is based on standard components, it is easy to replace them by more elaborate parts or to add more functions.

Possible improvements include:

- Include listener position relative to loudspeakers with sound generation.
- Audio Streaming for use in collaborative VR sessions and video conferencing.
- Support of different sound file formats. Currently only mono and stereo Windows wave files (WAV) are supported, but other file formats like MP3 could be added by using compression managers.
- Support for other surround sound formats and systems like Ambisonics.
- Graphical representation of the sound parameters in 3D space (e.g., position or orientation) in an appropriate display, for instance by using OpenGL.
- Configuration via a web interface. Currently the Audio Server has no other configuration option than the socket port. After all the Audio Server detects automatically which resources are available.

6 Management of Project

The project took longer than previously planned but was finished just in time. The main causes for the delays were:

- The definition of the single task contents was not clear enough at the beginning.
- Investigating the different already available solutions took too much time.
- The defined tasks were not treated closely enough according to the time schedule.

The closer the date of submission comes the more it was crucial not to include too much. Concentrating on the most important parts was very helpful.

The following pages show the project tasks and a corresponding Gantt chart of the time schedule.

6.1 Project tasks

The Table 6.1 lists the main tasks defined for this project.

Brunel University
Audio Server for Virtual Reality Applications
Marc Schreier

Table 6.1 Project tasks

6.2 Gantt chart

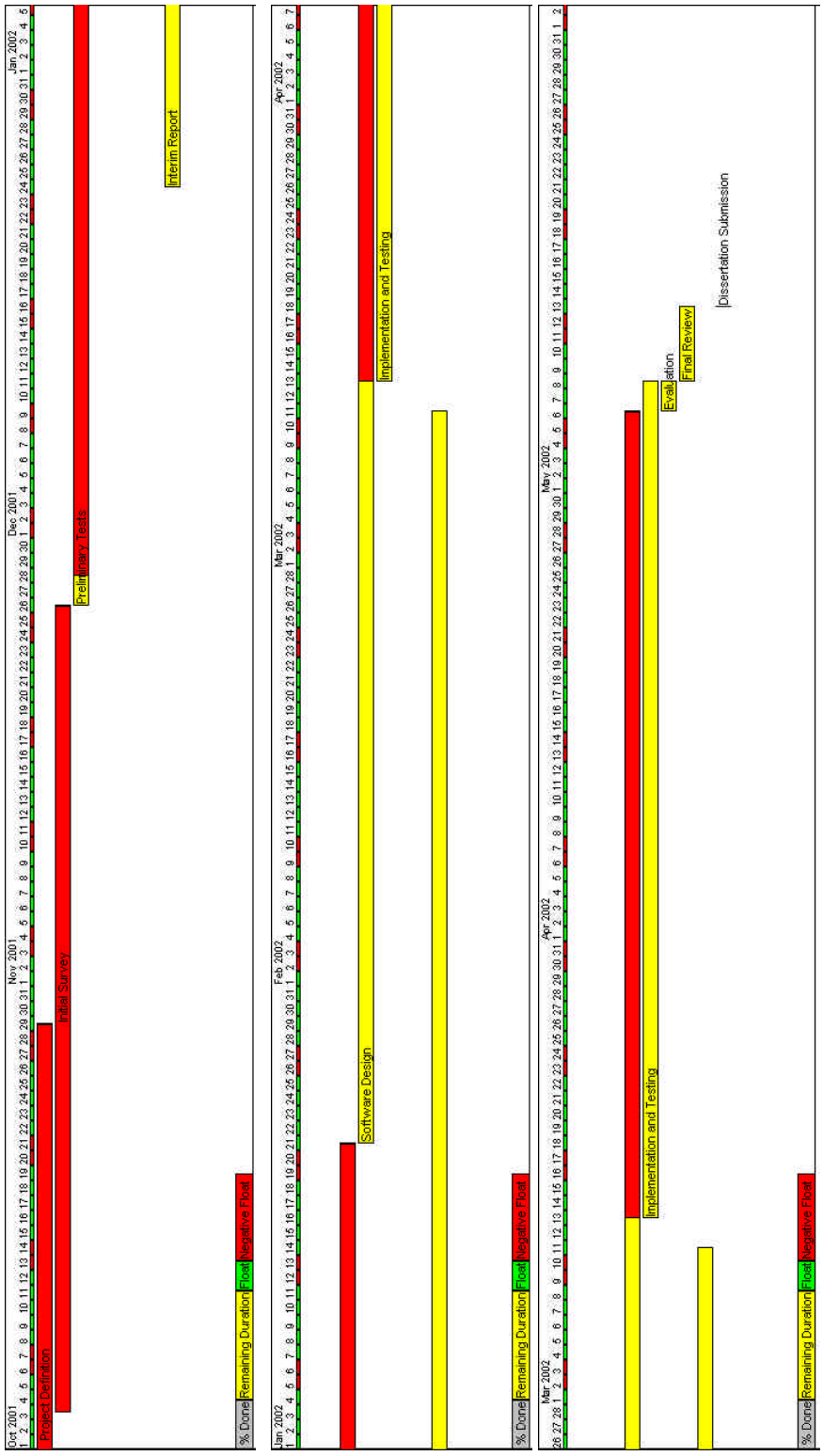

Figure 6.1 Gantt chart

7 Bibliography

Brix, Sporer and Plogsties, 2001, An European Approach to 3D-Audio, 100th AES Convention, Convention Paper 5314

C. P. Brown and R. O. Duda, An efficient HRTF model for 3-D sound, in *WASPAA '97* (1997 IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics, Mohonk Mountain House, New Paltz, NY, Oct. 1997).

Das S. and Goudeseune C. (1994), VSS Sound Server Manual, NSCA Audio Group, http://www.isl.uiuc.edu/software/vss_reference/vss3.0ref.html

Fahy F. (2001), Foundations of Engineering Acoustics, Academic Press, London, ISBN 0-12-247665-4

Gardner W. (1992), The Virtual Acoustic Room, Computer Science and Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts

Gerzon M.A. (1985), Ambisonics in Multichannel Broadcasting and Video, J. Audio Eng. Soc., vol. 33 no. 11, pp. 859-871 (1985 Nov.)

Hämälä T. (2002), OpenAL++ - An object oriented toolkit for real-time spatial sound, University of Umeå, Sweden

Hoag Sr. K.J. (1998), Facilitating rich acoustical environments in virtual worlds, NAVAL POSTGRADUATE SCHOOL, Monterey, California

Larsson P., Västfjäll D. and Kleiner M. (2001), Do we really live in a silent world? The mis(use) of audio in virtual environments, AVR II and CONVR 2001, Conference at Chalmers, Gothenburg, Sweden

Levergood T.M., Payne A.C., Gettys J., Winfield Treese G. and Stewart L.C. (1993), AudioFile - A Network-Transparent System for Distributed Audio Applications, Digital Equipment Corporation, Cambridge Research Lab, USENIX Summer Conference

Microsoft Developer Network (MSDN), Platform SDK: Networking and Distributed Services, Graphics and Multimedia Services, http://msdn.microsoft.com/

Microsoft DirectX, http://www.microsoft.com/directx/

OpenAL, Open Audio Library, http://www.openal.org/ (Loki Entertainment has stopped development, the software was made available also by Creative Labs as Open Source at http://opensource.creative.com)

Perraux J.-M., Boussard P. and Lot J.-M. (1998), Virtual Sound Source Positioning and Mixing in 5.1, First COST-G6 Workshop on Digital Audio Effects (DAFX98), November 19-21, Barcelona, Spain

Savioja L. (1999), Modeling Techniques for Virtual Acoustics, Publications in Telecommunications Software and Multimedia, Helsinki University of Technology, Finland

Sirotin V., Debeloff V. and Urri Y. (1999), DirectX Programming with Visual C++, Addison-Wesley-Longman, Munich, ISBN 3-8273-1389-9

Tomlinson H. (2000), 5.1 Surround Sound – Up and Running, Focal Press, Boston, ISBN 0-240-80383-3

VRML, Virtual Reality Modeling Language, http://www.web3d.org

VSS, Virtual Sound Server, http://cage.ncsa.uiuc.edu/adg/VSS/, NCSA Audio Group

Zwicker E. and Fastl H. (1999), Psychoacoustics – Facts and Models, 2nd Edition, Springer-Verlag, Berlin, ISBN 3-540-65063-6

8 Appendix A: Table of Abbreviations

AC-3	Audio Compression Standard 3 (Dolby Digital)
API	Application Programming Interface
DSP	Digital Signal Processor
EAX	Environmental Audio Effects
GUI	Graphical User Interface
HRTF	Head Related Transfer Functions
ITU	International Telecommunications Union
MIDI	Musical Instruments Digital Interface
MMI	Man Machine Interface
OpenAL	Open Audio Library
OpenGL	Open Graphics Library
PC	Personal Computer
SDK	Software Development Kit
VR	Virtual Reality
VRML	Virtual Reality Modeling Language

9 Appendix B: List of used hardware and software

- Hardware:

  Mainboard: Asus TUSL2-C, 512 MB RAM

  CPU: Intel Pentium III, 1.133 GHz

  Sound Card: Creative Labs Sound Blaster Audigy Platinum EX

- Software:

  Creative Labs EAX SDK

  Microsoft DirectX 8.1 SDK

  Microsoft Office 97 and 2000

  Microsoft Windows 2000

  Microsoft Visual Studio 6.0

  OpenAL SDK

  PlanBee (for Management of the project)

  Visio Technical 4.0

10 Appendix C: Prototype implementations

10.1 Server implementation example

A prototype implementation was programmed to demonstrate the basic functions described in chapter 2. The graphical user interface shown in Figure 10.1 has a log window for messages and errors, a two-dimensional display of sound sources near the listener position and a status window for all active 3D sound sources.

The current implementation has the following features:

- Sound file management using default and cache sound directories.

  The directories are created as subdirectories of the current application path.

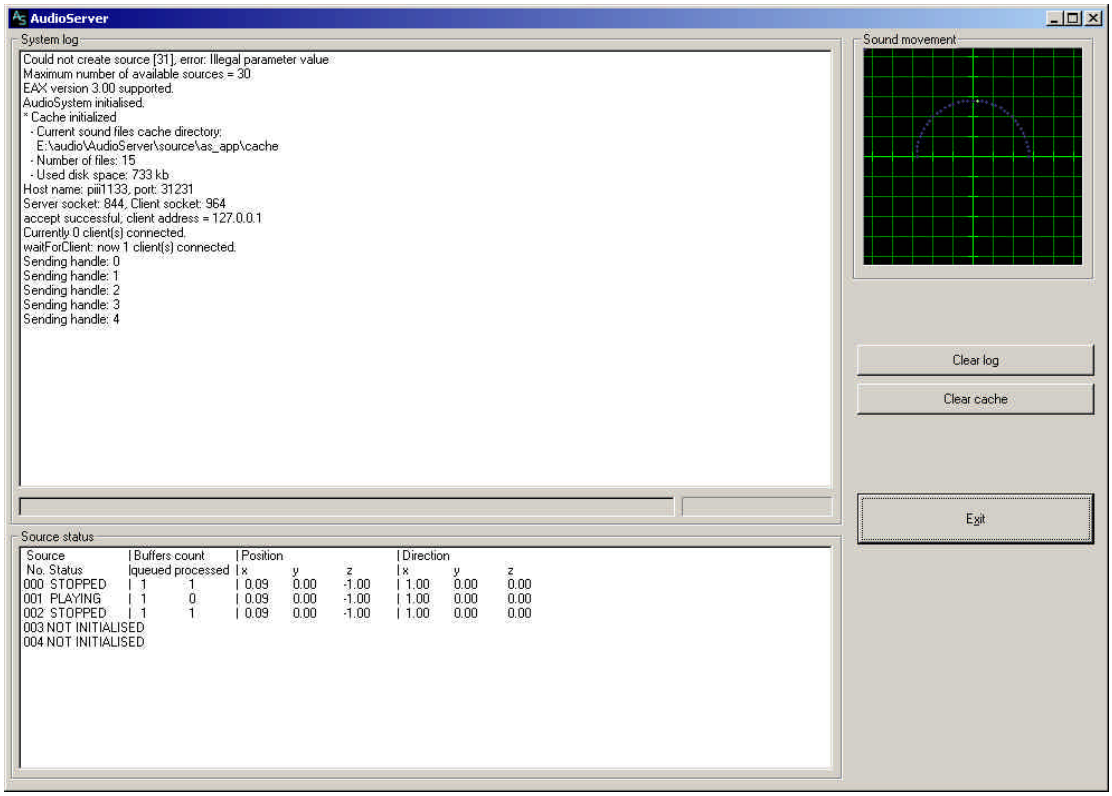

Figure 10.1 Audio Server prototype

- Dynamic allocation of 3D sound sources and sound buffers.

  At any time, sound files can be loaded into sound buffers and assigned to 3D sound sources.

- Exclusive client connection.

  Only one client can connect at a time. When the client disconnects, all previously allocated resources are de-allocated. The Audio Server then waits for a new connection.

- File uploading.

  A sound file can be uploaded to the server by the client. The file is stored inside the sound cache directory.

- Logging of commands and errors.

  Important activity information and errors are shown in a list box.

- Status display of sound sources.

  Another list box shows properties of currently allocated sound sources, e.g., playing status, buffer control, position and orientation.

- Graphical display of sound source position.

  Sound sources located near the listener position at <0,0,0> are displayed in a two-dimensional grid. Moving sources leave a trail of their last known position.

The source code of the Audio Server implementation would be too large to fit into this document.

10.2 Client implementations

The test clients only require a socket connection to send the AudioServer commands. To make the client application more easily readable the AudioServer commands can be encapsulated into the classes used for communication.

The first implementation was written to test all the functions implemented with the AudioServer step by step.

Another application was written that plays the sound of a Star Wars light-sabre moving one time around the listener. Near the left and the right position two different sounds are played to simulate different vibration effects.

An implementation was used with the COVISE/COVER environment to do the evaluation, where a sound was generated at random position and the test person had to detect the direction with the 3D mouse.

The last implementation was realised again inside the VRML browser of the COVISE/COVER software.

Simple sample code for a client (the encapsulated Audio Server commands are highlighted):

```
int main(int argc, char** argv)
{
    ASSound* test; // class that encapsulates communication
and AudioServer commands

    if (argc!=4)
    {
        cerr << "Syntax: astest <server> <port>
<soundfile.wav>" << endl;
        return 0;
    }
```

```
    // create one sound for the tests
    test = new ASSound(argv[1], atol(argv[2]), argv[3]);
    if (!test->isValid())
    {
        cerr << "could not initialize sound" << endl;
        return 0;
    }
    if (!test->connect())
    {
        cerr << "could not connect with Audio Server" <<
endl;
    cerr << "--- TEST_LOOP ---" << endl;
    cerr << "Setting looping on... ";
    test->setLoop(true);
    cerr << "Playing sound... ";
    test->start();

    cerr << "Waiting 5s...";
    Sleep(5000);

    cerr << "Stopping sound... ";
    test->stop();

    cerr << "Done." << endl;
    if (test) test->quit();

    delete test;

    return 0;
}
```

Example for the command Loop in a class member function:

```
/** Determine if the file should be played continuously.
  @param loopOn true if file should be looped
*/
void ASSound::setLoop(bool loopOn)
{
    char msg[MAX_BUFLEN];
    sprintf(msg, "SET_SOUND %d LOOP %d", handle, (loopOn) ?
1 : 0);
    socket->write_string(msg);
}
```