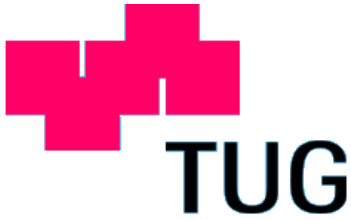


Graz University of Technology
Erzherzog-Johann-University

# Three-Dimensional Audio Interfaces for the Blind

Christopher Frauenberger

Diploma Thesis

Graz, January 2003

Supervisor: Prof. Gernot Kubin
Signal Processing and Speech Communication Laboratory (SPSC Lab)

**Three-Dimensional Audio Interfaces for the Blind**

Christopher Frauenberger
Kontakt : c.frauenberger@gmx.net - http://home.pages.at/frauenberger

Betreut durch Prof. Gernot Kubin
Signal Processing and Speech Communication Laboratory (SPSC Lab)
Kontakt : g.kubin@ieee.org - http://spsc.inw.tugraz.at

Diplomarbeit
Institut für Nachrichtentechnik und Wellenausbreitung, Technische Universität Graz
11. Jänner 2003, 62 Seiten

## Zusammenfassung

Die vorliegende Diplomarbeit befasst sich mit der Entwicklung neuer Ausgabeverfahren von Computern zur Verbesserung des Zugangs von sehbehinderten und blinden Benutzern zu moderner Informationstechnologie. Ein Überblick über bestehende Technologien bestätigt, dass sehbehinderte Menschen starke Nachteile im Umgang mit Computern haben. Das ist vor allem auf die sequentielle Art bisheriger Ausgabeformen zurückzuführen.

Der Ansatz, der in dieser Arbeit verfolgt wird, stützt sich auf die sehr ausgeprägte menschliche Fähigkeit des räumlichen Hörens. Ziel ist es, den Informationsgehalt der Ausgabe gegenüber den bisher üblichen Screenreadern oder Braillezeilen zu erhöhen. Ein dreidimensionaler akustischer Bildschirm kann mehrere Informationsquellen beinhalten und der Benutzer kann sie gegeneinander abgrenzen, wenn sie verschieden positioniert werden können. So kann der Informationsfluss und damit die Effizienz der Bedienung für Sehbehinderte deutlich verbessert werden.

Mit dieser Diplomarbeit wird ein System zur Erzeugung virtueller akustischer Realität (VAR) präsentiert. Es erlaubt Benutzern, sich in einer virtuellen Umgebung alleine durch das Gehöhrte zu orientieren und sich in ihr mittels eines Joysticks zu bewegen. Eine binaurale Audiowiedergabe implementiert Algorithmen zum Richtungshören und zur Raumakustik, um eine authentische Simulation zu erreichen. Über einen Headtracker werden auch Kopfbewegungen berücksichtigt. Eine Programmierschnittstelle (API) bietet Funktionen zur Steuerung des Inhalts und erleichtert die Erstellung von Benutzerprogrammen. Die Signalverarbeitung selbst wird von einem digitalen Signalprozessor übernommen. Ein Beispiel zeigt die Entwicklung eines Desktopersatzes für sehbehinderte Benutzer.

Neben den technischen Details zur Realisierung des Systems wird auch auf die speziellen Anforderungen für sehbehinderte und blinde Menschen eingegangen, die für die Entwicklung von Benutzerschnittstellen berücksichtigt werden müssen. Ausserdem werden Attribute zur Evaluation und ein Usabilitytest vorgeschlagen.

Klassifikation (Computing Reviews 1998): H.1.2 (User/Machine Systems), H.5.2 (User Interfaces), H.5.5 (Sound and Music Computing)

Schlagworte: Akustischer Bildschirm, Sehbehinderung, räumliches Hören, Ambisonic

**Three-Dimensional Audio Interfaces for the Blind**

Christopher Frauenberger
Contact : c.frauenberger@gmx.net - http://home.pages.at/frauenberger

Supervised by Prof. Gernot Kubin
Signal Processing and Speech Communication Laboratory (SPSC Lab)
Contact : g.kubin@ieee.org - http://spsc.inw.tugraz.at

Diploma thesis
Institute of Communications and Wave Propagation, Graz University of Technology
11th January 2003, 62 pages

## Abstract

This thesis is dealing with alternative interaction modes for visually impaired and blind people to use computers. An overview over currently used techniques shows that these people are at a significant disadvantage when accessing computers in comparison to people without disabilities. This mainly results from the sequential manner of customary assistive technology.

The aim of the proposed approach is to exploit the human hearing capabilities to a better degree than this is done by screen-readers. A surrounding, three-dimensional audio interface is potentially increasing the information flow between a computer and the user. The human auditory system is able to segregate different sound sources, if they can be distributed in a surrounding environment. This allows multiple sound streams to be presented at once.

This thesis presents a virtual audio reality (VAR) system which allows computer users to explore a virtual environment only by their sense of hearing. The used binaural audio rendering implements directional hearing and room acoustics via headphones to provide an authentic simulation of a real room. Users can freely move around using a joystick. A headtracker improves the simulation by providing the possibility of head movements as additional orientation aid. The proposed application programming interface (API) is intended to ease the development of user applications for this VAR system. It provides an easy to use C++ interface to the audio rendering layer. The signal processing is performed by a digital signal processor (DSP). A use case shows how a desktop replacement for visually impaired users can be implemented using this API.

Besides the details of the technical realisation, this thesis also investigates the user requirements for the target group. Their special abilities and disabilities are stated which must be considered when designing interfaces for this group of users. Finally, evaluation aspects are discussed and a usability test is proposed.

Classification(Computing Reviews 1998): H.1.2 (User/Machine Systems), H.5.2 (User Interfaces), H.5.5 (Sound and Music Computing)

Keywords: audio interfaces, visual impairment, spatial sound, Ambisonic, ray tracing

# Acknowledgements :

First of all, I would like to dedicate this thesis to my parents for their support throughout my studies. They made these lines possible and encouraged me at any point of my studies while always respecting the way I did them.

I would like to thank Prof. Gernot Kubin and DI Christian Feldbauer for their supervision and that they gave me the chance to write my thesis on this topic.

Many thanks to the ISIS group and the Berufsförderungsinstitut, especially Walerich Berger and Dietmar Ogris. They provided me with essential information about the needs of visually impaired people and gave me the possibility to promote the idea and vision of this project on various occasions. I would like to express my deep respect for their efforts on the field of integration of disabled people into our society. The Bundessozialamt Steiermark was kindly funding the project. Texas Instruments supported us by providing many hardware components.

Christopher Frauenberger Graz, 11th January 2003

# Contents

# List of Tables

# List of Figures

# Listings

# Chapter 1

# Introduction

The intention of this diploma thesis is to investigate the possibilities of using Virtual Audio Reality (VAR) as an interface to computers for visually impaired and blind users. Because of the sequential nature of commonly used technologies, it was found to be essential to develop a new interaction mode to increase the information flow between user and computer. Visually impaired and blind people rely on tactile and acoustical interaction modes. While tactile devices like braille lines can only provide a limited amount of information per time, audio has the capability to provide a lot more information at once if made surrounding and spatial. Despite that this project is very challenging from the technical point of view it also deals with the social problem of the accessibility of modern information technology for people with special needs.

The project was launched in January, 2002 and was funded by the Federal Social Welfare Office Styria. The ISIS (Information, Service, Integration and Schooling) project formed the organisational framework and was a very important partner. The information provided by the ISIS team concerning the target group, the help with official representatives of the people concerned and the public relation which presented the project to the non-universitary world were essential for the success of the work. The project report of the preceding study project provides more information about the projects and institutions involved [2].

The first chapter of this thesis is intended to introduce the topic in general, gives an overview of the current state of the art of assistive technology for blind computer users and provides the initial approach which was chosen to develop a prototype system. In chapter two, the audio interface is specified. The user and system requirements are determined and design issues are discussed to provide interface designers with strategies of how they can find a proper acoustical representation of information in audio interfaces. Chapter three discusses the realisation of virtual audio reality, provides an overview of what is essential for spatial hearing, shows the algorithms needed for spatial perception and room acoustics and provides information of how the system was implemented on a

Digital Signal Processor (DSP). A high level Application Programming Interface (API) was developed to provide a simple to use interface to the VAR which is described in detail in chapter four. This chapter includes also an example of how a virtual scene can be created and used by a user application. Finally, chapter five is concluding the work, describes possible evaluation methods in terms of objective quality assessment and usability tests and provides an outlook on future work.

## 1.1 Starting point

Development can be conducted with more effort if a bigger market is available. Developing for minor groups is therefore seldom economical, but the amount of visually impaired people is often underestimated. According to the World Health Organisation there are 40–45 million blind and 135 million visually impaired people in the world [3]. Together this equals the inhabitants of Germany, Spain and France.

The percentage of computer users with disabilities is also surprisingly high. Figure 1.1 shows the relations between Internet users affected by disabilities [4]. With $3,5\%$, visually

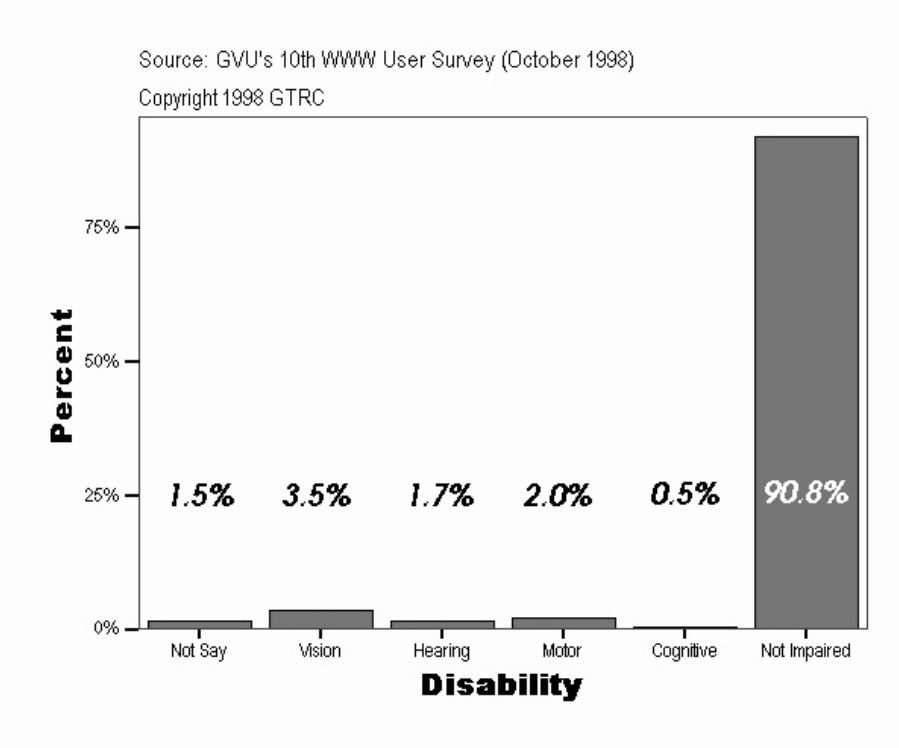


Figure 1.1: Internet access and disabilities

impaired people are the largest group among computer users with disabilities.

The user-characteristics are different from the average user not just for the lack of visual

perception. An important fact is the age distribution among users, shown in Figure 1.2 [4]. An interface designer must consider serving comparatively more young and old

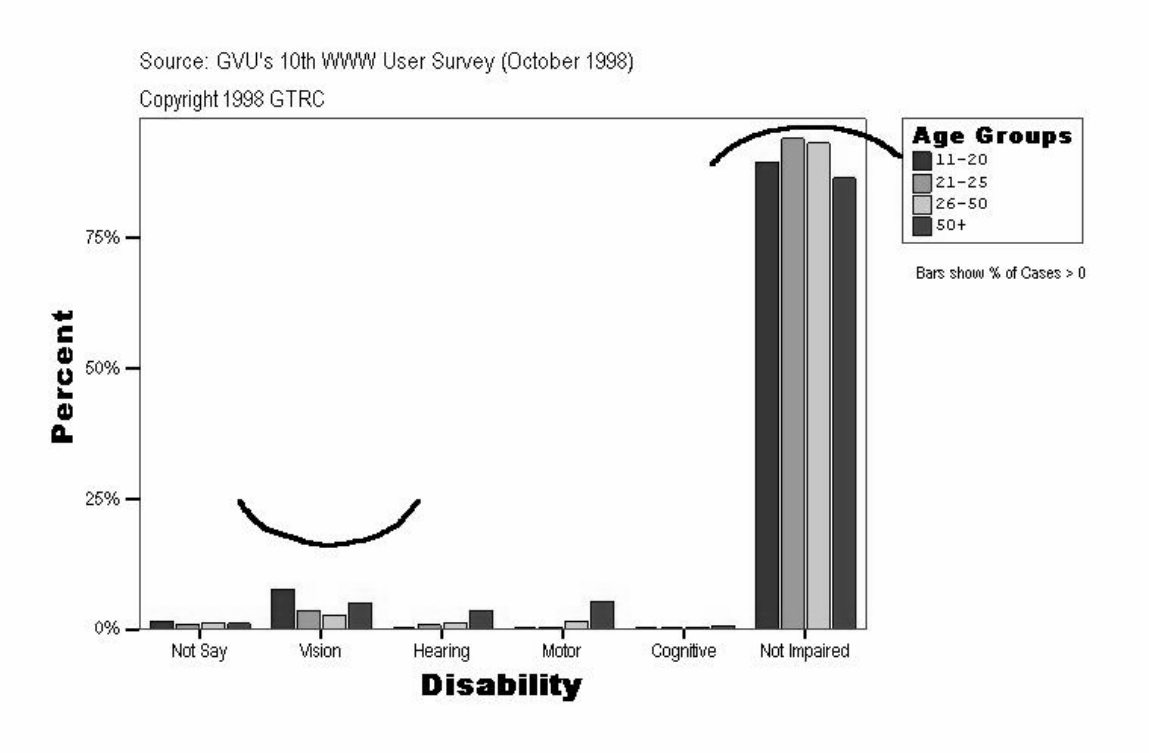


Figure 1.2: Age groups of Internet users

people[1]. The visually impaired start using the Internet at a younger age and are still more interested in the Internet when they are over 50 years old.

Ninety percent of the visually impaired are not totally blind which would allow to use simple visual cues in most cases, but it would also lead to a smaller excluded minority - the completely blind. The same problem occurs if interfaces are exclusively based on braille because $20\%$ of the users cannot read braille.

Obviously it is not possible to provide generic interfaces for all disabled people considering all possible combinations of disabilities and people's pre-knowledge, but an auditory approach supported by other assistive technologies is able to reach more than most of the alternatives.

The most common assistive technologies for the visually impaired are text-to-speech screen readers in combination with refreshable braille lines. Systems like JAWS[2] [5] were developed to make the Windows[3] operating system accessible. With its internal software

[1]The chart shows relations between the disability groups; the absolute count in the vision group was: 23 (11–20), 23 (21–25), 86 (26–50), 43 (50+)

[2]JAWS$^{TM}$for Windows is a registered trademark of Freedom Scientific

[3]Windows$^{TM}$is a registered trademark of Microsoft Corp.

speech synthesiser and the computer's sound card information from the screen is read aloud, providing technology to access a wide variety of Windows applications. It also outputs on refreshable braille displays to make the information accessible through the tactile mode. A major application nowadays is browsing the Internet. JAWS is directly processing HTML (HyperText Markup Language) and provides a structural interpretation of a site, but the variety of different techniques like Flash[4], Java[5] or Java Script makes it very difficult to keep track of the evolution of the World Wide Web. Other currently used systems include enlarging hardware and software, optical character readers and simple audio note takers; an investigation of the use of these technologies in programs for students is given in [6].

The most important drawback of all currently used systems is the sequential nature. Information can be structured to make the navigation as easy as possible, but it is not possible to make more information available at once.

## 1.2 Approach

To be able to increase the information flow between a visually impaired user and a computer the sequential techniques need to be extended. Human auditory perception is capable of more than just following a single voice or sound if it is possible to introduce spatial simulation techniques. If humans can distinguish the position from where sound is heard, they also can segregate its content, and are therefore able to perceive more information at once. Hearing is also a sense which allows different levels of intenseness of perception, where the range reaches from background sound to speech. This enables us to adjust the priority of information to the desired attention of the user.

The initial idea of this project is to create a virtual auditory reality (VAR) to provide information on different places within a (as much as possible) natural environment. Visually impaired people should be able to explore computers as they do in their everyday life when entering a room they do not know. VAR enables visually impaired people to get a first overview of an environment without the necessity to explore it in detail.

Technically, the first challenge to solve is the positioning of virtual sound sources in virtual rooms. The creation of the imagination of directional hearing can be done with various methods like wave synthesis or Ambisonic. If desired, to be produced via headphones, head related transfer functions (HRTFs) are needed to model the influence of ears, the head and shoulders onto sound waves.These techniques do not provide distance information which is essential to be able to find a sound source position. Distance cues can be encoded using the loudness, respectively the change of loudness in motion, but also with the reflections of the sound wave on the enclosing walls which reach the

[4] Flash™ is a registered trademark of Macromedia, Inc.

[5] Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc.

ears temporally delayed. Finally, the nature of the environment needs to be modelled. Parameters like room size, reverberation or wall properties must be considered to create a realistic perception.

This thesis is presenting an approach to create VAR using an Ambisonic algorithm which was adapted for the use of headphones. Reflections are computed and the loudness shift is integrated to implement distant cues. A reverberation algorithm enhances the natural perception of the created scene. These algorithms were implemented on a digital signal processor. An application programming interface (API) is provided to have easy access to the 3D modelling.

# Chapter 2

# Audio Interface

In comparison to visual displays, audio interfaces have a number of advantages. The displays can surround the user and offer a larger display area. Commonly used computer screens are offering a small cutout of a two-dimensional surface at a very high resolution. Audio is capable of a surrounding, three-dimensional displaying area but at significant lower resolution. The localisation blur depends on the angle of incidence. In the horizontal plane the direction of the lowest blur is the front with up to $\pm 3,6°$, the one with the highest blur to the left respectively to the right with up to $\pm 10°$ [7]. Even so, audio interfaces might contain a lot of information, if one can take advantage of the larger display area and the special abilities of the human auditory system.

The extensive use of auditory displays for technical purposes started with data exploration tools. The need of providing much information at once in scientific applications often made visual-only interfaces confusing and too complex. Audio as additional mode extended the possibilities [8]. For instance seismic signals consisting of very big sample sets were hard to visualise, but easy and fast to explore if linked to a sound's pitch. Audio interfaces for the visually impaired are mainly based on text-to-speech technology, but first attempts to provide an audio interface for typical visual applications were presented in [9] where a simple screen with various windows was displayed using different sounds as window qualifiers. Evaluation tests showed that in cooperation with text-to-speech technology, the system was usable for visual impaired users and the subjects were able to perform complex tasks. However, there were a number of problems identified. Most of these were related to the high extra load imposed on the user's memory to recognise the qualifiers. Right now no commercial product is available which follows up the approach to map visual environments to audio interfaces. Most effort is conducted to extend the text-to-speech systems to be compatible with different programmes.

## 2.1 Mapping

Introducing spatial sound to audio interfaces changes the strategy of converting visual information to audio interfaces. The most obvious advantage over sequential techniques is the natural perception, humans are using their directional hearing for orientation at all times. Besides the representation of information, in the range of generic sound to speech, information is also provided with a position property. Audio interfaces can be modelled in the space. This allows multiple information sources and grouping of information.

In the style of screen icons, "Earcons" can be defined which represent the properties of the information, including the position. The container in which all these earcons are embedded is modelled as a room in the virtual audio space and the user himself resembles the pointer device which is capable of movements and actions. This allows an auditory representation of a so called window, icon, menu and pointer (*wimp* [9]) style of interface on which most of the commonly used operating systems are based.

This visual to auditory mapping can be illustrated by the example of a Windows desktop containing the usual icons, the task bar and an opened Explorer application. The user

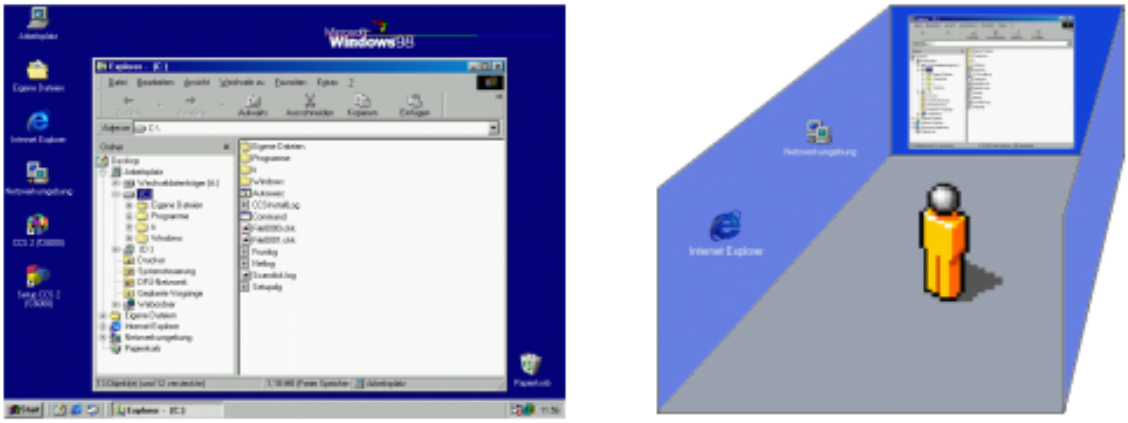

Figure 2.1: Mapping of visual information to an auditory display

as pointer device might virtually "walk" to a desired earcon and perform any action (e.g. execute, rename, view properties) on it.

The desktop application is only one of various possible applications where this mapping can be performed. Every menu, every list or even drawing applications can be mapped by following the same strategy:

1. Identifying the desired information content.
2. Assigning earcons to the information and defining the auditory representation.
3. Grouping earcons to parent earcons if useful.

4. Defining the desired actions which need to be performed on the earcons.
5. Defining a proper space in the virtual environment as the enclosing virtual room of the interface.
6. Placing the earcons in the VAR.

Spatial audio interfaces provide the possibility to cover most of the needed applications. However, the quality of the result depends on how well the mapping was performed. To improve the mapping, it is necessary to know as much as possible about the target group, what audio representation might achieve the desired goal and which technical possibilities are available. These problems are investigated in the next sections.

## 2.2 User Requirement Specification

The user requirement specification has two key issues: 1) The target group has certain abilities and disabilities which differ from a common computer user and need to be considered when designing audio interfaces. 2) The target group uses the interface under certain circumstances which might restrict the utilisation of special techniques or special devices. E.g. the use of loudspeaker arrays as an interface in an Internet cafe might not be suitable.

### 2.2.1 Target Group

The target group is visually impaired or totally blind. This restricts interaction modes to audio and tactile devices. Many studies proved that they have better other senses to compensate for their lack of vision. Auditory compensation might consist of a re-organisation and reallocation at the level of the cortex (structural hypothesis) so that auditory and tactile areas function better or are the result of a better development due to the vision impairment (strategic hypothesis) [10]. However, the consideration of the differences in perception and imagination is inevitable for interface designers [11].

Most of the visually impaired were educated to navigate according to their listening skills in their everyday life. Mobility training is providing them with a highly sensitive warning, alerting and scanning system which allows visually impaired people to avoid the risks they are exposed while doing any physical activity. In [12] a mobility skills training system based on a computer generated three-dimensional audio environment was introduced which was intended to reduce the risk of accidents for the unexperienced visually impaired. In general, the target group has a very distinct ability of orientation on the base of hearing. First experiences with the prototype showed that visually impaired users were more confident in moving around in a VAR than not visually impaired people.

Individuals who are congenitally blind rely on their remaining senses ever since they were born and therefore have the most distinct hearing and sense of touch among the visually impaired on average. On the other hand, congenitally blind have limited imagination of space and rely very much on their memory when moving around in a room. People who went blind later can build up a better imagination of reality. Although the memory skills of the visually impaired are generally above the average, these facts must be considered when modelling virtual rooms. Too complex rooms and combinations of rooms impose extra load on the user's memory and may confuse users.

### 2.2.2 Environment

To be able to determine all restrictions which may apply on the development of audio interfaces for this user group, one must investigate the environment of where the system is intended to be at service and what kind of application it might have.

The use of the system at a stationary working place would allow loudspeakers as the acoustical interface device. The realisation of directional hearing with loudspeaker arrays is possible and often of better quality than with headphones. The major drawback is the impact on the environment. It is not possible to use such arrays in public places or offices. Vice versa the impact of the environment on the system also influences the simulation quality. Properties of virtual rooms can not be simulated accurately without knowing the real room in which the interface will be used. Undesired reflections and artefacts might falsify the result and lead to disorientation. However, if mobile computing is desired, headphones are the only way of acoustical interface device that is possible for VAR.

Feedback devices like keyboard and mouse also need to be adapted for the target group. While a keyboard is suitable because it is basically tactile, a mouse is hardly usable for visually impaired user. The lack of a clear boundary and the impossibility to check the movements performed simultaneously on screen is confusing. In the system presented by this thesis, the user himself is the pointer device. Intuitive movements can be achieved by using a joystick, but experiments with the prototype showed that especially congenitally blind people do have problems with a joystick. They had no clear imagination of the implementation of movements when using the joystick. A touch board device might satisfy the requirements better.

## 2.3 System Requirement Specification

The technical challenge of creating virtual audio reality demands special input-output devices and significant computational power. To be able to provide a system to the people concerned which is accessible not only in terms of usability but also in terms of costs, it was important to use only components which are customary and widely

used. This section describes the chosen components and their properties with which the prototype was built. It also discusses differences between this system and other assistive technology.

### 2.3.1 Components

To be able to compute the algorithms which are presented in chapter 3 more computational power is needed than can be obtained from a normal PC (see the algorithm sections for an estimation of resource consumption). This is also because the audio interface is intended to be on the level of the operating system and not a user application itself; high computational load due to the interface would restrict the applications which the system is intended to present. These facts caused the employment of a digital signal processor (DSP) which is controlled by the host PC but is doing the most costly calculations. A prerequisite for the connection between DSP and the host computer is given by the amount of data to be transferred. The host is controlling the virtual scene and is therefore also responsible for its content. The bandwidth of the connection must allow to transfer audio data in real-time into a DSP memory buffer. With a PCI (Peripheral component interconnect) interface, available in almost all modern computers, this prerequisite is fulfilled. PCI can yield 132MBps (million bytes per second) in the minimal implementation which would allow to transfer 1500 audio channels (44kHz, 16bit) in theory. Practically a throughput of 50–80MBps can be achieved which equals up to 450 audio channels and is by far sufficient even if a lot of control traffic needs to be transferred as well.

The influence of the quality of headphones is not fully determined yet. It was found that the impulse response of headphones has more influence on the localisation process than the frequency response. This can be explained with the importance of the inter aural time delay (ITD) on the localisation of sound sources. The law of the first wave-front ([7]) states that if sound waves reach the listeners ears with more than $630\mu s - 1ms$ time delay, only the first incoming wave determines the direction from which the sound is heard. This is caused by a human head's dimension which, with a maximum way difference of approx. 20cm between the ears, produces ITDs smaller than $630\mu s - 1ms$ if a single sound source is heard. For headphones this means that the onset time and therefore the impulse response is the most important quality criterion in terms of localisation of sound sources. Electrostatic headphones are used with the prototype system which are known for their impulse fidelity. However, the minimal requirements of the headphones quality were not investigated in this thesis.

The localisation of sound sources can be improved significantly, if head movements are considered [13, 14]. Head movements provide the user with two additional cues of determining the direction: 1) Subliminal head movements solve ambiguous localisation problems, most often with decisions of whether the source is in front or behind. 2)

| Type | Specifications |
|---|---|
| Texas Instruments, TMDS3260D6701 Evaluation Module | DSP 6701, up to 133MHz, 1GFLOP, integrated CODEC, 48kHz sample rate stereo, PCI card with Windows NT low level API |
| Intersense Intertrax2 | Gyroscopic Headtracker (piezo-electric sensors) serial port connected, 3 degrees of freedom (Pitch, Yaw, Roll), internal update rate 256Hz |
| STAX, Basic System II, SR-202 / SRM-212 | Electrostatic Earspeakersystem, Frequency range 7Hz - 41kHz, sound pressure level 100dB / 100Vrms. |

Table 2.1: Prototype components

Head rotation can be used to bring the source into a direction where the resolution of directional hearing is the best, to the front. For this reason the presented system is using a headtracker device which is mounted on the headphones and is providing the host with information about the current head position.

The pointer device in this system was chosen to be a joystick for the reasons stated in the user requirement specifications. A joystick with three axis is employed to provide the user with the maximum of mobility. Besides the normal X and Y transversal movements, this joystick provides a Z rotation and thus support for bringing the source into the direction of maximum resolution via the joystick too.

Table 2.1 summarises the used components and states details on their specifications.

### 2.3.2 System Description

The prototype is a customary personal computer extended with the components mentioned above. The PC is a Pentium III with 1.2GHz, 256MB RAM and the host's operating system is Windows XP™. Figure 2.2 shows a simple block diagram with all key components.

### 2.3.3 Comparison with Common Assistive Technologies

The most common assistive technology for the visually impaired is refreshable braille. These devices are using deform able piezo elements to lift pins within a braille cell to form a valid braille token. Due to the very specialised market just a small number of braille line devices are produced which makes them very expensive. A braille display with 44 cells is priced at about $5,500.-, with 84 cells approximately $10,500.- and they are not liable to the same price decline as computer technology does. Most of the European countries provide funding for disabled people with the need of braille devices if

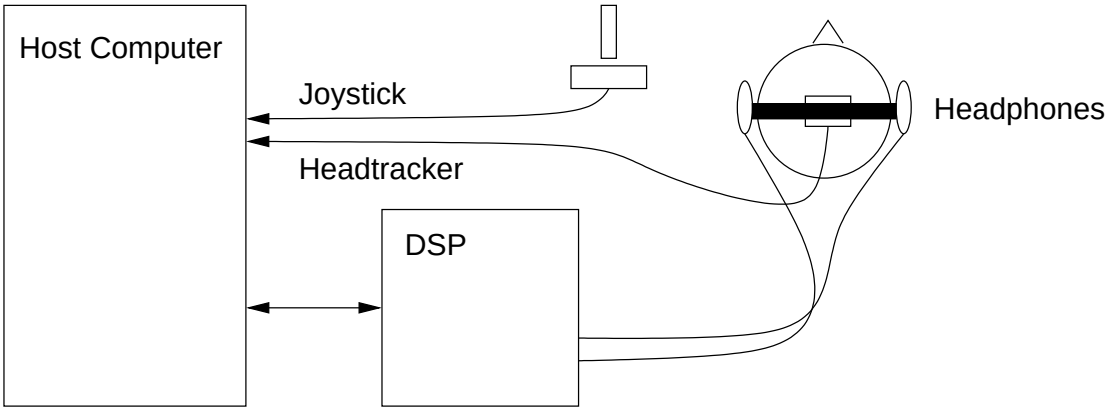


Figure 2.2: Block diagram of the prototype

the necessity is proven. Although screen reader software is available for less than $1000.- visually impaired people who want to use computers in their professional life need to raise a lot of money to become an equally empowered manpower.

The value of the prototype is about $6000.- including a non-commercial DSP evaluation board and a highly sophisticated earspeaker set. If the minimal requirements of this system can be determined and applied to a product, the system would be available at lower costs. This would provide professional computer access to the visually impaired at costs comparable to those of visually unimpaired people.

## 2.4 Design Issues

The major aim of this project is to improve the usability and the effectivity of audio interfaces for the visually impaired. Introducing the possibility of spatialisation provides the interface designer to use various effects of human perception which were not available with monaural techniques. This section is dealing with psychoacoustic effects and sound design approaches to take advantage of spatialisation.

### 2.4.1 The Cocktail Party Effect

The "cocktail party effect" is the ability of listeners to focus the attention on a single sound source among a cacophony of conversations and background noise [15]. This selective attention is known for some time and was investigated from the early 1950's onwards. Much of the early work is related to the problems of flight controllers which had to deal with intermixed voices of many pilots over a single loudspeaker which led to confusions.

The cues which cause this phenomena are not entirely known yet. From the listeners

point of view the task is intuitive and simple but to be able to simulate the effect within a virtual audio reality one must know about the cues which lead to the cocktail party effect to be able to enhance the listeners ability to separate sound sources. It is assumed that physiological and psychological properties of the human hearing are contributing to this effect. An interaction of the auditory system, the central nervous system, but also the high-level perceptual and language processing is thought to be responsible.

The determination of the cues which lead to the cocktail party effect is mostly based on experiments. In [15] an investigation of the literature was conducted and the paper summarises what is known about the effect. The most important cues for sound source segregation were found to be:

- Spatial disparity between channels
- Spatial continuity within channels
  The segregation is more difficult if sources are moved too fast.
- Different voices
  The timbre of different voices is significant to be able to segregate them.
- Pitch shifts between voices
- Content continuity
  Continuous audio streams can be detected better if the content is using words which fit in the respective context.

Despite the technical possibility to provide spatial positioning of sound sources, the interface designer needs also to find suitable properties of the acoustical representation to maximise the ability of sound source segregation.

### 2.4.2 Acoustical Representation, Sound Design

Beside the considerations mentioned above the acoustical representation is also subject to design patterns which are also significant for the amount of information that can be transferred between the system and the user. Audio allows to scale the level of attention. Listeners can perceive background sound subliminally while voice claims the highest attention. The information content of an audio interface, identified during the mapping process can therefore be classified and assigned with different levels of priority. The aim is to find an acoustical representation for information with the lowest need of attention of the listener. In "Designing Audio Aura" [16] an information system was introduced providing low priority information via background auditory cues, that is tied to the people's physical actions in the working place. Depending on the user's position within an office they are provided with information which they might be interested in

For instance, if facing an empty office the system might provide the information that the colleague was in the office earlier this day or that he is attending a conference etc. Being in the canteen the system could inform the user, if a new mail has arrived. They introduced four different sound designs according to the desired levels of attention: sound effects, music, voice, and rich world. Sound effects contain simple sounds which can be perceived with lowest attention, music is extending the earcon model presented in [17] where meaning is carried out through short melodic phrases. Voice is used to provide important information and subsequent the rich world combines all designs into a multi-layered environment. Following these design principals interface designers can tune the sound design to achieve the maximum information flow.

Integration of non-speech sound into audio interfaces and the term "Earcon" was first introduced in [17] where the first design principals were stated. Subsequent structural methods were developed to find a framework which allows interface designers to use non-speech sound most effectively [18]. The two questions that must be answered are: what sounds should be used and where should they be used. In [18] a series of experiments were performed to investigate which sounds can be used without annoying the user but providing the desired information. It also provides strategies to identify the information which can be presented with non-speech sound.

The strategies and principals of sound design are a key issue in designing audio interfaces to be able to take advantage of the spatial factor. Only if the information mapping and auditory representation is well defined, the system can achieve the goal of improving the usability for the visually impaired.

# Chapter 3

# Virtual Audio Reality

The quality of the proposed audio based human-computer interface is ultimately depending on the accuracy of the three-dimensional audio rendering. Although the computing power of DSPs is increasing, the real-time processing of audio data in order to create a virtual environment of high quality is still difficult and under extensive investigation. Key issues of creating virtual audio reality are 1) directional encoding, 2) reflections of the sound in the enclosing room and, 3) reverberation modelling. Algorithms are presented for each of these issues, including the background theory, resource consumption estimations and implementation aspects.

In [19] the current state of the art in developing three-dimensional audio interfaces for the visually impaired is discussed and the paper concludes that accurate generalised HRTF measurements, head-tracking, reverberation and VAR encoding and representation are the areas of most significance for interface designers. This chapter presents solutions for the technical realisation of VAR considering these areas of significance.

## 3.1 Spatial Perception

Spatial audio perception is under investigation for some time but still some mechanisms are not fully understood. One problem is that spatial perception is linked to various psychological effects which are difficult to describe in technical systems. Another is the significant difference of the instruments humans use to perceive audio. The ears of every individual are shaped in a different way and spatial hearing was *learned* with these ears since birth. This makes it a highly personalised matter.

Some general information about which cues are responsible for directional hearing is provided by this section, subsequently the head related transfer functions (HRTFs) are discussed in more detail.

### 3.1.1 Cues Leading to Three-Dimensional Perception

For spatial hearing humans determine the differences of the incident sound waves in both ears. It is assumed that also structure-borne sound via bones is contributing to spatial perception, but no clear cues have been identified by now [7, 21]. To simulate these effects, the cues which influence spatial perception need to be identified.

The most important cue is the inter aural time delay (ITD) [13]. Due to the different location of the ears the sound waves reach the ears temporally delayed. However, some positions result in the same ITD and form the "Cones of Confusion". Another important cue is the inter aural level difference (ILD). ILD refers to the difference of the sound pressure level of the incident sound waves. Again the positioning information is ambiguous at certain points and forms cones of confusion. Both, ILD and ITD are subject to spectral variation caused by diffraction.

Beside these binaural cues, monaural cues exist. They describe the impact of the outer human auditory system consisting of the eardrum, the auditory canal, the pinna, the head and the shoulder echos. This transfer function (HRTF) is dependent on the angle of incidence and, therefore is another important cue for directional hearing.

The ILD and ITD cues can be seen as the result of the comparison of two HRTFs for each ear (for the same sound source the angle of incidence and, therefore, the HRTF is different). For detailed information about spatial hearing refer to [7].

### 3.1.2 Head Related Transfer Functions

No model exists which synthesises the HRTFs from what we know about the human auditory system. In fact, all available data is the result of elaborate experiments where special microphones are placed near the eardrum and the HRTFs are determined from the recorded signals. The major drawback is that the functions are not universally valid, the HRTFs might fit for one listener better than for another. A more or less generalised set of HRTFs can be produced by calculating the average over many measurements or if an artificial head is used which remodels the most common properties of real heads.

The second approach was chosen by the Media Lab at the Massachusetts Institute of Technology. The KEMAR dummy head microphone was used to produce an extensive set of HRTFs at a total of 710 different positions [20]. The set can be downloaded at the Media Lab's Internet site together with useful tools to make the data available to programmes. These KEMAR transfer functions were the base of the HRTFs used with the prototype.

The HRTFs consider the ITD, ILD, head and shoulders shape, the auditory canal and the pinna, but do not contribute to cues which determine the possible distance of sound sources. As mentioned before, the intensity, reverberation and reflections are mainly

responsible for the sense of distance. Possible other cues which contribute to spatial perception and the sense of distance include the distinction between frequency spectra of stimuli at near and far distances or the influence of bone conducted stimuli[21]. However, insufficient empirical data is available and the significance of the cues could not yet be determined fully.

## 3.2 Ambisonic

Ambisonic is a three-dimensional sound field reproduction system. It has the ability to localise sounds to a better degree than stereo or Dolby Surround[1] [22]. Although Ambisonic was originally designed for the use with loudspeaker arrays it was chosen as the directional encoding algorithm in this project because if applied to headphones it offers a number of advantages: 1) economical resource consumption if used to encode more sound sources simultaneously , 2) effective rotation mechanism available, 3) no need for costly and inaccurate interpolations between HRTFs.

This section provides the reader with a short introduction in the Ambisonic theory and shows how the algorithm is adapted to the use with headphones.

### 3.2.1 Theoretical Background

The Ambisonic algorithm is based on the matching of interfering sound waves produced by a loudspeaker array and the original sound wave to be recreated at a certain listening point. A necessary prerequisite for this method is that the sound waves are plane waves. This is a drawback because that is not the case if the listening point is too close to the sound source. However, with providing other distance cues this was not found to be a problem. The recreation of the original sound field can be achieved up to a desired order of spherical harmonics by an Ambisonic system of the same order.

With the prototype only a two-dimensional approach was used. A loudspeaker array in the horizontal plane consisting of $n$ loudspeakers which are reasonable far away from the listening point to produce plane sound waves can reproduce a plane sound wave from any direction up to the $m^{th}$ harmonic where $n = (2m + 1)$. For the three-dimensional approach this would change to $n = (m + 1)^2$. Figure 3.1 shows the Ambisonic array of third order used with the prototype. The aim of Ambisonic is to find the loudspeaker signals which interfere at the listening point to match the original sound wave and therefore create the same perception of the sound field. The matching conditions are developed by representing plane waves as a series of cylindrical Bessel functions and

[1] Dolby is a registered trademark of Dolby Laboratories, Dolby Surround is a trademark of Dolby Laboratories

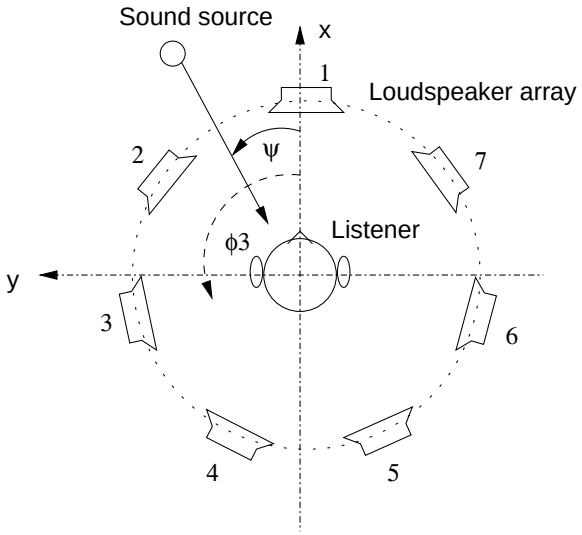


Figure 3.1: Two-dimensional Ambisonic array of third order

comparing similar terms of the equations: on the one hand the series of the original sound wave, on the other the sum of the loudspeaker signals at the listening point. The equations 3.1 to 3.3 are the matching conditions where $P_\psi$ is the sound-pressure level of the source and $P_n$ is the sound-pressure level of the nth loudspeaker.

$$P_\psi = \sum_{n=1}^{N} P_n \tag{3.1}$$

$$P_\psi \cos m\psi = \sum_{n=1}^{N} P_n \cos m\phi_n \tag{3.2}$$

$$P_\psi \sin m\psi = \sum_{n=1}^{N} P_n \sin m\phi_n \tag{3.3}$$

This comparison leads to the Ambisonic signals and is termed "Ambisonic encoding". The directional independent Ambisonic signal $W = P_\psi$ results from 3.1. The Ambisonic signals $X$ and $Y$ are given by:

$$\begin{aligned} X &= P_\psi \cos\psi \\ Y &= P_\psi \sin\psi \end{aligned} \tag{3.4}$$

Subsequently, for systems of second order the Ambisonic signals $U$ and $V$ are given by:

$$\begin{aligned} U &= P_\psi \cos 2\psi \\ V &= P_\psi \sin 2\psi \end{aligned} \tag{3.5}$$

For third order systems the Ambisonic signals $X3$ and $Y3$ are similar to $U$ and $V$, but with $3\psi$ as argument for the cosine and sine-terms.

These Ambisonic signals are not the actual loudspeaker signals but are intermediate results on which some important operations can be performed. A rotation of the whole system can be achieved simply by applying a rotation matrix on the Ambisonic signals. This comes in favour for the use with a headtracker where the whole system is rotated according to the head's orientation. Additionally, Ambisonic signals of different sound sources can be added before calculating the actual loudspeaker signals which makes the system very scaleable and suitable for multiple sound sources as needed with an audio interface.

In the following, "Ambisonic decoding" means to calculate the loudspeaker signals from the given Ambisonic signals. For a system of third order (7 Ambisonic signals) the loudspeaker signals are given by

$$\begin{aligned} P_n &= \frac{1}{N}(W + 2X\cos\phi_n + 2Y\sin\phi_n + \\ &\quad 2U\cos 2\phi_n + 2V\sin 2\phi_n + 2X3\cos 3\phi_n + 2Y3\cos 3\phi_n) \end{aligned} \tag{3.6}$$

For a detailed description of Ambisonic including comparisons with stereo and Dolby techniques, identification of recreation errors and other Ambisonic system considerations refer to [22].

### 3.2.2 Applying the Ambisonic System to Headphones

To be able to simulate spatial positioning of sound sources with headphones, the signals need to be convolved with the head related transfer functions (HRTFs) according to the angle of incidence. HRTFs cannot be found by a mathematical model of the human auditory system but are the result of extensive measurements. Therefore not all arbritrary angles of incidence are available which makes interpolations necessary. Such interpolations are inaccurate but above all very costly.

Ambisonic offers a directional encoding system which results in loudspeaker signals where the array of loudspeakers is pre-defined and located at fixed positions during simulation. This enables us to use a static set of HRTFs without the need of interpolations. The advantages of Ambisonic over the direct encoding with HRTFs can be summarised by:

- Accurate directional encoding without the need of interpolations.
- Simply extendable for multiple sources. Encoding of sources and their reflections can be done economically by summing the Ambisonic signals.
- A static set of only a few HRTFs is needed which saves storage space and makes them exchangeable more easily.
- The most costly part in terms of computational power, the convolution with the HRTFs, is independent of the amount of used sound sources.

| | Operations per second | Memory consumption |
|---|---|---|
| Convolution | 114.4 Mio | 21* frame size |
| FFT | 81.3 Mio | 43* frame size |

Table 3.1: Comparison of FFT and Convolution version of HRTF filtering

More detailed information on binaural reproduction systems using Ambisonic can be obtained from [23, 13, 24].

### 3.2.3 Resource Consumption Estimation

The estimation of the needed CPU power and memory consumption of the Ambisonic algorithm can be divided into three sections: 1) Ambisonic encoding, 2) Ambisonic decoding, 3) HRTF convolution [23]. The estimation is assuming an Ambisonic system of third order in the horizonal plane as used with the prototype.

**Ambisonic encoding :** To calculate the 7 Ambisonic signals from the audio input 7 multiplications with 7 different factors (sine and cosine values) are necessary. The following operations for processing one audio sample are needed: 1 LOAD[2] for the audio sample, 3 LOADs for cosine factors (lookup table), 3 LOADS for sine factors and 7 MPYs[3]. Assuming 16kHz sampling rate the minimum CPU power per source is 224000 operations per second. The needed memory depends on the frame size chosen. The 7 Ambisonic signals need to be stored as whole frames. The preceeding frame is also needed for the convolution.

**Ambisonic decoding :** Decoding the 7 Ambisonic signals to the 7 loudspeaker signals can be simplified to a weighted summation. With 7 LOADs and 7 ADDs[4] the power consumption is 224000 operations per second.

**HRTF convolution :** Applying the HRTFs on the decoded Ambisonic signals can be conducted in two ways. One is to use ordinary convolution which is more costly but needs less memory space. The other is to use the fast Fourier transformation (FFT) to transform the two operands into the frequency domain where they can be multiplied. Doing an inverse FFT results in the desired convolved signal. Table 3.1 provides a consumption estimation for both versions.

The estimations show clearly that the encoding of sound sources is by far less costly than the filtering, but as a matter of fact this filtering only needs to be done once regardless of the number of sound sources.

[2]LOAD: loading operation for one operand

[3]MPY: multiplication of two numbers

[4]ADD: addition of two numbers

## 3.3 Geometrical Room Acoustics

Geometrical room acoustics uses the laws of sound wave propagation to determine the sound field caused by an emitting sound source in an enclosing room. Reflection and diffraction are the two mechanisms which determine a sound field from the geometrical point of view.

The principles of Fermat state that every wave propagation is proceeding from the sender to the receiver in the shortest possible time. If the sound wave impinges on an obstacle it depends on the size of the object whether it is reflected or diffracted by the object. The ruling factor is the wave length. An object with dimensions larger than the wave length is reflecting the wave, smaller dimensions result in diffractions. For audible sound waves this means that objects larger than 17m@20Hz respectively 12mm@20kHz are causing reflections. The reflection follows the two raytracing laws: 1) incoming and outgoing waves lie in the same plane, 2) the angle of incidence is equal to the angle of reflection [25].

Based on these laws the theory of mirror sources was developed which provides a simple approach to calculate the reflections of a sound source within an enclosing room.

### 3.3.1 Purpose

It was determined that early reflections (reaching the listener in a time period of approx. 50ms after the direct sound) are significantly contributing to the sense of space and sound source localisation [26]. In experiments, simulated auralisation and real recording of sound tracks were compared in order to evaluate a virtual auditory environment. It was found that the most audible differences occurred at very low frequencies ($<$ 400Hz) and very high frequencies ($>$8kHz). The low frequency modelling error might be explained by the lack of a diffraction model in the experimental model. At high frequencies the error might be caused by a systematical problem of the experiment. Different headphones equalisation was applied for recorded and auralised sound tracks.

Because of higher order reflections, the number of sound waves which reaches the listener is exponentially increasing with time. This is the reason why only early reflections are capable to contribute to the sound source localisation. After approx. 50ms the density of reflections in a room reaches a value where the listener is only perceiving a decaying reverberation.

### 3.3.2 Algorithm

The theory of mirror sources was introduced in [25] and proposes the mirroring of the sound source at the enclosing walls. These mirror sources can be treated as normal

sources but with additional attenuation because of the wall absorption. The location of the sound sources implies the time delay of the incident sound wave. Figure 3.2 shows an example of mirror sources of second order for a rectangular room.

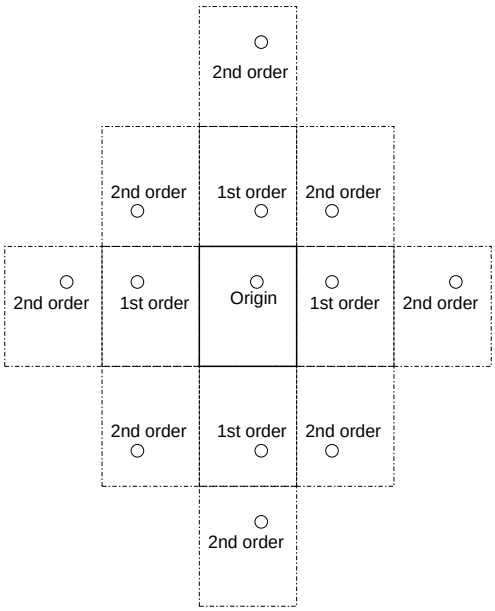


Figure 3.2: Mirror sources of second order for a rectangular room

### 3.3.3 Resource Consumption Estimation

The amount of mirror sources is given by $N = 4(m!)$ for a rectangular room with $m$ being the mirroring order. To determine the position of one mirror source LOADs for the coordinates of the original source, LOADs for the distance to the wall and ADDs are necessary. For a system of second order with 12 mirror sources this results in 34 LOADs and 22 ADDs per original source. Calculating the path and time differences is much more costly because square root functions and MPYs are needed. However, this computational effort is not significant for the overall need of CPU power because the mirror source positions are updated at a very low rate (frame rate) compared to the sampling rate.

The memory resource consumption is not significant for the needed storage of the mirror source properties like position, path difference, time difference and direction, but for the delay line length determined by the maximum delay time. The maximum path difference from a mirror source and the original source to the listener is given by $2 * y$ where $x, y$ are the room dimension and $x < y$. [5] For a typical room of 10m x 10m this means a maximum path difference of 20m or 56ms. At a sampling rate of 16kHz this equals 889

[5] The listener is placed in one corner, the original source in the corner along the longest wall

samples or 2667 samples for 48kHz per source. The memory assignment for the audio buffers needs to consider this minimal length.

## 3.4 Reverberation

In the time after the early reflections the listener can not segregate among single incident reflections, but perceives a stochastic reverberation signal. Figure 3.3 illustrates the time evolution after the direct sound. The late reverberation does not contribute to the

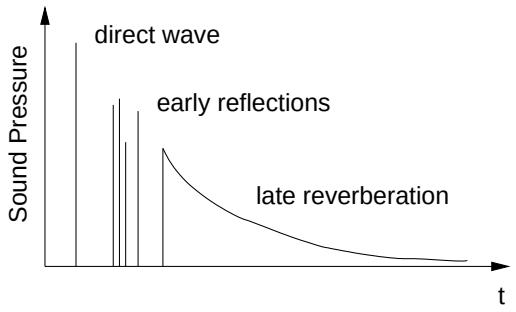


Figure 3.3: Reflections after the direct sound

localisation of sound sources but is significant for the sense of space and determines the properties of the simulated room.

### 3.4.1 Room qualifiers

The late reverberation is important for the sense of ambience. It was found that beside the length of the reverberation, the shape of the reverberation curve is important for the perception too. Digital reverberation generators like the Lexicon 480L™[6] which are mainly employed in professional studios have sets of parameters which describe the room's behaviour.

The most important qualifiers are:

- Reverberation Time
  The reverberation time (RT) depends on the dimensions of the room and the properties of the walls. Reverberation can be perceived until it is attenuated by -15dB because of the presence of direct sound.

- Shape of the Reverberation Curve
  Under certain circumstances reverberation curves are not entirely decaying. They might ascend first, having a maximum in the middle and decay later.

[6] Lexicon 480L is a registered trademark of Lexicon Inc.

- High Frequency Cutoff
  The reflections at the walls lead to an absorption of high frequencies.
- Bass Multiply
  Very low frequencies can have a longer reverberation time.
- Reflection Density
  The intenseness of the reverberation is dependent on the density of the incident reflections.
- Predelay
  The time period between the direct sound and the start of late reverberation is dependent on the relative position of the listener and the sound sources. Up to 40ms delay time is possible before the listener is segregating the direct sound from the reverberation and perceiving it as a different signal.

### 3.4.2 Related Work

In recorded music great effort has been made in simulating reverberation through some electro-acoustic devices. Artificial reverberators based on discrete-time signal processing were first introduced in [27] (early 1960's).

Room reverberation can be obtained by convolving the sound signal with a measured or synthetic impulse response. This technique provides high accuracy but the computational power to do a convolution with long filters is significant. This approach is suitable for predictive auralisation of concert halls or any other static implementation [28]. However, for any interactive manipulation complex updating schemes are necessary so that this method is not applicable in our system.

Generating artificial reverberation without using room impulse responses is mostly based on feedback delay networks (FDNs). Figure 3.4 shows a schematic block diagram [29]. The general class of FDNs for artificial reverberation consists of all-pass or unitary delay

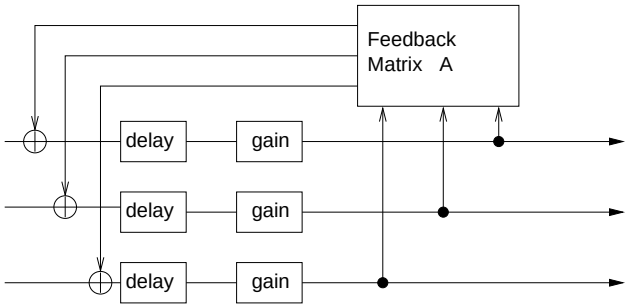


Figure 3.4: Feedback delay network

networks with lossless feedback. This approach cannot guarantee the same degree of

accuracy but it provides an economical method and a good parametrisation for dynamic control. The reverberation time, decay and density can be controlled through the delay length, gain and matrix properties. The analysis and synthesis of parameter sets to obtain natural reverberation in order to recreate a realistic room is shown in [30].

Other reverberation algorithms follow the same principle; Gardner and Datorro proposed algorithms based on complex structures of all-pass filters and delay lines; Matlab[7] implementations and descriptions can be found in [31].

### 3.4.3 Implemented Approach

Because of the limited CPU power available in our system for reverberation, it was decided to employ a FDN algorithm proposed in [28]. The design is similar to figure 3.4 with the difference that 12 input signals are processed, provided by the early reflection module. The signals computed according to the geometrical properties of the room are a suitable input for the reverberation algorithm for two reasons: 1) the use of early reflection signals as input ensures that the late reverberation is succeeding the early reflections as shown in 3.3 and, 2) the time delay between the 12 reflection signals increases the density of the reverberation.

The main control parameter for reverberation is the reverberation time (RT). With a given RT the implemented algorithm is using the following parameters to control the reverberation:

$$\begin{aligned} \text{Delay line length} \quad \sigma &= 3 * \frac{Framelength}{Samplingrate} \\ \text{Gain} \quad g &= 10^{\log(\frac{1}{15})/\frac{RT}{\sigma}} \\ \text{Feedback matrix} \quad A &= \text{Hadamard 12 x 12} \end{aligned} \tag{3.7}$$

The feedback matrix is a uniform Hadamard matrix. It is distributing the energy to all signals creating a high density of reverberation. The matrix provides coefficients with equal amplitudes and can be implemented very efficiently using "butterfly networks". The gain and delay parameters determine the decay of the impulse response. After the implementation of this algorithm in Matlab and after testing various sets of parameters it was decided to fix the length of the delay line for all input lines and to control the RT with the loop gain given by equation 3.7 (the equation refers to an attenuation of $-15$dB because of the reasons mentioned before). Figure 3.5 shows an impulse response of the algorithm using RT=200ms, frame length=7ms.

[7] Matlab is a registered trademark of Mathworks Inc.

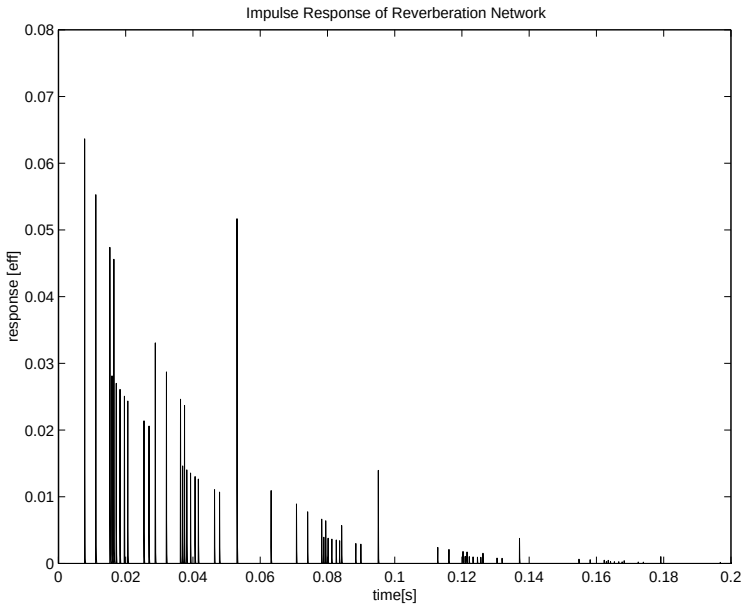


Figure 3.5: Impulse response of the FDN, RT=200ms, frame length=7ms

### 3.4.4 Resource Consumption Estimation

The generation of reverberation is done with the sum of reflections of the sound sources which implies that the resource consumption is independent of the amount of sound sources used. The estimation can be divided into two parts: 1) the memory needed by the delay lines, 2) the CPU power needed to calculate the feedback network.

The needed storage space depends on the chosen length of the delay lines $\sigma$. The implemented approach uses a buffer of 3*frame size per input line. For early reflections of second order this means 12*3*frame size[8] samples as overall buffer size. Additionally, it is necessary to provide memory space for the feedback matrix (144 words) and the gain functions (12 words).

The matrix multiplication for a 12 element vector (input line samples) and the 12 x 12 Hadamard matrix can be calculated with 144 MPYs and 144 ADDs without optimisation. Another 12 ADDs are needed to apply the feedback on the input signal. Including the 144 + 12 + 12 LOADs for loading the operands and the 12 MPYs to apply the gain this makes an overall of 468 operations per sample, accordingly 7,5 MOPS[9] for 16kHz sampling rate or 20,5 MOPS for 44,1kHz sampling rate.

[8]The frame size used in the prototype is 128 samples

[9]Million operations per second

## 3.5 DSP Implementation

The prototype was implemented on a TMDS3260D6701 evaluation board by Texas Instruments using the Code Composer Studio v2[10]. This section discusses implementation issues, provides an overview of the signal path and the interaction of the described algorithms. The data handling is shown and the problem of dynamic updating is discussed.

### 3.5.1 Data Handling

The key issues regarding the data handling are: 1) the control data structure (ControlHPI[11]), 2) the audio buffers and 3) the audio CODEC data management. Only the latter is dealing with the DSP only, the other need to be viewed from the point of the DSP and the host PC.

The CODEC data handling is managed by the internal DMA (direct memory access) controller of the DSP. Two buffers of the size of the used frame are filled alternatingly to provide a DMA channel with data which it delivers to the audio CODEC in respect of the chosen sampling rate. This is the reason why it is necessary to specify the sampling rate at startup and why it cannot be altered during operation. If the DMA channel has finished the transmission of one frame, an interrupt is triggered which causes the call of an interrupt service routine. There, the next buffer with valid output data is specified and all the signal processing is done to be able to provide the following frame.

The ControlHPI is a predefined memory space in the DSP's memory which contains all the data to control the behaviour of the programme. It provides information about the user, the sound sources, the room and general control switches. For a detailed description see appendix A. The ControlHPI structure is accessible to both, the DSP and the host PC simultaneously. Besides some profiling information the host PC writes the information and the DSP is reading it. The transfer is conducted via the PCI interface using the low-level API by Texas Instruments.

Providing the DSP programme with the audio data in real time is the most critical part of the data handling. Because of the reasons mentioned in section 3.3.3 the audio buffers have to have a minimal length and a certain data section which must not be overwritten. The approach to solve this is a ring buffer topology. A memory section, reserved as the audio buffer for one sound source, is adapted as a ring buffer by using modulo operations and is controlled by two pointers which segregate the DSP area from the host PC area. Figure 3.6 illustrates the approach. While the host is moving forward pointer P1 by writing the audio data in section $\overline{\text{P1 P2}}$, the DSP is freeing memory by moving pointer P2 whenever data in $\overline{\text{P2 P1}}$ is not needed anymore. The DSP has to keep the DSP area

[10] Code Composer Studio is a registered trademark by Texas Instruments
[11] Control Host Port Interface

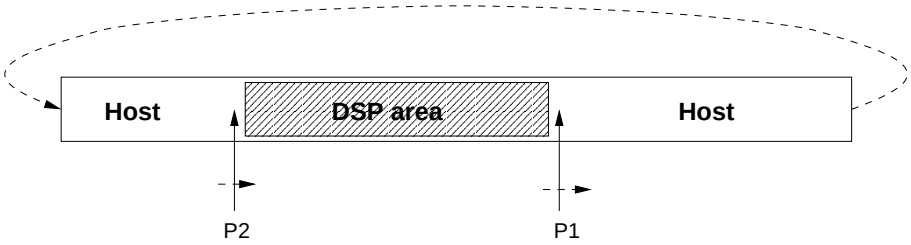


Figure 3.6: Ring buffer topology with two pointers

big enough to hold all sound pointers (direct sound and mirror sources) but also takes care about freeing sufficient memory for the host to be able to do effective and save buffering. This buffering is also considering that the host PC does not run under a real time operating system and the availability of data can not be fully guaranteed at any time.

Considering the capabilities of the EVM board the memory sections were allocated as shown in table 3.2. This limits the prototype system to 15 simultaneous sound sources.

| Length | Location | Description |
|---|---|---|
| 8 kByte | internal | ControlHPI structure |
| 56 kByte | internal | Data section (all variables, stack etc.) |
| 64 kByte | internal | Programme section (executable code) |
| 240 kByte | SBSRAM | Audio buffers (15 à 16 kByte which equals 8192 samples, 0.5s@16kHz per ring buffer) |
| 16 kByte | SBSRAM | Delay lines for reverberation (divided into 12 input lines à 768 Byte or 384 samples, 24ms@16kHz) |

Table 3.2: Memory allocation at the DSP

The SBSRAM is the fastest external RAM available on the board (running at 133MHz), any access to the slower SDRAM is decreasing the performance of the system significantly which results in less possible sources and/or a lower possible sampling rate.

### 3.5.2 Algorithms

The interaction of the proposed algorithms is illustrated in Figure 3.7. The block diagram shows the signal path of a single sound source, defines the input and output data of each block and identifies the interfaces where data from other sound sources are contributing to the final output. It is important to point out that the early reflections module and the Ambisonic encode module is serving every sound source while the reverberation and Ambisonic decode sections are dealing with sums of sound source signals. This implies

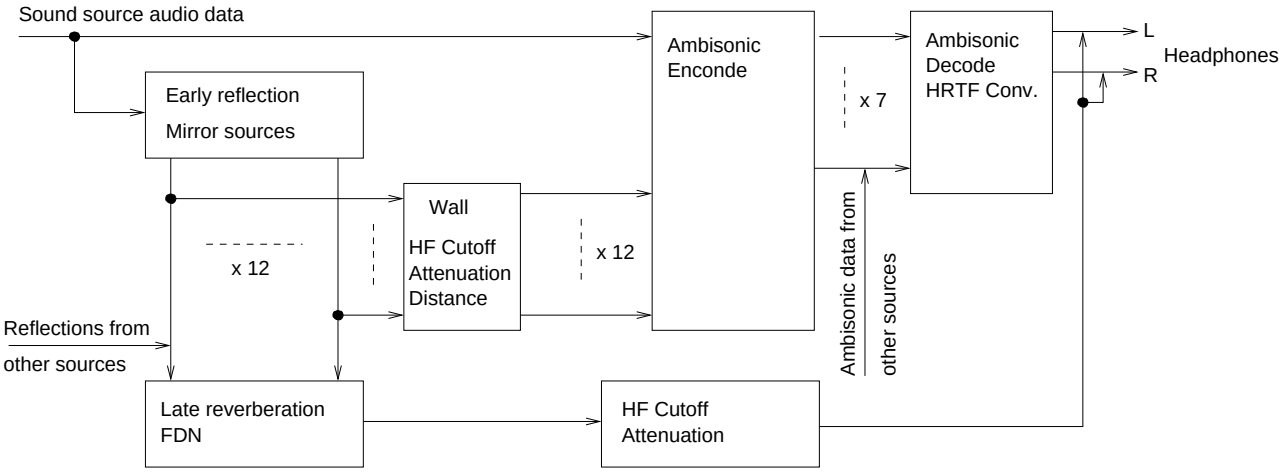


Figure 3.7: Block diagram for the signal path of a single sound source

that only the early reflections and the Ambisonic encode module consume resources depending on the amount of sound sources while the others have constant consumption.

Subsequently, the implementation of the algorithms is described in detail. For more information refer to the software documentation [32].

### Ambisonic

The Ambisonic algorithm of third order was implemented in a straight-forward way. Encoding is using a cosine lookup table to improve performance, decoding employs a hand optimised filter convolution function provided by the standard mathematical library (DSPLIB).

The used HRTFs were extracted from the download-able data from the MIT Media Lab [20]. They were modified to fit the chosen sampling rate and cut to the used frame size. Ambisonic decoding, thus calculating the loudspeaker signals, was integrated into the HRTFs so that these modified transfer functions can be applied to the Ambisonic signals directly [23].

### Early Reflections

The prototype system calculates the mirror sources of second order for each real sound source. The properties of these mirror sources include the direction considering the user orientation, the path difference to the user and the time difference compared to the direct sound. The implementation is using pointers in the audio buffer which may be adjusted to the delay time so that no additional memory for delay lines is needed. Figure 3.8 illustrates the approach.

If the sources are statically positioned in a room there is no need to calculate the mirror

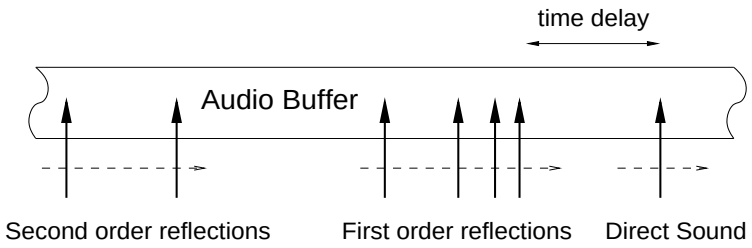


Figure 3.8: Mirror source pointers in the audio buffer

source positions online. However, the implementation in the prototype is calculating the positions with every processed frame which provides the user application with the possibility to use moved sources. The updating of the pointers is realised with a variable delay through linear interpolation. This method is also applicable to the direct sound pointer which makes the modelling of the Doppler effect possible. Although the implementation provides this possibility, the prototype is not taking advantage of it because for the small velocities used in our experiments it would have only minor effects.

#### Late Reverberation

The late reverberation algorithm was implemented as shown in section 3.4.3. The twelve delay lines are filled with the accumulated early reflection signals from each active sound source. Like the Ambisonic decoding the calculation of the reverberation output is invoked after processing all sound sources which makes it's resource consumption independent of the amount of sound sources. The reverberation time parameter which is part of the ControlHPI and accessible from the user application, controls the gain in the feedback loop according to equation 3.7. The feedback matrix is also part of the ControlHPI and therefore adjustable if desired; a Hadamard matrix is used by default.

### 3.5.3 System Latency

Latencies are significant in two areas of the system: 1) The signal latency is determined by the needed processing time of the audio data. 2) The parameter latency refers to update rates and update mechanisms of parameters controlling the DSP programme.

The audio processing on the DSP is done in real-time. This means that the programme is fed with a frame of data and the processing has to be accomplished in the time corresponding to the frame. The time depends on the sampling rate and the frame length. These two attributes are therefore also determining the signal latency. The time to provide the DSP programme with audio data does only contribute to the signal latency when a new buffer starts playing and the first chunk of data has to be transferred (see section 4.1.2).

The problem of changing parameters affects mainly the positioning data of the listener and the sound sources. These values are updated permanently by the user application while parameters like room properties are fixed at the system's start in most cases. The aim is to ensure that the system reacts on altered parameters as fast and accurate as possible, but also prevents that the user hears non-continuous changes which might lead to disorientation.

It was found that especially the listeners orientation is a critical value because it has the highest rate of change. The importance of the use of minor head rotations for orientation was discussed in 2.3.1. The new value for the head and user orientation is read by the DSP programme every time a single frame processed. Updating the listener orientation is done by modifying it by a quarter of the difference between the preceeding value and the new value. This implies a certain system latency but it also ensures that the user does not notice steps while updating the orientation. The maximum value which the orientation may be altered was determined by experiments and the implemented approach was found to be sufficient for the experiments, but it is based on some assumptions. The implemented update algorithm is neither considering different frame lengths or sampling rates nor the update rate and the maximum angle velocity of the used headtracker and joystick. Further investigations are necessary to develop a stable update algorithm which can be used under different prerequisites.

The overlying API is responsible for the update of the listener and source positions. The DSP programme is reading the current values from the ControlHPI with every new frame without restrictions of the maximum value of change. This can cause problems because user applications have direct access to these values and for future versions of the programme an update algorithm needs to be developed to restrict the maximum speed of listeners and sound sources on the level of the DSP programme.

# Chapter 4

# Application Programming Interface

The aim of an application programming interface is to introduce a layer to the system hierarchy which fulfils the following requirements: 1) providing full functionality of the underlying layers while hiding the way it is done (behaviour hiding), 2) encapsulating data which is not necessarily important to the user of the API (data hiding), 3) providing a standard interface to user applications which might, if well designed, be longer valid than the implementation of the interface and, 4) making user applications independent of the used hardware by providing the same functionality for different equipment (hardware hiding). An application programming interface is potentially improving the quality of a system's architecture which is defined by key properties like functionality, usability, security, modifiability, portability, reusability, integrability and testability [33].

This chapter presents two APIs which provide user applications with intuitive functions to use the proposed audio rendering system. Section 4.1 describes the AISound3D API which was developed as the interface to the DSP programme described in chapter 3. The second API proposed in section 4.2 provides hardware independent functionality to user applications for input devices like the headtracker and the joystick. Figure 4.1 illustrates the cooperation of the APIs with their neighbouring layers. The user application which is modelling a virtual audio reality incorporates the APIs in order to realise the VAR. Setting up the scene is done with methods from the AISound3D API. It controls the DSP programme and is responsible for providing it with the necessary audio data. During simulation the user application can obtain information about the current state of the feedback devices from the AIInput API and adjust the scene accordingly.

Related development was conducted at Bell Laboratories where a multilayered software architecture for real-time audio rendering was introduced [34]. In comparison to our approach this architecture intends to allow a user application to perform low level DSP operations over audio data but also provides a layer for geometry based room acoustics.

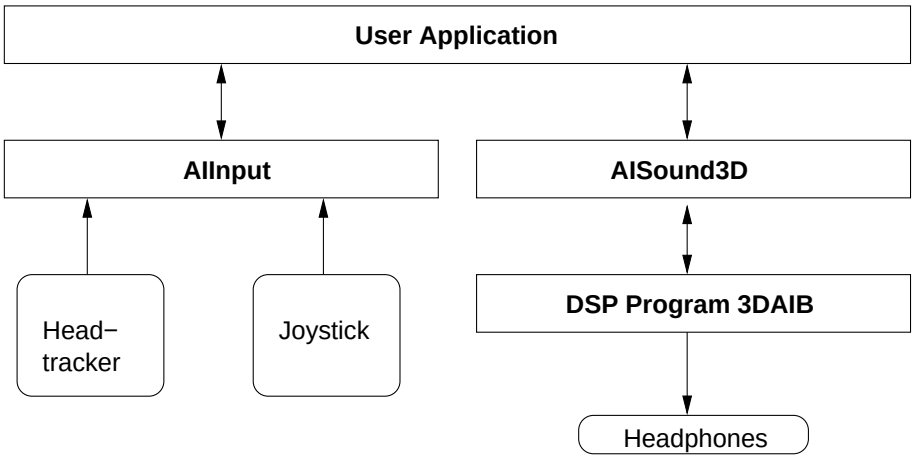


Figure 4.1: Software Architecture

This might be more generic, but does imply the necessity of deeper understanding in the field of signal processing by the user to create a virtual environment.

Subsequently, the two APIs are described in more detail and an example of creating a scene is given in order to clarify the APIs usage.

## 4.1 AISound3D

The AISound3D API was developed to provide control over the creation process of a virtual audio reality. With the availability of commercial hardware which employs powerful signal processing, many interface standards for audio rendering were proposed from various organisations. As a matter of fact most of the standards do not differ from each other very much. All of them base their interface design on listeners and sound sources as objects provided with the needed properties and functionality. The most common interface is DirectSound3D[1] which is part of Microsoft's DirectX component structure. DirectX consists of an extensive collection of APIs which handle graphical output, feedback devices and sound output with the aim to provide game developers with a common platform. OpenAL[2] is one of the alternative standards but did not reach the distribution of DirectSound3D. For these reasons the AISound3D API was developed in the style of DirectSound3D to introduce a quasi compliant standard for VAR creation. It was designed to be adapted to full compliancy with the minimal possible effort, but due to the extended requirements of our application and the complexity of DirectSound3D it was decided to develop an independent interface design for a start.

The AISound3D API employs a low-level driver API to access the DSP board. This driver

[1] DirectX and DirectSound3D are registered trademarks of Microsoft Inc.

[2] `www.openal.org` maintained and hosted by Loki Entertainment Software

is responsible for all low-level communication with the hosted DSP board via the PCI interface. It provides direct reading and writing of DSP memory and is able to control the DSP's state using the host port interface (HPI) which manages the communication at the DSP. A detailed description of the driver's capabilities is available from the technical documentation of the EVM board [35].

Another third-party component was used to support a wide range of audio data formats. Libsndfile[3] is a C library for reading and writing files containing sampled sound through one standard interface. It supports Microsoft WAV, SGI AIFF, Sun AU and, raw data formats among others. The source code is published under the Gnu Lesser General public license[4] and was developed platform independently to function on different operating systems like MacOS, Windows or Linux.

AISound3D was implemented in C++ and is extensively employing object orientated design patterns. Figure 4.2 shows the class diagram resulted from the design process which describes the most important components and their relations. A complete class diagram containing all methods and attributes of the classes following the UML[5] conventions can be found in appendix B.

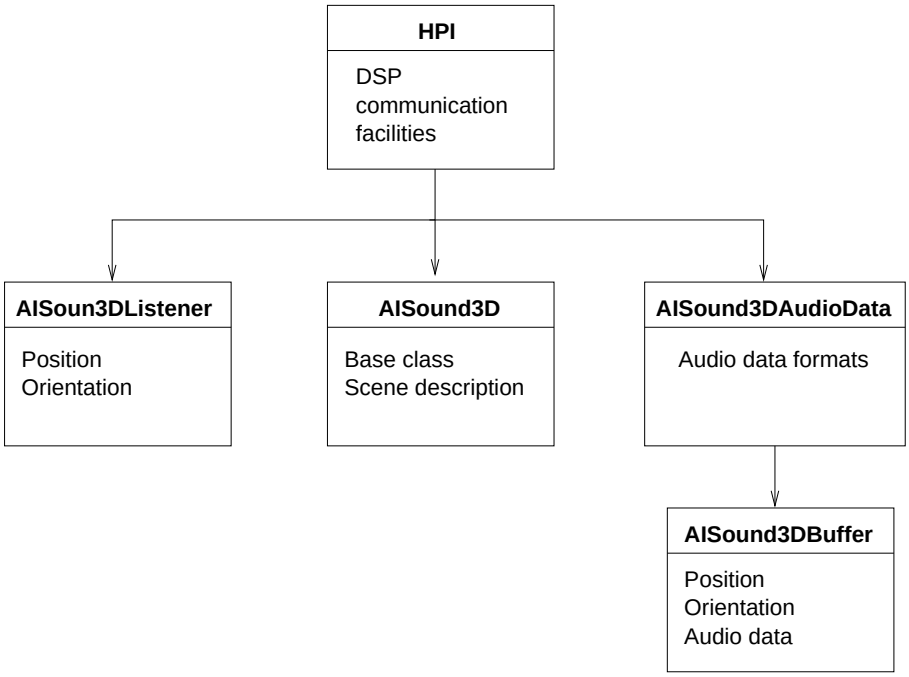


Figure 4.2: Class diagram

In the centre of the design a main class called AISound3D manages the virtual scene. It defines the room in which the scene is taking place and handles references to the

[3] http://www.zip.com.au/~erikd/libsndfile
[4] The copyright is controlled by http://www.gnu.org/copyleft/lesser.html
[5] Unified Modelling Language http://www.uml.org

scene's content objects like listeners and sound sources. In the following sections the most important methods and data structures required to create a virtual audio reality are described in detail. For a complete reference to the API refer to the online documentation available from the project home-page[6].

### 4.1.1 Main Class

The main class AISound3D is responsible of the following tasks: 1) initialising the connection to the DSP board and the board itself, 2) defining the room and its qualifiers and, 3) managing the listener and sound source objects in the room. The following sections describe the provided functionalities for these tasks:

#### Initialisation

The `init` method establishes the connection to the DSP and performs all initialising tasks. The DSP programme is loaded, valid default values are written to the ControlHPI section and the sampling rate is set. Finally, the DSP is set into the running state from where on the control is conducted only via the ControlHPI structure. This method must be launched before any other method can be used, but must not be used more than once. The object's destructor method is responsible of resetting and disconnecting the board.

#### Environment Definition

The properties of the enclosing environment are determined by an attribute of the AISound3D object which encapsulates the values within a data structure. By using the `getRoom` and `setRoom` method an AIS3DRoom structure, containing all information about the room (see 4.1.4), can be obtained, or set.

#### Buffer Management

All used sound sources need to be registered with the base class before they can be used by the user application. The reason for this are the limited resources at the DSP. The main class AISound3D controls the maximum amount of sound sources which can be created. This implies that the sound buffers have not only to be created by the `createBuffer` method, but also need to be removed using the `killBuffer` method to keep the registry up to date. In the current configuration a maximum of 15 sound sources can be created (see 3.2 for the memory allocation at the DSP).

[6] `http://spsc.inw.tugraz.at/3DAIB/`

### Listener Management

As a matter of fact, only one listener can be served at a time so that an attribute of the main class determines the currently active listener. To be able to switch fast between listeners, for example in order to change its position, it is possible to exchange the active listener while operating using the `get/setActiveListeners` methods. Listeners may be created without the proposed `createListener` method, but it was introduced as a convenient way of creating an instance and setting it as the active listener in one step.

### Debugging

In the *Debug*-configuration of the API the base class provides a method to obtain the DSP status and ControlHPI memory section. This is easing development as one can dump the state of the programme and print it to the standard output.

## 4.1.2 Audio Buffers

According to the DirectSound3DBuffer structures the AlSound3D API also proposes a class which describes a sound source with all needed properties as a buffer. The buffer properties are encapsulated in the AlS3DBuffer data structure to have all key data combined (see section 4.1.4). This includes the data source, the position, the orientation and the velocity. Additionally, simple methods are provided to switch the sound source on (play) or to mute (pause) it.

As illustrated in figure 4.2 the AlSound3DBuffer class not only inherits attributes and methods from the base communication class HPI, it is also derived from the AlSound3DAudioData class which is not intended to be used directly by a user application. The AlSound3DAudioData class implements functionality for handling audio data from files of different format types employing the libsndfile library to access the data. The class abstracts the sound buffer data and provides the derived AlSound3DBuffer with methods to read whole frames and to obtain information about the data status. As far as the functionality is visible in the AlSound3DBuffer class the methods are described here.

When created, an AlSound3DBuffer instance launches a separate thread which is defined as part of the class. This thread is responsible for reading the audio data and filling it into the respective memory section at the DSP. With every frame transmitted it is accessing the attributes of its "mother"-instance of AlSound3DBuffer to receive order how to operate (play, pause, end of file). The multi-threading approach is necessary to ensure that more buffers can be used simultaneously and the data streaming is conducted in the background. Figure 4.3 illustrates the collaboration of the thread and its relating AlSound3DBuffer object.

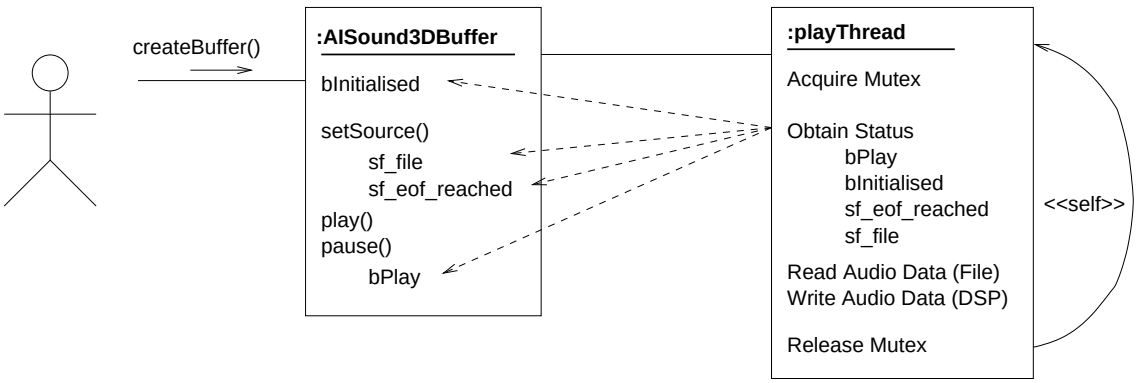


Figure 4.3: Collaboration diagram (UML standard) for the playThread method

The functional requirements for this class can be separated into the following sections:

### Buffer Property Management

The properties of the buffer can be altered by replacing the used AlS3DBuffer data structure which contains all necessary fields (see section 4.1.4). User applications have easy access to the data structure using the `getAllParameter` respectively the `setAllParameter` method.

### Data Source Management

A data source can be specified for this buffer using a string for the file name. The string might contain absolute or relative paths in order to localise the file on the system's file-system. A method is provided which opens the file non-exclusively for reading, if the provided format can be determined from the header and is supported. The format does not have to be specified by the user application. The data source may be altered even while the buffer is playing. For playing in an infinite loop there is a switch implemented in the AlS3DBuffer control structure. The class object re-opens the file when the EOF[7] is reached.

### Buffer Actions

Two simple methods cause the buffer to start streaming audio data to the DSP or to pause streaming. There was no "stop" method implemented because for starting the same audio data from the beginning it is proposed to reassign the data source. However, for future versions of this API a "rewind" and a "fast-forward" function could be useful. Internally, the play and pause methods do switch just one boolean variable which is

[7]End of file mark

queried by the streaming thread whenever it has transmitted a frame. The latency for pausing or playing the stream is therefore one frame at the maximum. For the currently used frame size of 2000 samples this equals to 125ms@16kHz sampling rate.

### 4.1.3 Listeners

Although the proposed system allows only one active listener, it is improving the usability of the API to provide a separate AISound3DListener class to stress information hiding and behaviour hiding. The properties of a listener are encapsulated in the according data structure AIS3DListener (see section4.1.4). Additionally, it enables a user application to configure an additional listener with no impact on the ongoing simulation. A newly configured listener can be used to model fast geometrical transformations within a room because the API allows to switch the active listener during the simulation. The listener might therefore be re-positioned easily.

The listener class only provides functions to obtain or set the property data structure at the time being. However, except for the reason of the desired compatibility to DirectSound3D, there is other good reason to implement this class. Further functionality for listeners like uploading personalised HRTFs can be integrated easily without the need of changing existing user applications.

### 4.1.4 Control Structures

The used data structures centralise the key properties of the objects within the virtual scene. There is a data structure introduced for the room, the listener and the sound sources, respectively. Instances of the structures can be created independently from the class objects. No changes take effect until the class objects are called to use the structures. This is a convenient way of altering more properties of objects during operation. Additionally, the clarity in terms of simple usability is improved by encapsulating related information. Subsequently, the three data structures with a short description of their fields are provided. For detailed information refer to the API online documentation[8].

[8] `http://spsc.inw.tugraz.at/3DAIB/`

### AIS3DRoom

| Type | Identifier | Description | Unit |
|---|---|---|---|
| uint | iDimension[2] | The room size [x,y] | m |
| int | iEarlyWallDamping | The attenuation caused by the reflection on a wall | dB. |
| uint | iEarlyPredelay | An additional time period from direct sound wave and first early reflection | ms |
| uint | iEarlyCutOff | Cut-off-frequency of a high-pass in order to attenuate high frequencies after reflection | Hz |
| uint | iReverbTime | The reverberation time | ms |
| uint | iReverbCutOff | Cut-off-frequency of a high-pass in order to attenuate reverberation signals | Hz |
| uint | iReverbSigma | The delay line length for the reverberation algorithm (currently unused because fixed) | ms |
| uint | iReverbFB[12][12] | Feedback matrix for the reverberation algorithm | - |
| float | fMixDirEarly | Mixing directive for direct sound and early reflection signals | - |
| float | fMixDirReverb | Mixing directive for direct sound and reverberation signals | - |

### AIS3DBuffer

| Type | Identifier | Description | Unit |
|---|---|---|---|
| float | fPosition[2] | Position of the sound source [x,y] | m |
| int | iVelocity | Velocity for autonomous movements (not yet implemented) | m/s |
| uint | iMinDistance | Minimal distance between the sound source and the listener. For distances less than this value the $1/r$ rule is not applied any more to prevent clipping | m |
| int | iOrientation | Orientation of the sound source. Makes other than isotropic emitting sound sources possible (not yet implemented) | ° |
| bool | bLoop | Switch to infinite loop play | - |

**AIS3DListener**

| Type | Identifier | Description | Unit |
|---|---|---|---|
| float | fPosition[2] | Position of the listener [x,y] | m |
| int | iVelocity | Velocity for autonomous movements (not yet implemented) | m/s |
| int | iHeadOrientation | Orientation of the head | ° |
| int | iOrientation | Orientation of the body | ° |

## 4.2 AIInput

The goal of the AIInput API is to introduce a layer between the hardware components and the user application. Thus, abstracting the hardware provides the user application with a common interface for a wide range of hardware and easily allows the integration of new hardware without the need of changing any user application code. This approach is following the intention of the DirectInput API proposed by Microsoft as part of DirectX. Internally it uses the driver APIs for the headtracker provided by its manufacturer and the DirectX interface to access a joystick. In the current version of the software there is just one type of headtracker supported, but any kind of DirectX compatible joystick may be used.

After creating an instance from the class, the following methods are provided regarding the two supported types of input devices:

### 4.2.1 Headtracker

The headtracker needs to be initialised with the `initHeadtracker` method which reports back whether the operation succeeded or not. The underlying API from Intersense checks all possible connectors for any type of headtracker of this branch. The headtracker is reset and calibrated during initialisation which implies that the zero angle position is determined from the position of the headtracker at startup time. This might not be the desired starting point and that is why the AIInput class provides a `resetHeadTracker` method to calibrate the headtracker to the zero angle position whenever it is needed. This is also indispensable due to the gyroscopic technology of the used headtracker. There was a significant drift observed while operating especially when very slow angle rates were occurring so that periodical calibration was compulsory.

To obtain data from the headtracker, a method was implemented which delivers a AIIHeadTrackerData structure with all information encapsulated. This structure contains:

**AllHeadTrackerData**

| Type | Identifier | Description | Unit |
|---|---|---|---|
| float | fOrientation[3] | Values for all three degrees of freedom; Azimuth, Elevation and Roll | ° |
| float | fTimeStamp | A timestamp provided by the headtracker API to have control over the update rate | - |

### 4.2.2 Joystick

Similar to the headtracker functions the AllInput class provides an `initJoystick` method to obtain control over a connected joystick. The API is using the DirectInput API and if any compatible joystick is found and can be acquired, the initialising method reports the success back to the user application. There is no need for a calibrating function as there was with the headtracker because this functionality is integrated into the Windows operating system as part of the "Control Panel".

To provide the data within a similar structure, the corresponding `getJoystickData` method returns an AllJoystickData structure containing the following fields:

**AllJoystickData**

| Type | Identifier | Description | Unit |
|---|---|---|---|
| long | lAxis[3] | Values for the three axis used: x Axis (left-/right), y Axis (forward/backward), Rx Axis (roll right/left) | - |
| float | fTimeStamp | A timestamp to have control over the update rate | - |

## 4.3 Creating a Scene

The following use-case is intended to clarify the usage of the proposed API in order to create a virtual audio reality. Starting with designing the desired environment, this section provides a guideline of how to realise the VAR from the user applications point of view. The presented code is legal C++ standard and might be used as written down here, but does not consider any framework, e.g. an additional graphical output or any timer initialisation.

As a meaningful example the desktop replacement introduced in section 2.1 will be the base for this use-case. A commonly used Windows desktop will be mapped to a virtual audio reality and it is shown how such a possible user application is created using the proposed libraries. The intention of this section is to show the API capabilities and

not to conduct extensive usability design so that proper sound data for this example is assumed to be available.

The screen is mapped to a rectangular room of 10m by 10m. The display contains shortcuts to four applications, shortcuts to private documents and the trash container. To increase usability, the applications will be gathered in the upper left corner, the private documents in the upper right and the trash is placed in the lower right corner. The applications and documents are grouped. This means that they are represented by only one source while the user is reasonably far away, but if the user comes close enough to be able to segregate the individual sources, the group source is muted and the individual sources are played instead. Figure 4.4 shows the desired design.

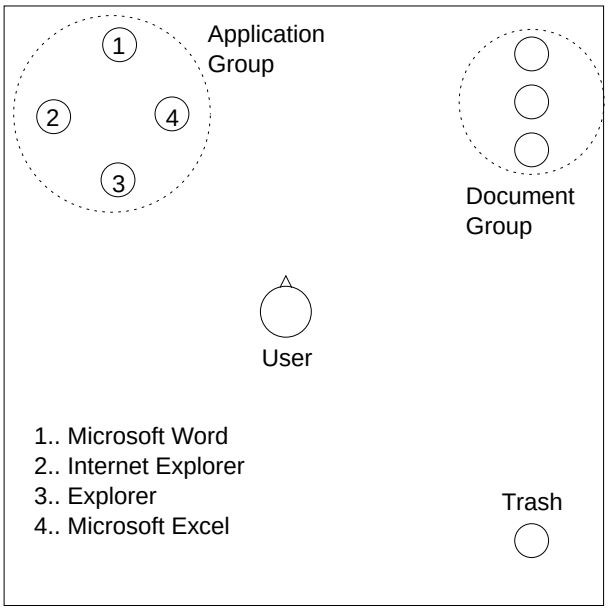


Figure 4.4: The desired design of a VAR mapped from a common Windows desktop

The following sections show how to initialise the API, create the environment and manage the content.

### 4.3.1 Creating the Environment

To be able to use the API, the header files AlSound3D.h and AlInput.h need to be included and made available in the compiler path such as the libraries need to be available to the linker.

Before any creation definition can be used, the API needs to be initialised. In this stage all connections are set up and the DSP is loaded with all necessary data. In this example the sampling rate is set to 16kHz.

Listing 4.1: Initialisation

```
aiSound = new AlSound3D();
aiInput = new AlInput();
if (!aiSound->init(16000)) { exit(-1); }
if (!(aiInput->initHeadTracker()) ||
    !(aiInput->initJoystick())) {
      exit(-1);
}
```

Subsequently, the room can be defined with:

Listing 4.2: Room Definition

```
AlS3DRoom *aiRoom;
aiRoom = aiSound->getRoom();
aiRoom->iDimension[0]=10;
aiRoom->iDimension[1]=10;
aiSound->setRoom(aiRoom);
```

The room properties are first obtained from the DSP because the received structure contains all the default values set by the DSP. This is the proposed procedure to ensure that the structure contains reasonable values.

To complete the initialisation process, the listener is positioned at the starting point and is set active to the system:

Listing 4.3: Listener Setting

```
AlS3DListener *aiListenerProp = new AlS3DListener;
aiListenerProp->fPosition[0]=5;
aiListenerProp->fPosition[1]=5;
aiListenerProp->iOrientation=0;
aiListenerProp->iHeadOrientation=0;
aiListener = new AlSound3DListener();
aiListener->setAllParameters(aiListenerProp);
aiSound->setActiveListener(aiListener);
```

The next task is to introduce mobility to the system. Therefore the properties of the listener need to be updated periodically with the data from AlInput. The headtracker provides a maximum update rate of 256 samples per second which sets the upper limit for the user application. It was found sufficiently to update the listener properties 30 times per second. The following code illustrates how to update the structures. It should be placed in a timer callback function to be periodically executed.

Listing 4.4: Listener update

```
aiInput->getHeadTrackerData(&aiTracker);
aiInput->getJoystickData(&aiJoystick);
```

```
aiListenerProp = aiListener->getAllParameters();
aiListenerProp->iHeadOrientation=(int)aiTracker.fOrientation[0];
aiListenerProp->iOrientation +=(int)aiJoystick.lAxis[2]*2/1000;
if (aiListenerProp->iOrientation < 0)
    aiListenerProp->iOrientation+=360;
aiListenerProp->iOrientation %= 360;
fPhi = ((float)aiListenerProp->iOrientation)/360.0*2*3.141528;
aiListenerProp->fPosition[0] +=
    (0.05*((float)aiJoystick.lAxis[0])/1000.0) * cos(fPhi) +
    (0.05*((float)aiJoystick.lAxis[1])/1000.0) * sin(fPhi);
aiListenerProp->fPosition[1] -=
    (0.05*((float)aiJoystick.lAxis[0])/1000.0) * sin(fPhi) +
    (0.05*((float)aiJoystick.lAxis[1])/1000.0) * cos(fPhi);
aiListener->setAllParameters(aiListenerProp);
aiSound->setActiveListener(aiListener);
```

This code updates the head orientation of the user according to the headtracker data and the user's position and body orientation with data from the joystick. The scaling factors were determined by experiments and are not ideal because of the reasons mentioned in section 3.5.3. In future versions of this API the position should be altered by setting the velocity property of the listener only.

At this point the environment is set up and the DSP programme is running. The following section describes how to introduce the sound sources to the scene.

### 4.3.2 Managing Audio Sources

Sound buffers can be created very similarly to listener objects. In this use-case we need to define 10 sound buffers: 4 applications, 3 private documents, the trash container and the two sound sources which cover the two groups (applications and documents). The creation of one buffer is shown here as representative for all the others:

Listing 4.5: Buffer Creation

```
AlSound3DBuffer aiBuffer[10];
...
  aiBuffer[i] = aiSound->createBuffer();
  aiBufferProp = new AlS3DBuffer;
  aiBufferProp->fPosition[0]=1;
  aiBufferProp->fPosition[1]=2;
  aiBufferProp->bLoop = true;
  aiBuffer[i]->setSource("WindowsWord.wav");
  aiBuffer[i]->setAllParameters(aiBufferProp);
```

In this example the Windows Word earcon is created with the data source file *WindowsWord.wav* and the `bLoop` switch set which causes the system to play it in an infinite loop.

The sound sources are not audible until the according `play` method is issued. To be able to implement the grouping facility, either the group sound source or the single earcons of the elements should be active depending on the listeners position. The following example shows how to set the appropriate sound sources active. This code should also be placed in the timer callback to be executed periodically.

Listing 4.6: Grouping of Elements

```
if ((aiListenerProp->fPosition[0] < 4) &&
    (aiListenerProp->fPosition[1] > 7)) {
   aiBuffer[0]->pause();
   for (i=1;i<5;i++) aiBuffer[i]->play();
} else {
   aiBuffer[0]->play();
   for (i=1;i<5;i++) aiBuffer[i]->pause();
}
```

This causes the system to play the grouping source whenever the listener is outside of a certain region of the room. If the listener is within the region the grouping source is muted and the single elements are played. This increases the usability and introduces a hierarchy of sounds to the system.

The system is now up and running and the user may move around in the room and explore its content. In order to link the earcons with actions like launching a programme or starting a screen-reader for accessing documents, the user-application has to determine the listeners position and other feedback actions like joystick buttons. The API does not support this embedded functionality at the moment, but it might be introduced easily in further development.

At the time being, the performance of the DSP programme does not allow us to exploit the possible 15 sources. It currently supports only 4 sources simultaneously because the code is still in alpha stage and not optimised. Therefore, this use-case can only be implemented if the unused buffers are killed before new ones are opened to keep the amount of active sources low.

.

# Chapter 5

# Conclusion

The proposed audio rendering system is capable of producing virtual audio reality in order to model a user interface for people with visual impairment. The current state of development allows user applications to create VAR and provides techniques to manage its content. The first contacts of the target group with the system were encouraging and the approach to exploit the capabilities of the human auditory system to improve the access for visually impaired and blind people to computers is promising. However, there is much investigation underway in the field of three-dimensional audio rendering, but comparatively little is conducted in using it as computer interfaces.

The current system is still in the prototype stage. It employs an evaluation DSP board as key hardware device because customary sound cards would neither provide sufficient computational power, nor would it be possible to program them in this way. However, the proposed system showed that it is possible to create an interactive audio interface. The following list summarises the state of the development:

- Virtual audio reality simulation with
  - Ambisonic directional hearing
  - Geometrical room acoustics, early reflections of $2^{nd}$ order (12 in a rectangular room)
  - Reverberation algorithm
- Rectangular room with up to four sound sources simultaneously
- Free positioning of user and sources
- Interactive movement of the listener
- DirectSound3D like programming interface

- DirectInput like programming interfaces supporting the Intersense Intertrax2 head-tracker and any DirectInput compatible joystick
- Sound files as data source in various formats
- Test-platform providing all implemented features of the proposed libraries

The employed technologies are state of the art, but there is no other approach available at the moment which combines these techniques to a VAR system as computer interface for the visually impaired. In [19] the author concludes that there is still a long way to go before arriving at the goal of presenting a usable three-dimensional audio interface for the people concerned. However, the efforts on developing audio interfaces for blind users are increasing. The International Community for Auditory Display[1] (ICAD) will held a one day workshop on "Auditory Displays for the Blind" at their 9th meeting in Boston in July, 2003.

The subsequent section investigates which quality attributes can be defined to evaluate a virtual audio reality. Usability is defined as part of the overall system acceptance. The differences between real world systems and their virtual replacements in terms of usability testing are stated as well. Subsequently, a usability test plan is proposed as basis for further investigations. Concluding, an outlook on future work is stated. It provides development approaches with the goal to make the system applicable in the real world.

## 5.1 Evaluation

To be able to evaluate the proposed system it is necessary to define clear attributes in terms of quality and usability. Many different aspects need to be considered to be able to define the overall system acceptability. Besides marketing and time-to-market aspects, the practical issues in terms of usability from the user point-of-view define the quality of a software product. These practical issues are the subject of this investigation while neglecting the social and marketing aspects. Figure 5.1 shows usability as software attribute in the context of the overall system acceptance and states the main issues to be considered.

The ease of learning for novice users is an essential property of usability and follows the approach of the proposed system to introduce an intuitive interface for visually impaired people. Efficiency goes along with effectiveness and demands the possibility of fulfilling tasks within the system. It describes the accuracy and the completeness of the achieved goals in relation to the effort. This effort may be measured in time to complete the task, but may also be a quantitative measurement of how often the help system was used.

[1] http://www.icad.org

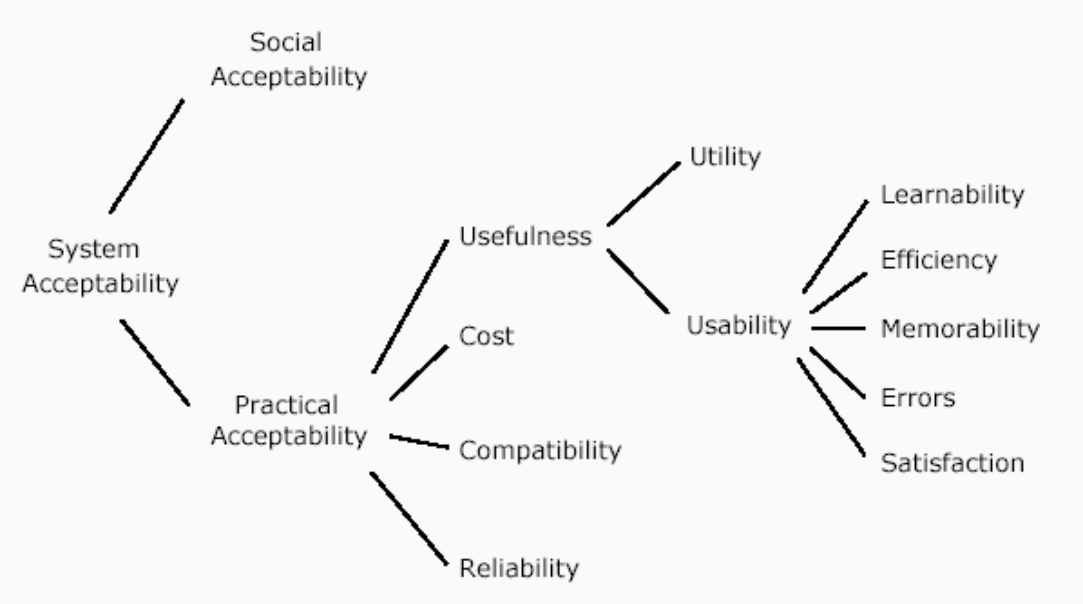


Figure 5.1: A model of attributes of system acceptability, from Figure 1 of [1]

Efficiency relates also to the frequency of errors occurring and in our case heavily depends on the accuracy of the audio rendering. Memorability is the ease of using the system intermittently for casual users. This comes again in favour of the achieved intuitiveness of the proposed interface. Finally, the overall satisfaction is stated in [1] as an attribute of usability. In comparison to the other attributes this attribute is hard to measure and highly subjective. It can only be determined using questionnaires or user interviews.

As stated in [36] the quality measures are different in real and virtual environments. The following section defines attributes for virtual audio reality to provide objective and subjective methods to determine the quality of VAR systems.

### 5.1.1 Defining Quality for Virtual Audio Reality

The definition of quality for virtual environments is crucial to be able to compare real world solutions with their virtual replacements. Especially in the field of audio, the simulation of the real world is always restricted by assumptions and approximations. These constraints result from limited processing power, the personalised auditory capabilities and cross-modal interaction. Just-noticeable-differences (JNBs) are defined to disregard irrelevant information and optimise the system's performance. This leads to a perception-based, plausible reproduction instead of a physics-based, authentic copy of the real world. Figure 5.2 illustrates the approach. This implies that for the definition of quality attributes the application needs to be clearly identified. In the case of the proposed system the goal is to develop a generic, interactive user interface for common computers with the capability to host earcons. These earcons should be acoustical repre-

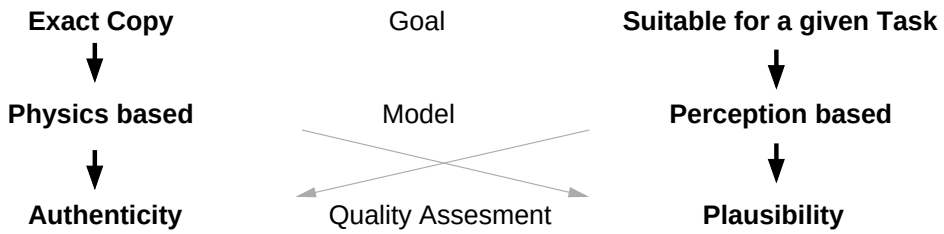


Figure 5.2: Authentic versus plausible reproduction

sentation of information, usually displayed graphically, providing the same functionality.

These considerations allow a classification of quality attributes: physically measurable properties and subjective, perception based properties. Table 5.1 summarises the identified physically measurable properties.

| Attribute | Description |
|---|---|
| Frequency resolution | The quality of the sound used in the system is determined by the maximum supported sampling rate. |
| Spatial resolution | Depends on the employed spatialisation algorithm. The just-noticeable-difference limits of the human auditory system need to be considered. |
| Scalability | Determined by the capabilities of the used hardware and algorithms. Restricts the flexibility and amount of content. |
| Dynamic behaviour | The dynamic behaviour of the system is determined by the following properties:<br>• System Latency (e.g. stop/play)<br>• Update rate (e.g. Headtracker, Joystick)<br>• Maximum velocities of sources and listeners<br>• Considered frequency effects (Doppler effect) |

Table 5.1: Physical Quality Attributes for VAR

The subjective attributes of quality are harder to determine because they result from user tests and subjective perception. However, many approaches were introduced to define an audio descriptive language to make results comparable. The nature of spatial sound perception was extensively studied in the field of concert hall acoustics. Sets of descriptive terms were introduced to qualify the perception (an overview is given in [37]). To be able to re-use these audio descriptive analyses for virtual environments two systematic problems have to be taken into account: 1) A VAR does usually not

control all modalities and the remaining modalities may influence the overall perception (cross-modal interaction). 2) Any dynamic change in the virtual scene caused by user interaction may not conform to the user's expectation. This is caused by insufficient modes of interaction (e.g. joysticks).

Looking at these two systematic problems for this special application of VAR, the following conclusions can be drawn: On the one hand, due to the lack of vision, the cross-modal interaction is not a significant problem with this user group. On the other hand, this lack also decreases the intuitiveness of movements conducted via feedback devices like joysticks. It was found that visually impaired people have very little experience with such feedback devices and are struggling to predict results from the interaction.

Table 5.2 summarises meaningful attributes for subjective quality assessment. These attributes are a subset selected from the attributes proposed in [37, 26, 36] and adapted for this application.

These descriptive terms can be used to conduct user tests when provided with according adjectives for the positive and negative end of the scale (e.g. well-defined $\leftrightarrow$ ill-defined). Concerning the analysis and representation of the collected data one must consider the relationship between preference rating of subjects and direct attribute rating. Therefore, a *preference mapping* was introduced in [37]. The method is based on a least square regression to determine the multivariate calibration of the descriptive terms.

The proposed qualifiers for an objective and subjective evaluation of virtual audio reality are only dealing with the simulation quality itself. Applications employing VAR like the desktop replacement introduced in chapter 4 need to be designed and evaluated separately. Usability engineering for user applications include the following tasks:

- Content design
- Goal-orientated interaction design
- Iterative Design
  - Prototyping
  - Evaluation
- Follow-up studies

Designing the content includes the acoustical representation of information and needs to answer questions about the user expectations of various sounds as well as the assigning of the proper level of attention for content. Interaction design determines the desired possible actions and their results for the achieved goals. Subsequently, the interface is developed iteratively using prototypes and usability testing methods.

| Attribute | Description |
|---|---|
| Sense of space | To perceive the enclosing environment, the attribute is essential for the sense of presence in a VAR. |
| Reverberation | This attribute does not necessarily correlate with the sense of space. It is also an important cue to distinguish between environments. |
| Location accuracy | Due to the personal differences in the localisation process, this is also a subjective attribute. It depends on the used simulation method, but also on provided features to resolve ambiguities (head movements). |
| Segregation, Confusion | To be able to segregate between sound sources is one of the main advantages of VAR over conventional, sequential audio interfaces. |
| Sense of distance | The sense of distance depends not only on the implemented algorithm, but also on the enclosing room. |
| Externalisation | Especially when three-dimensional audio rendering is produced via headphones, the in-head localisation of sound sources is a significant problem. |
| Clarity | Specifies how well the content of the sound sources can be perceived. |
| Dynamic accuracy | The dynamic properties relate to their corresponding physical properties (latency, update rate, resolution), but might effect users subjectively in a different way.<br>• Responsiveness (audible feedback on actions)<br>• Smoothness (continuous changes)<br>• Steadiness (e.g. of position during change) |
| Naturalness | Determining the plausibility of the simulation |

Table 5.2: Subjective Quality Attributes for VAR

### 5.1.2 Proposed Usability Test

A prototype of the system presented was built during the work on this thesis, but no evaluation tests were conducted so far. The usability test proposed in this section is intended to be a guideline for further investigations. It is important to carry out extensive usability tests with this system because of the very specific user requirements. Comparatively little is known about interface design for visually impaired people using virtual environments. The evaluation of this system is essential to prove the approach of employing VAR as a user interface for the people concerned.

In the first stage of usability testing a test plan needs to be developed. The subsequent sections resemble a simplified test plan according to [38] in order to provide a basis for a real usability test.

#### Purpose

The purpose of the usability test is to improve the quality of the presented VAR system. It has to be investigated whether VAR is capable of being used as human-computer interface for visually impaired users. To be able to exploit the theoretical improvements over sequential techniques the perceived accuracy and plausibility have to be determined.

#### Problem Statement

The goal of this test is to qualify the proposed VAR system according to the descriptive terms stated in section 5.1.1. The result should reflect the overall performance of the system and therefore its capabilities as an interface.

The quality attributes should be assessed in two stages: 1) The perception in statical simulation and 2) Behaviour of the system during listener movements.

#### User Profile

Special abilities and disabilities of the target group are investigated in section 2.2. For this usability test the target group can be classified into:

- *Novice users*
  Visually impaired people who have no pre-knowledge about computers. This group mainly tests the learnability of the system.
- *Casual users*
  Subjects with experience of computers and common assistive technology. They might represent the typical end-user. They have access to computers, but do not

use them in professional life. The user group uses the system to perform the most common tasks like text processing or obtaining information from the Internet.

- *Experts*
  People who either use computers in their professional life or use them regularly at home. They are familiar with common assistive technologies and have extensive background on using it effectively.

While the first two groups test the system for intuitiveness and learnability, the latter clearly addresses effectiveness. It is desirable to have at least three test participants for each user group.

### Test Design

The prototype has to be set up in a quiet room where only the test participant and the test facilitator are present. After stating the test purpose and process the usage of the system is explained to the test user. The test participants have to judge over the system according to a fixed set of attributes and adjectives.

The first test concerns the static properties. No interaction is carried out. The test user assesses the quality attributes while one single sound source is played in the virtual environment. Subsequently, a second source is played and the whole set of attributes is assessed again. A possible set for this test includes:

| Negative-end | Attribute | Positive end |
|---|---|---|
| ill-defined | Sense of Space | well-defined |
| dry | Reverberation | reverberant |
| ambiguous | Localisation Accuracy | clear |
| ambiguous | Distance | clear |
| artificial | Naturalness | natural |
| confusing | Segregation of sources | clear |

Dynamic attributes can be assessed by introducing user interaction. The task is to localise a sound source by moving the listening point as close as possible to the source. The test will be conducted with one source and two simultaneously playing sources. Quality attributes to be assessed are:

| Negative-end | Attribute | Positive end |
|---|---|---|
| slow | Responsiveness | fast |
| non-continuous | Smoothness | continuous |
| changing | Steadiness | stable |

Despite these qualifiers the test produces a lot of bottom-line data: The time to accom-

plish each task has to be measured, the way users took to the source, the accuracy of localisation and the usage of head movements and body rotation.

The results of such a usability test may indicate strengths and weaknesses of the proposed system providing the basis for any future work.

## 5.2 Future Work

There is still a long way to go until a system like this is applicable as a user interface in the real world. At the time being, it is only the approach to use VAR as human-computer interface for visually impaired users. Nevertheless, the results are encouraging. The prerequisite for any further development is a usability test, carried out to determine the performance of the proposed system.

The key to a high quality VAR is the audio rendering. The proposed spatialisation algorithms have still room for improvements. The Ambisonic algorithm uses very simple HRTFs at the moment. Research has proven that shorter HRTFs, modified by windowing techniques, improve the sound source localisation [23]. Additional distance cues may support users to build up the right cognitive model of the virtual scene presented. Just-noticeable difference (JNB) levels can reduce the necessary number of computed early reflections when considering hearing thresholds. Furthermore, the implementation of the convolution with the HRTFs is faster when using an FFT algorithm. The increasing power of DSPs and the stated systematic improvements might allow to introduce more accurate room acoustic algorithms. Furthermore, it increases the amount of possible content in a VAR.

With the increasing capabilities of customary PCs and sound cards it will also become possible to do all calculations at the host and no longer on a DSP. This would drastically decrease the costs of the system and makes it even more available for the people concerned. Real-time computer music software like Pd by Miller Puckette[2] already provides DSP like capabilities which could be used to compute the simulation.

No matter how the underlying signal processing is implemented, the simulation must be designed in a very flexible way. A higher number of possible sound sources and virtual rooms of more complex geometry increase the possible field of applications. A lot of different technologies are available for accessing documents via audio. Examples include the DAISY[3] project or the VoiceXML[4] standard. They might very well be collaborating with the proposed system to enrich the content.

Additionally, the integration of the VAR into the underlying operating system must be

[2] `http://crca.ucsd.edu/~msp/`

[3] `http://www.daisy.org`

[4] Voice eXtendable Markup Language, `http://www.voicexml.org`

strengthened. Similar to the JAWS screen-reader software, a VAR-based user interface needs to collaborate closely with the underlying operating system.

The prototype, as practical result of this thesis, proved that VAR with accurate positioning of sound sources is possible and capable of being used as user interface for visually impaired users. User interface design needs extensive usability engineering. As stated in this thesis, the special abilities and disabilities of the target group differ significantly from the average user. Therefore, every design process needs to be evaluated by usability testing methods to ensure a user-centred design.

# Appendix A

# ControlHPI

| Storage class | Identifier | Description | | |
|---|---|---|---|---|
| *General* | | | | |
| | | bits | Name | Comments |
| uint | Control | 1,0 | Algorithm control | 00 - Normal operation<br>01 - Direct sound Ambisonic<br>11 - Straight through, no algorithms |
| uint | SampleRate | Sampling rate [samples/sec] | | |
| *Source Control* | | | | |
| uint | SourceControl | On/Off switches of the sources 0-14bit | | |
| float | SourcePosition | Source Position [source#] [x,y] | | |
| uint | SourceGain | Source Gain (0x8000 = gain 1) | | |
| uint | SourceBufferPointer | Source Buffer Pointers [source#][p1,p2], p1-p2 is DSP Area, p2-p1 is Host Area, refer to section 3.5.1 | | |
| *User Control* | | | | |
| float | UserPosition | User Position [m,m] | | |
| uint | UserHeadOrientation | User Head Orientation [0..360] | | |
| uint | UserOrientation | User Orientation [0..360] | | |

| Storage class | Identifier | Description |
|---|---|---|
| *Room Control* | | |
| uint | Room | Room size [m,m] |
| int | EarlyWallDamping | Wall damping for early reflections [dB] |
| uint | EarlyPredelay | Predelay of early reflections [ms] |
| uint | EarlyCutoff | Cutoff frequency of early reflections [Hz] |
| uint | ReverbTime | Reverberation time [ms] |
| uint | ReverbCutoff | Cutoff frequency for reverberation [Hz] |
| uint | ReverbSigma | Delay line lengths (12) for reverberation algorithm [ms] |
| float | ReverbFBMatrix | Feedback matrix for reverberation algorithm |
| float | MixDirEarly | Mixing directive out = dir + MixDirEarly*Early |
| float | MixDirReverb | Mixing directive out = dir + MixDirReverb*Reverb |
| *Profiling* | | |
| uint | CPULoad | Ratio of interrupt service routine processing time to frame length [%] |

# Appendix B

# Class Diagram AISound3D

# Bibliography


[1] J. Nielsen, *Usability Engineering*, Academic Press, London, 1993, ISBN 0125184050.

[2] C. Frauenberger, "Three-dimensional Audio Interfaces for the Blind," Study project report, Institute of Communications and Wave Propagation, Graz University of Technology, April 2002, `http://spsc.inw.tugraz.at/3DAIB`.

[3] United Nations World Health Organization, "Prevention of blindness," `http://www.who.int/pbd/pbl/pbl_home.htm`, 2002.

[4] Visualization Graphic and Usability Center (GVU), "10th WWW User Survey, Disability," `http://www.gvu.gatech.edu/user_surveys/survey-1998-10/graphs/general/q1%2.htm`, 1998.

[5] Freedom Scientific, "Company Homepage," `http://www.freedomscientific.com/`, 2002.

[6] S. Lewis B. Edwards, "The Use of Technology in Programs for Students with Visual Impairments in Florida," *Journal of Visual Impairment & Blindness*, pp. 302–311, May 1998.

[7] J. Blauert, *Räumliches Hören*, S.Hirzel Verlag Stuttgart, 1974.

[8] G. Grinstein S. Smith, R. D. Bergeron, "Stereophonic and Surface Sound Generation for Explaratory Data Analysis," in *Proc. ACM Conference of Computer Human Interactions*. April 1–5 1990, pp. 125–132, ACM Press, Seatle, Washington, USA.

[9] A. D. N. Edwards, "The Design of Auditory Interfaces for Visually Disabled Users," in *Proc. ACM Conference of Computer Human Interactions*. May 15–19 1988, pp. 83–88, ACM Press, Washington, D. C., USA.

[10] P. Ghesquiere et.al., "The Significance of Auditory Study to University Students who are Blind," *Journal of Visual Impairment & Blindness*, pp. 40–45, January 1999.

[11] D. M. Lane B. N. Walker, "Psychophysical Scaling of Sonification Mappings: A Comparison of Visually Impaired and Sighted User," in *ICAD Proceedings*. ICAD: International Conference on Auditory Display, July–August 2001, Espoo, Finland.

[12] A. Cram D. P. Inman, K. Loge, "Teaching Orientation and Mobility Skills to Blind Children using Computer Generated 3-D Sound Environments," in *ICAD Proceedings*. ICAD: International Conference on Auditory Display, April 2–5 2000, Espoo, Finland.

[13] S. Leitner, "Aufnahme- und Wiedergabesysteme zur Berechnung eines dynamisch modifizierbaren binauralen Signalpaares," M.S. thesis, Institute of Communications and Wave Propagation, Institute of Electronic Music and Acoustics, Graz University of Technology, 2000.

[14] P. Minnaar et.al, "The Importance of Head Movements for Binaural Room Synthesis," in *ICAD Proceedings*. ICAD: International Conference on Auditory Display, July–August 2001, Espoo, Finland.

[15] B. Arons, "A Review of the Cocktail Party Effect," *Journal of the American Voice I/O Society*, vol. 12, pp. 35–50, 1992.

[16] R. Want E. D. Mynatt, M. Back, "Designing Audio Aura," in *Proc. ACM Conference of Computer Human Interactions*. 1998, pp. 566–573, ACM Press, Los Angeles, California, USA.

[17] R. M. Greenberg M. M. Blattner, D. A. Sumikawa, "Earcons and Icons: Their Structure and Common Design Principles," *Human-Computer Interaction*, vol. 4, no. 1, pp. 11–44, 1989.

[18] S. A. Brewster, *Providing a structured method for integrating non-speech audio into human-computer interfaces*, Ph.D. thesis, University of York, UK, 1994, `http://www.dcs.gla.ac.uk/~stephen/papers/Brewster_thesis.pdf`.

[19] A. I. Tew D. T. Murphy, M. C. Kelly, "3D Audio in the $21^{st}$ Century," in *Proc. $8^{th}$ International Conference, Computers Helping People with Special Needs, ICCHP 2002*, July 2002, pp. 562–564, Linz, Austria.

[20] K. Martin B. Gardner, "HRTF Measurements of a KEMAR Dummy-Head Microphone," Tech. Rep., MIT Media Lab Perceptual Computing, 1994.

[21] E. Dooijes J. Mulder, "Spatial Audio in Graphical Applications," Technical report CS-R9434, CWI-Amsterdam, 1991.

[22] J. S. Bamford, "An Analysis of Ambisonic Sound Systems of First and Second Order," M.S. thesis, University of Waterloo, `http://audiolab.uwaterloo.ca/~jeffb/thesis/thesis.html`, 1995.

[23] Piotr Majdak M. Noisternig, "A Head Position Related Binaural Sound Reproduction System," M.S. thesis, Institute of Electronic Music and Acoustics, Graz University of Music and Arts, 2002.

[24] A. Sontacchi et.al, "An Objektive Model of Localisation in Binaural Sound Reproduction Systems," in *AES: Proceedings*. AES: Audio Engineering Society, June 2002, St.Petersburg, Russia.

[25] A. Helmuth L. Cremer, Mueller, *Die wissenschaftlichen Grundlagen der Raumakustik*, vol. 1, Hirzel Verlag Stuttgart, 2nd edition, 1978.

[26] H. Järveläinen T. Lokki, "Subjective Evaluation of Auralization of Physics-Based Room Acoustics Modelling," in *ICAD Proceedings*. ICAD: International Conference on Auditory Display, July–August 2001, Espoo, Finland.

[27] B. F. Logan M. R. Schröder, "Colorless Artificial Reverberation," *Journal of the Audio Engineering Society*, vol. 9, no. 3, 1961.

[28] J. M. Jot, "Efficient Models for Reverberation and Distance Rendering in Computer Music an Virtual Audio Reality," in *ICMC Proceedings*. ICMC: International Computer Music Conference, September 1997, Thessaloniki, Greece.

[29] J. M. Jot L. Dahl, "A Reverberator Based on Absorbant All-Pass Filters," in *DAFx-00: Proceedings*. DAFx: COST G-6 Conference on Digital Audio Effects, December 2000, Verona, Italy.

[30] J. M. Jot, "An Analysis/Synthesis Approach to Real-Time Artificial Reverberation," in *ICASSP-92 Proceedings*, 1992.

[31] F. A. Beltrán et. al., "Matlab Implemetation of Reverberation Algorithms," in *DAFx-00: Proceedings*. DAFx: COST G-6 Conference on Digital Audio Effects, December 9-11 1999, Trondheim, Norway.

[32] C. Frauenberger, "Software documentation, 3DAIB DSP program," online: `http://spsc.inw.tugraz.at/3daib`, 2002.

[33] R. Kazman L. Bass, P. Clements, *Software Architecture in Practice*, The SEI Series in Software Engineering. AddisonWesley Longman, Inc, 1998, ISBN 0-201-19930-0.

[34] N. Tsingos, "A Versatile Software Architecture for Virtual Audio Simulation," in *ICAD Proceedings*. ICAD: International Conference on Auditory Display, July–August 2001, Espoo, Finland.

[35] Texas Instruments, *TMS320C6201/6701 Evaluation Module Technical Reference - SPRU305*, 2002, Part of the online documentation of the Code Composer Studio Software.

[36] R. S. Pelligrini, "Quality Assessment of Auditory Virtual Environments," in *ICAD Proceedings*. ICAD: International Conference on Auditory Display, July–August 2001, Espoo, Finland.

[37] K. Koivuniemi N. Zacharov, "Audio Descriptive Analysis & Mapping of Spatial Sound Displays," in *ICAD Proceedings*. ICAD: International Conference on Auditory Display, July–August 2001, Espoo, Finland.

[38] K. Andrews, "Human-Computer Interaction," `http://courses.iicm.edu/hci`, 2002, Lecture Notes.