FACHHOCHSCHULE WEDEL
UNIVERSITY OF APPLIED SCIENCES
DIPLOMA THESIS

Submitted to the department of
Media Information Science

Integration of Multichannel Sound in Virtual Environments

Submitted from : Severin Sönke Todt
Alter Hof 2
25451 Quickborn

Compiled : 9th semester

Turned in : 2002 – 02 – 12

Certified by : Prof. Dr. Ülzmann
Fachhochschule Wedel
University of Applied Sciences
Feldstraße 143
22880 Wedel
Germany

Prof. Dr. Kolb
Fachhochschule Wedel
University of Applied Sciences
Feldstraße 143
22880 Wedel
Germany

In memory of my loved grandfather

Kurt W. Mewes

Integration of Multichannel Sound in

Virtual Environments

by

Severin S. Todt

Submitted to the Media Information Science Department,
Fachhochschule Wedel – University of Computer Sciences,
on February 12, 2002

Abstract

In this work, a real – time 3-D sound API is developed based on a standard hardware platform using a loudspeaker array of eight loudspeakers and Ambisonics for audio reproduction in the context of virtual reality.

The basic human sound perception is introduced to identify relevant factors on 3-D sound perception. The natural sound system is further investigated within a system analysis of human listening

The cognition flows into the compilation of minimum requirements on real – time 3-D sound APIs which are in the following used to appraise related API implementations and justify the development of a new API approach.

For implementation purposes and audio reproduction a hardware system is composed that is integrated in the available infrastructure of the virtual reality laboratory at the Fachhochschule Wedel serving as an integrated virtual reality system housing virtual reality applications as well as real – time 3-D sound processing tasks.

The real – time 3-D sound API's kernel functionality is implemented and tested based on the system being set up proving the parallel processing ability of graphical and 3-D sound tasks in real – time using one machine and the possibility of easy integration of real – time 3-D sound functionality in virtual reality applications.

Acknowledgements

I like to thank Julia J. Lambeck for her understanding and patience. Without her motivation and support this work would not be finished yet.
Thanks to my colleagues, Andreas Kolb, Henry Kleta, Katrin Fitz, Christine Nickel and Daniel Bekowies for their understanding that work has to stand back sometimes.
Concentrating on this work was only possible through the open minded attitude of Prof. Dr. Harms and the confidence of Prof. Dr. Ülzmann.
Without the tolerance, the staying power and regard of my parents, especially around Christmas time, the completion of this thesis would not have been possible.

Severin S. Todt

Wedel, February 2002

Table of Contents

Table of Figures

1. Introduction

Sounds are present in our everyday world. Sound as we hear it comes from all around us delivering information about our environment and specific objects in it.
Sound positioned in space, often called spatialised sound, 3-D sound or 3-D-Audio, utilises the human ability to distinguish sounds by their location in space, pitch and tone to deliver the information hold by the sound emitting object.
3-D sounds broaden the pure information about the object's nature by adding positional information to the sound.
This aspect can help to increase the sense of presence of virtual reality applications and to improve their immersiveness.
For a long time the capabilities of sound, especially 3-D sound have been ignored.
Considering the proven fact that adding sound to an interactive application is able to improve the overall interactivity and even the perceived image quality[1], it is highly desirable to include 3-D sound in virtual reality applications.

1.1. Motivation

The motivation for this thesis originates from the great benefits of 3-D sound integration in graphic based applications.
In western cultures the visual perception is more distinct than all of the other senses – in other cultures like in the African culture the auditory sense play a bigger role. [1]
The cultural differences give reason for the fact that the emphasis of most applications in the field of virtual reality is about the integration and presentation of visual aspects to gain the impression of realism.
An application can only appear as realistic as the amount and type of senses integrated in the application allow it to be. The more senses included in a virtual reality application the higher the immersiveness will be. The greater the diversity of senses accosted by the application the more realistic the generated impression will appear.

---

[1] cp.: Hennig, Alexander, Die andere Wirklichkeit, München, Addison Wesley, 1996

As a matter of course the sense of hearing should be part of any virtual reality application in order to obtain the highest immersiveness possible.
3-D sound can contribute to the sense of immersivity in a 3-D environment through the enrichment of feedback channels which can be helpful redundant to a visual cue or can provide feedback for actions and situations that are out of the listener's field of view.[2] Several studies have proven that the integration of high quality sound in graphic based application has the capability to improve the subjective perception of the image quality. Sound can complete the felt visual impression by giving additional feedback and information regarding the pictured scene. It can be emitted when two objects of the actual scene collide, making information about the object's material and the vigorousness of the collision available. Invisible objects can easily be pursued or audibly observed through the use of spatialised sound. With sound integration the recognition of objects in a scene can be improved[3] and the time delay between object appearance and object perception can be shortened.[4]
With the benefits of sound integration, sound can be used to improve the quality and ease of interaction within applications.

Although sound is used a lot in the entertainment industry like in film business and computer game programming there is a dearth of applications using sound effects computed in real time. Most of the available programs use sound only as environmental coloration, playing sounds of the scratch without computing any effects concerning the user input in real time. Mostly the reproduction is limited to playing pre-spatialised sounds or ambient sounds, sometimes optimised for multichannel reproduction but not for real-time 3-D sound reproduction.

---

[2] cp. :Begault, Durand R. : 3-D sound For Virtual Reality And Multimedia, Moffet Field, NASA Ames Research Center, Academics Press Professional, 1994, Reprint 2000, p.14

[3] cp.: Davis, Elisabeth T. a.o. : Can Audio Enhance Visual Perception and Performance in a Virtual Environment, Atlanta, Georgia Institute of Technology, 1999, p. 4

[4] cp.: Begault, Durand R. : Head-Up Auditory Displays for Traffic Collision Avoidance System Advisories, Moffet Field, NASA Ames Research Center, The Human Factors and Ergonomics Society Inc., 1993, p. 707

To ensure easy access to sound features, to simplify the integration of sound and to consider user input in any context for computation of corresponding sound effects in real - time, an application programming interface is desirable.
Through the use of an API more applications will originate, expediting the development of more sound related technologies. Programmers will be shielded from the operating system subroutines and sound device access processes, giving more developers approach to the wide field of sound, resulting in further research in the field of sound.

1.2. Related Research

By the time this thesis is written a number of research projects concerning the implementation of sound in applications have been implemented.
In the following three representative sample research projects regarding the integration of 3-D sound in virtual reality applications will be resumed before the scope of this thesis is presented.
All of these projects resulted in a working hard- and software system offering the capacity of delivering 3-D sound effects computed in real time. They all offer an API, making it possible to create situation fitted audio applications.
But all of the approaches differ in the capacity of operating resources needed for computation, the extendibility, the ease, the financial aids needed and the transportability to other platforms.

1.2.1. University of Aizu, Japan

At the University of Aizu research is concentrated on representing multimedia technologies in 3-D space. Out of this focus the project "Virtual Reality Sound" was raised in 1999.
The project resulted in a hardware audio set-up making 3-D sound reproduction over headphones or a loudspeaker array consisting of 14 loudspeakers available.
The headphone reproduction is driven by the "Acoustetron II" audio card – a high cost audio card former manufactured by Crystal River Engineering™, proposed for the computation of spatialised sound in real - time.

A special control system, manufactured by Pioneer Corporation™, the "Pioneer Sound Field Control System"[5] makes the 14 loudspeakers accessible. For other reproduction systems the interface is designed to be open for extensions.
On the software side a toolkit called "Sound Spatialisation Resource Management Framework" written in C++ is used to compose multimedia applications.
In addition to simple being an application composer it offers support for efficient resource management.
The sound spatialisation is performed in conformity with the VRML 97 standard.[6]

1.2.2. GMD – National Research Centre for Information Technology, Germany

Research projects aiming at the introduction of virtual reality technologies for applications concerning visualisation and mediation of digital content are housed in the Competence Centre Virtual Reality at the GMD[7].
Two main audio related projects have been progressed in the past.
Within the "CyberStage" project an eight channel 3-D sound reproduction system was established using an additional vibrating floor to provide haptic audio feedback.
Secondly a 3-D sound reproduction system was developed being able to serve 3-D sound for a great amount of listeners. Using loudspeakers this system is interlinked with a 140 meter wide projection wall for visual feedback.
Both systems run on a SGI™ Octane hardware platform and are controlled over a "Sound Server" software API, designed for operation on SGI™ hardware platforms.
The "Sound Server" is used to perform the syntheses, processing and spatialisation of sound signals. To enable great amount of educible virtual sound sources and adjust it to the actual situation, the type and amount of rendered audio effects is adjustable.
For high level programming an additional simplified API is provided which offers easy access to the sound features as well as to the graphic processing pipeline.

---

[5] cp.: http://wwwsv1.u-aizu.ac.jp/~mcohen/spatial-media/PSFC/welcome.html, 11.02.2002

[6] cp.: Herder, Jens : A Sound Spatialisation Resource Management Framework, Aizu, The University of Aizu - Spatial Media Group, 1999

[7] cp. http://www.gmd.de/, 11.02.2002

1.2.3. Georgia Institute of Technology, U.S.A.

Research projects concentrating on finding better ways to communicate information between people and between computers and people are dealt with at the Georgia Tech Computer Interest Group[8].

The major project NAVE – Non expensive Automatic Virtual Environment aims at a high performance low cost virtual reality hard- and software set-up.

Within this project an environment including three projection walls and a dedicated audio workstation, totally based on of the shelf PC based systems was implemented.

Only operating three standard PC based systems for graphical issues and an extra PC system for audio processing the system it is able to run the projection walls and implement 3-D sound applications.

With the audio workstation four loudspeakers for surround sound and a "Thunder Seat" – a low frequency vibrating seat - are driven.

Over an API, connecting sound properties to objects positioned in 3-D space is possible.

Taken into account the listener and object position as well as the speed, the Doppler shift and environmental sound effects are computed in real - time.

1.3. Scope of this Thesis

As seen in world-wide projects, like the ones introduced, there is great interest in designing an API, offering easy access to sound integration for real time 3-D sound applications.

A great number of projects have examined this problem creating variations of operative soft- and hardware implementations.

Nearly all of the approaches come with the same major shortcomings. The projects and shortcomings can be distinguished in three categories.

One category of projects like the "Virtual Reality Sound" project at the University of Aizu depend on special and high cost hardware. Other hardware set-ups are conceivable in principle but would reach in a reimplementation of the software toolkit.

---

[8] cp.: http://www.gvu.gatech.edu/gvu/, 11.02.2002

Secondly approaches like the GMD's "Cyber Stage" and "Sound Server" seem to be expandable but are inflexible and hard to match to other hardware set-ups.
To base the 3-D sound system on hardware with a high level of divulgation is a step in the right direction, although the hardware that is relied on, often is too expansive to be really wide spread as shown in the "Cyber Stage" project which relies on SGI™ hardware capacities. Choosing to use hardware with greater prevalence makes 3-D sound applications available for more consumers and developers.
To overcome the last burden less expensive hardware systems are taken into account for an even vaster propagation of 3-D sound systems and applications as seen in a basic approach at the "NAVE" project at Georgia Tech representative for a third category of projects.

For a breakthrough of real time 3-D sound systems and applications it is necessary to follow the path paved within the "NAVE" and "Cyber Stage" projects and to go further beyond the projects' idea.
To rely on affordable and wide spread hardware is a decision benefiting the propagation of real-time 3-D sound systems. There has to be support for the greatest variation of available hardware. An 3-D sound API must be free of any hardware specifications . It must be easy adjustable for consumers to fetch their used hardware configuration. It should be affordable and useable.

This thesis investigates the possibilities of 3-D sound integration in virtual reality applications. The investigation includes the set-up of a base audio workstation, composed of off the shelf hardware components.
Pursuing the designation of an hardware and operating system independent 3-D sound API for real-time audio processing, available APIs are examined. Based on the results of the examination a new API will be developed to suffice the demands of real-time 3-D sound.
Keeping in mind the primary goal of an independent API, the API will matched to the installed audio hardware and the operating system running on the audio workstation set up in a first step to proof the rightness of the API's rudiment.

1.4. Organisation of this Thesis

Having introduced the scope of this thesis, the physics of sound perception and sound reproduction will be introduced in the "Background" chapter.
The "Background" chapter first explains the human abilities as well as the human weaknesses in sound perception to understand the assumptions made through the implementation of the API and the underlying algorithms.
The different methods for sound reproduction are also shown and discussed in the second chapter.
With the background it is then possible to expose the requirements necessary for a 3-D sound API. The goals and minimum requirements of an software approach are retained in the third chapter. Related software approaches are examined and their usability as well as their mightiness are discussed. Properties of the related APIs making a new software approach desirable are argued.
For extensive understanding of the 3-D sound mechanisms the natural 3-D sound system is analysed within the fourth chapter.
To implement and operate a 3-D sound API a hardware set-up is needed which is composed and evaluated in chapter five making the 3-D sound engineering process described in chapter six possible.
A test application using the 3-D sound API is used to proof the workability of the developed API. The test results are discussed and the API's usability is rated in chapter seven.
This thesis closes with the "Conclusion and Future Work" chapter, summarising the work and giving an insight in future research projects.

2. Background

This chapter gives a short introduction to the main aspects of human sound perception. An insight in software and hardware methods for audio reproduction is given showing their advantages and disadvantages as well as approaches on how to integrate a audio system in a virtual reality system are evaluated.

2.1. Sound Perception

The field of human sound perception has three scientific bases, the physical, psychological and neurophysiological aspects of sound perception.[9]
Since auditory neurophysiology examines the neurological structures resulting in the experienced sound, which is beyond the scope of this thesis, only the psychological and physical aspects are considered here.

2.1.1. Physical Acoustic Perspective

Physical acoustics focuses on the accruement, and the propagation up to the perception at the listener's ear drums of sound.
Sound waves created by a vibrating object are emitted in all direction by omni-directional sound objects and in a particular direction by directional sound objects. During propagation the sound waves interact with objects positioned within the direction of propagation. Throughout all the environment that the listener and the sound emitting object are in, the sound wave is manipulated many times offering the listener information about the sound object itself and the environment.
Using two ears the listener is capable of retrieving more information through intensity- and time differences of sound arrival at the ears while receiving the sound wave.

A sound wave emitted by an object carries information by nature. Familiar sounds can help the listener to classify or identify an object.

---

[9] cp.: Kendall, Gary : A 3-D sound Primer,
http://www.northwestern.edu/musicschool/classes/3D/pages/3DsoundPrimer.html, 11.02.2002

Even sounds that the listener is not familiar with, carry information about the voluminous or size of the emitting object for example. Thus even unknown objects can be classified through the sounds they are emitting (e.g. did you ever know a siren to be good?[10])

Furhter information is added to the sound while propagating through the environment and during perception at the listeners ear drums.

On the way through the environment the sound wave is effected by the environment itself, decreasing the intensity of the sound with covered distance, as well as through interfering objects.

The intensity decrease of the sound with distance can be approximated for sound propagation in air at average humidity with the inverse square law.[11] Given a reference intensity and distance an omnidirectional sound source's intensity will fall almost exactly 6 dB for each subsequent doubling of distance from a source.[12] More specific analysis have shown, that the inverse square law is not the best method to simulate intensity decrease over distance but still is accurate enough. Not only the intensity itself, the overall impression of the arriving sound waves emitted by one source is more important for distance perception then just the intensity itself.[13] The distance of an object can be specified by a listener for a specific moment. For moving objects the distance can be determined several times at constant time intervals to gain information about it's speed. There is a more accurate effect serving information about an object's speed relative to the listener. The Doppler shift causes a change in frequency of the perceived sound wave. Through movement in direction of sound propagation the wave length is shortened or lengthened otherwise.[1]

Emitting into space, sound waves encounter objects by which they are reflected, diffracted or absorbed.

---

[10] cp.: Groening, Matt : The Call of the Simpsons, Twentieth Century Fox, 1990

[11] cp.: Hennig, Alexander, Die andere Wirklichkeit, München, Addison Wesley, 1996

[12] cp.: Begault, Durand R. : 3-D sound For Virtual Reality And Multimedia, Moffet Field, NASA Ames Research Center, Academics Press Professional, 1994, Reprint 2000, p.71

[13] cp.: Begault, Durand R. : Preferred Sound Intensity Increase For Sensation Of Half Distance, Moffet Field, AES 93rd Convention San Francisco, 1992

Reflecting a sound decreases a sound's intensity and changes it's direction. If a soundwave bends around an objects an acoustical shadow at the backside of the object appears.
The acoustical shadow spreads out over an area depending on the wavelength and the object's size. Within the shadow area the sound is not perceivable.
A sound being absorbed by an object looses all of it's intensity when encountering the object.
Throughout sound propagation the sound is varied several times, causing all of a sound's soundwaves perceived by the listener to create a rich acoustic mixture. The waves are arriving at the listener at different times from different directions.
Typically there is one straight-line path along which the initial waves of each event first reach the listener. This initial direct sound provides the latest compromised information about the direction of the sound event. The bigger part of soundwaves, called early reflections, reaches the listener after several reflections at the surfaces or objects in the room serving information about the room's characteristics and additional information about the location of objects. The part reaching the listener's ears last, called reverberations, gives the listener information about the size and structure of the room he is in.
Every listener perceives the arriving sounds in a different way since they are filtered directly by a listener's body specific acoustical filter before reaching the ear drum. The sounds are at last affected by the listeners torso, head, pinnae (outer ear) and ear canals. Manipulation caused by the listener's body can be described as a filtering function, called "Head related transfer function" (HRTF). The HRTF varies between users and has to be taken into account when reproducing 3-D sounds using two channels. A further explanation of the HRTF is beyond the thesis but will be taken into account in future researches.
With the help of the HRTF the listener is able to appoint the position of the sound's corresponding object.

For describing sound events, a coordinate system is used with the listener's head being centre of it. (Fig. 1)

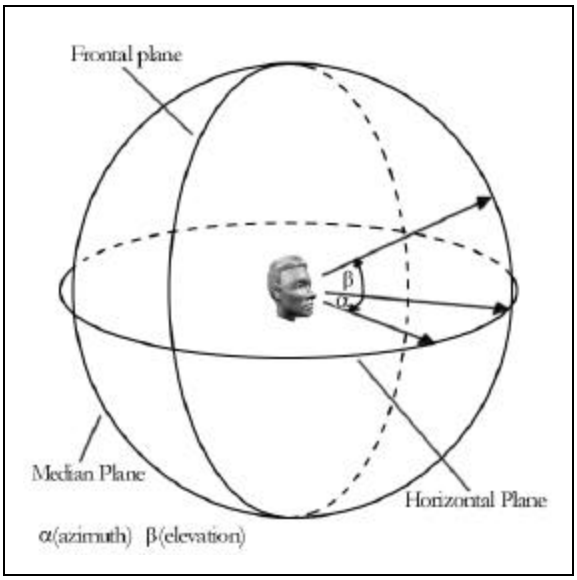

Figure 1 : Head related coordinate system

The coordinate system is described using three planes. The horizontal plane which is level with the listener's ears, the frontal or lateral plane which devides the listener's head vertically between the front and the back and the median plane which devides the listener's head vertically between the left and the right.

The position is then described as a vector expressed in terms of two angels. Azimuth, as orientation on the horizontal plane and elevation as orientation on the vertical plane with one scalar, the distance.[14]

An object's position is mainly derived from the time difference (internal time difference - ITD) and intensity difference (internal intensity difference – IID) between sound waves reaching the left and the right ear. The ear being closer to the sound source receives the sound waves earlier and with higher intensity then the other ear.

[14] cp.: Kendall, Gary : A 3-D sound Primer, http://www.northwestern.edu/musicschool/classes/3D/pages/3DsoundPrimer.html, 11.02.2002

2.1.2. Psychological Acoustic Perspective

Psychological acoustics focuses on the relationship between the acoustic waves at the eardrums and the perception of spatial imagery of the listener.

In several researches it has been shown that only the part arriving at the ears first is determining for the listener interpreted position of the object

This effect is called the "Precedence Effect" [15], since following sound information is ignored.

For Objects positioned at the median plane the emitted soundwaves reach both of the listeners ears at the same time, making determination whether the sound is coming form above, below form the front or from the back possible. (Fig. 2)

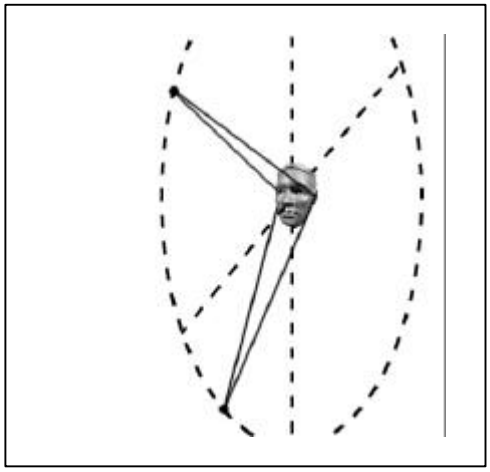

Figure 2 : Location error at the median plane

Additionally, sounds positioned on a circle around the axis pointing through the ears of the listener cause identical IIDs and ITDs making only imprecise positioning to the left or the right possible, forming the "Cone of confusion".[16] (Fig. 3)

---

[15] cp.: Pulkki, Ville, Spatial Sound Generation and Perception by Amplitude Panning Techniques, Helsinki, Laboratory of Acoustics an Audio Signal Processing - Helsinki University of Technology, Espoo 2001, Report 62, 2001, p.8

[16] cp.: Begault, Durand R. : 3-D sound For Virtual Reality And Multimedia, Moffet Field, NASA Ames Research Center, Academics Press Professional, 1994, Reprint 2000, p.41

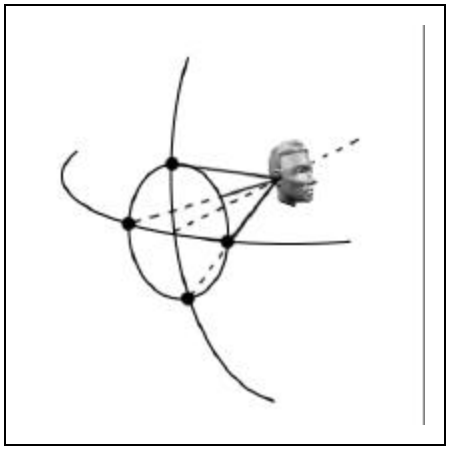

Figure 3 : The Cone of Confusion

Naturally the listener overcomes the positioning weaknesses through turning the head. With changing the heads position the ITD and IID change and make an accurate positioning possible.[17]

The resolution with which a sound object can be positioned in space is pretty low, concerning the human abilities. The highest resolution is evident in the horizontal dimension, especially in front of the listener where the minimum audible angle is about two degrees. That angle increases to near ten degrees at the sides and narrows to near six degrees in the rear. By comparison, the resolution in the vertical dimension is even lower. The vertical minimum audible angle in front is near nine degrees and steadily increases until overhead when it reaches 22 degrees.[18]

Having a visual counterpart of the acoustic impression the localisation improves in that way as front back reversals are eliminated.

Reproducing an artificial sound, the position the sound is being played back from can deviate from the corresponding object. Studies have shown that even a tolerance of eleven degrees between the visual and acoustical position is only noticeable by experts.[19]

---

[17] cp.: Tonnesen, Cindy; Steinmetz, Joe : 3-D sound Synthesis, Washington, The Encyclopedia of Virtual Environments, http://www.hitl.washington.edu/scivw/EVE/I.B.1.3DSoundSynthesis.html, 1993

[18] cp.: Kendall, Gary : A 3-D sound Primer, http://www.northwestern.edu/musicschool/classes/3D/pages/3DsoundPrimer.html, 11.02.2002

[19] cp.: Begault, Durand R. : Auditory And Non-Auditory Factors That Potentially Influence Virtual Acoustic Imagery, Moffet Field, Human Factors and Technology Devision - NASA Amees Research Center, 1999, p.8

The human ability to determine the distance of a sound object is even less precise and varies between individuals. For distance perception it is proven that the listener has to be familiar with the sound to determine the object's distance, otherwise the listener would not have any reference sound to compare the perceived sound's intensity and coloration with.
For this reason the sound object's visual counterpart is taken for distance determination if possible.

2.2. Spatial Sound Reproduction

Investigating different sound reproduction possibilities it is necessary to discern the available algorithms as well as the available hardware set-ups and their implementation. To fully understand audio reproduction concerns the main audio reproduction techniques are introduced assigning the abilities of every possibility regarding the reproduction of 3-D sound.
Audio reproduction is possible using loudspeakers or headphones. These two approaches are compared, showing possible operational areas, the advantages and disadvantages of each, leading to adaptable software algorithms for 3-D sound reproduction for both of them.

2.2.1. Spatial Auditory Displays

Monophonic sound reproduction describes the audio presentation using one channel. A monophonic sound system is incapable of reproducing the spatial characteristics of a sound. Since all colorations of sounds, including reverberation are reproduced from the location of the one channel only unnatural impressions can be generated.[20]

Two-channel stereo sound systems are far superior, enabling the reproduction of sound images that are spatially distributed between two channels - implemented using headphones or loudspeakers. When using loudspeakers they are normally set-up in a way that listener and the two speakers are forming a equilateral triangle.

---

[20] cp.: Gardner, William G.: 3-D Audio Using Loudspeakers, Boston, Massachusetts Institute of Technology, Kluwer Academic Publishers, Norwell, Massachusetts, 1998, pp.1-6

Playing the same sound over the two channels, it is then possible to generate a virtual source between these two channels. Through adjusting the amplitude the virtual source will be positioned further to the speaker with the higher intensity until it reaches the channel's location.

The loudspeaker reproduction supposes the listener staying in a small area between the speakers, called the stereophonic area, to guaranty the full effect.[21] (Fig.4)

With headphone reproduction the listener is freed of this restriction.

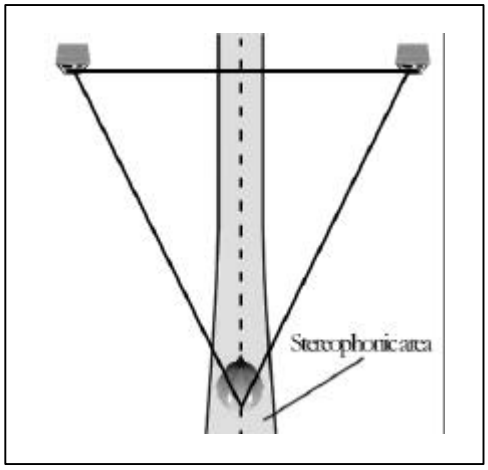

Figure 4: Stereo audio reproduction

The capabilities of stereo systems can be augmented by adding additional speakers to the sides or rear of the listener. (Fig.5)

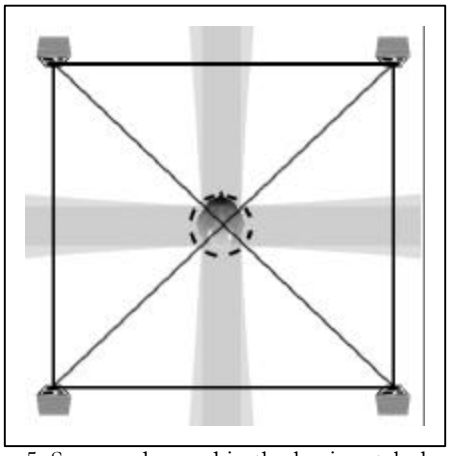

Figure 5: Surround sound in the horizontal plane using four speakers

[21] cp.: Dickreiter, Michael: Handbuch der Tonstudiotechnik, München, Saur, K.G, 1997, p.126

In these set-ups each channel receives a monophonic mixed and amplitude adjusted signal of the same sound source from the mixing console that all of the channels are connected to.[22]

The resulting surround systems are generally capable to reproduce sound images anywhere in the horizontal plane surrounding the listener. Actual surround sound systems are wide spread in the area of home cinema applications and in movie theatres. These surround sound system like implemented by Dolby Laboratories™ (Early 70s Dolby Stereo™, 1982 Dolby Surround™, 1987 Dolby ProLogic™, 1992 Dolby Surround Digital™, 1998 Dolby Digital Surround EX™) or Sony (1993 SDDS) are all recommendations for speaker positioning and describing coding conventions for audio data transmission. They are based on a "N.1" channel layout working with N full bandwith channels used for sound localisation and one limited bandwith channel for low frequency. The latest implementation from Dolby Laboratories™, Dolby Digital Surround EX™ rests upon a 6.1 layout. There three channels in the front (left, center, right) and three in the back, the so called surround channels (surround left, surround center, surround right). These are used to position sounds being played back[23]. A loudspeaker playing back not locatable low-frequency sounds can be positioned anywhere. Although the 6.1 layout offers good possibilities to position a sound being played back, the sound can still be only positioned somewhere in the horizontal plane not being able to produce a feeling of elevation. (Fig.6)

---

[22] cp.: Jot, Jean-Marc, Synthesising Three-Dimensional Sound Scenes in Audio or Multimedia Production and Interactive Human – Computer Interfaces, Ircam - Centre Georges-Pompidou, 5th International Conference, Interface to Real & Virtual Worlds, Montpellier, 1996

[23] cp.: Faust, Simon: 3-D Akustik mit Lautsprechern, Aachen, RWTH Aachen Virtual Reality and 3D Visualization, 2000, pp.5-6

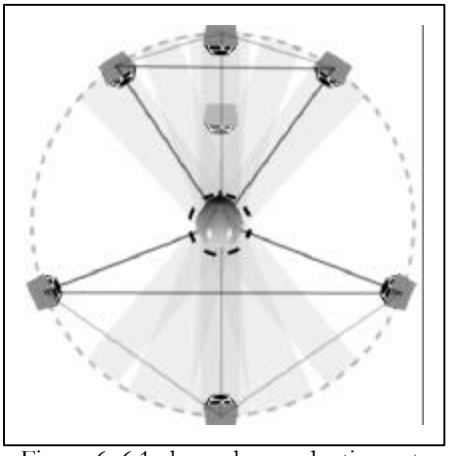

Figure 6: 6.1 channel reproduction set-up

A spatial auditory display instead is a system capable of reproducing sound images positioned arbitrarily around a listener in 3-D space. When using loudspeakers, a great number of transducers is positioned completely surrounding the listener to reproduce the intended position. Typically the loudspeakers are then positioned forming a cube using eight channels, one for each corner of the cube or positioned as two hexagons on top each other built of twelve loudspeakers. Other loudspeaker set-ups like forming a sphere out of a lot more loudspeakers is possible but difficult to build and connected to higher costs.

For positioning of sound in 3-D space over headphones an algorithm for reproducing the acoustical signals at the listener's ears that would occur in the natural listening situation is used. This method, called binaural audio simulates the HRTF and is further explained in the next section.

2.2.2. Software Techniques for 3-D sound generation

There are three categories of software approaches for 3-D sound generation. They especially differ in the necessary CPU power and the amount of hardware they rely on. All of the approaches have the same audio rendering pipeline in common.

The envisaged techniques all start with a original monophonic sound.

This sound is enriched through audio effects occurring due to the simulated environment it is positioned in and the physical aspects of sound propagation, e.g. Doppler shift, reflections and diffraction. This effected sound is then positioned in 3-D space using a specific technique.(Fig. 7)

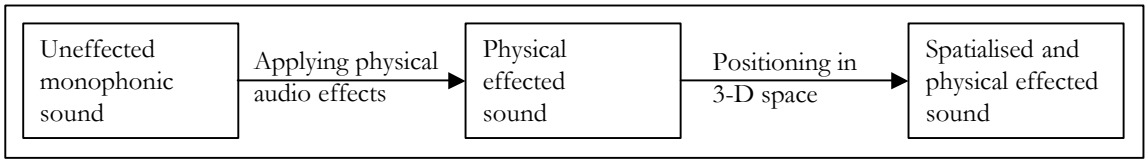

Figure 7 : Sound rendering pipeline

With the binaural audio reproduction algorithm a technique is described requiring maximum CPU power but relying on the fewest hardware requirements. The binaural reproduction algorithm uses a given HRTF to synthesise a 3-D sound imagery.
HRTFs can be measured using special microphone techniques placed in the ear canals of a listener. Playing back a predefined sound from approximately 100 different positions around the listener, the HRTF of one listener can be destined.
This HRTF can then be used to simulate the effects caused by the torso, head, shoulder an the outer ear, as described in the "Introduction" chapter, when reproducing the sound over headphones. When reproducing sound using headphones, applying HRTF to spatialise sound, is the only possibility to create the 3-D impression since the headphones shield the inner ear from any other influences which would be helpful for spatialisiation under natural conditions.
A measured HRTF could be used for other listeners but would result in pour localisation abilities.
To overcome the burden of measuring individual HRTF for each listener, averaged HRTF can be used, still causing errors in localisation but with lower influence.
To improve the listener's ability to perceive the sound at the position that it was supposed to originate from it is necessary to include head tracking for a precise capturing of the listener's look-at point.
This has to be done, since head rotation is used for sounds to be localised within the cone of confusion or on the median plane.[24]

---

[24] cp.: Gardner, William G.: 3-D Audio Using Loudspeakers, Boston, Massachusetts Institute of Technology, Kluwer Academic Publishers, Norwell, Massachusetts, 1998, pp. 11-12

When applying this technique to a 3-D sound reproduction system, not only the physical effects have to be applied to the sound, in addition the HRTF has to be applied to it, resulting in higher CPU load.

Applying the binaural technique to a stereo set-up of two loudspeakers is also possible. Then sound is played back using two loudspeakers positioned as usually in normal stereo reproduction. For this approach to work the listeners position has to be known in advance to apply the right variation of HRTF. A change in position during audio reproduction would be possible with a positional tracking delivering the actual location. As mentioned above there is only a little space of optimal sound impression that the listener has to stay in, limiting the listener's movement.
Additionally reproducing sound using two loudspeakers result in cross talk, when the emitted soundwaves addressed to the left ear reach the other ear too, causing unpredictable influence on 3-D sound location.
This effect can be normalised using cross talk cancellation,[25] resulting in further CPU power requirements.

Using an array of loudspeakers other approaches can be used to produce a 3-D sound imagery.
Based on the stereo sound reproduction where a virtual sound source is generated through adjusting the amplitudes of the used two channels, there can three loudspeakers positioned in 3-D space be used to generate a virtual sound source. Out of a set of loudspeakers surrounding the listeners, normally in shape of a sphere or a cube, this technique uses three loudspeakers to produce spatial sound impressions.

[25] cp. :Begault, Durand R. : 3-D sound For Virtual Reality And Multimedia, Moffet Field, NASA Ames Research Center, Academics Press Professional, 1994, Reprint 2000, p.176

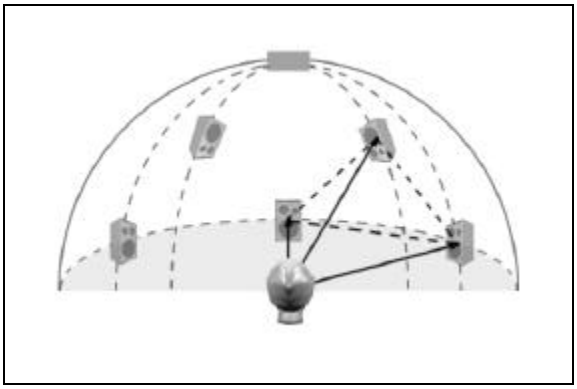

Figure 8 : Triangulation of loudspeakers and loudspeaker positioning

For the technique to work the loudspeakers of the used triplet generating the virtual source must not be in the same plane with the listener.
Combinations of triangles build of the loudspeaker array are not overlapping and the used triangles have as short sides as possible. (Fig. 8)
Knowing the position of the virtual source to be created, the triangle of loudspeakers around it are destined and the according amplitudes for the loudspeakers are calculated.
This technique implies a listener position in the focus of all of the used loudspeakers. The same restriction valid within stereo reproduction for the listener position are valid here. The optimal localisation can only be reached in a small listening area where the listener has to stay in.
The technique is called Vector Based Amplitude Panning (VBAP) and was introduced first by Ville Pulkki in August 2001.[26]

The CPU power necessary for 3-D sound reproduction can be reduced more by simplifying the VBAP technique. Not a triangle is used to generate the virtual source and the spatial imagery connected to it. All of the loudspeakers in the array contribute to the reproduction of the virtual source.

---

[26] cp.: Pulkki, Ville, Spatial Sound Generation and Perception by Amplitude Panning Techniques, Helsinki, Laboratory of Acoustics an Audio Signal Processing - Helsinki University of Technology, Espoo 2001, Report 62, 2001, pp.17-18

The amount of speakers, the vector between the listener position and the intended virtual source position as well as the vector between the listener and the speaker is taken into account when calculating the corresponding amplitude for each speaker in the array.
This simplification leads to abridgement of calculation time and decrement of used CPU load through leaving out the triangulation. This approach is causing the fewest CPU load while relying on the biggest hardware demands. It is called Ambisonic Sound Reproduction. [27]

2.2.3. Headphone Versus Loudspeaker Reproduction

Comparing headphone to loudspeaker reproduction several different aspects have to be taken into account. The aspect of ease of use is taken into account as well as the reachable audio quality and the costs of reproduction.

The reproduction of 3-D spatial acoustic imageries using headphones is considered optimal because of their relative immunity to acoustic detriments and the consequent power with which a recording engineer can predict resulting perceived spatial attributes. With substantially less efforts at least the same bandwith and the same acoustic pressure is reachable as with loudspeaker reproduction. The intensity perceived in natural situations can always be played back using headphones. Headphone–based displays allow the greatest degree of control over the location of a spatial source.
Additionally the listener is shielded from any disturbing background noises and reflections caused through a lack of sound-damping in the listening room.[28] Finally, serving audio for a great number of listeners at different locations, reproducing the same 3-D sound impression is possible using headphones.

---

[27] cp. :Pulkki, Ville, Spatial Sound Generation and Perception by Amplitude Panning Techniques, Helsinki, Laboratory of Acoustics an Audio Signal Processing - Helsinki University of Technology, Espoo 2001, Report 62, 2001, pp.14-18

[28] cp.: Begault, Durand R. : Auditory And Non-Auditory Factors That Potentially Influence Virtual Acoustic Imagery, Moffet Field, Human Factors and Technology Devision - NASA Amees Research Center, 1999

There are disadvantages connected to the reproduction using headphones. Obviously the listener has to be connected to something, carrying technical equipment. Even if the headphone is connected through infrared or is radio controlled, the listener still has to wear the headphones themselves.
The algorithm needed to successfully implement a realistic binaural 3-D sound reproduction uses a high amount of CPU power. Trying to base the binaural reproduction on non individualised HRTF will alleviate the computation but result in a imprecise localisation of 3-D sounds in space and errors in reversal, azimuth and elevation occur.[29] Moreover the headphone reproduction causes in-head localisation of sounds, the sound seeming to originate from inside the head. This is caused by a fixed audio reproduction neglecting the turn of the head. Capturing the look-at direction using head tracking techniques resulting in the known disadvantages is necessary then to overcome the inside location effect.

Headphone listening conditions can be approximated using stereo loudspeakers in combination with cross talk cancellation as mentioned. Relying on binaural algorithms for reproduction of a 3-D sound imagery using loudspeaker does not solve any of the problems mentioned for headphones except for the headphones that do not have to be carried any more. In contrast the loudspeaker reproduction using the same algorithm as used for headphone reproduction causes more problems. Since the 3-D sound impression will be most successful when the listener's position is fixed an known in advance, the listener has ideal to stay at one position. Even turning the head would cause the destruction of the 3-D sound impression. Head tracking could be done but there would no way to adjust the sound in that way that the averted ear would receive the right audio cues.

While headphones were shielding the listener from any influence by the environment the listener is in, effects like reflections or noise can disturb the sound perception now. Other approaches for loudspeaker reproduction are available.

---

[29] cp.: Begault, Durand R., Wenzel, Elisabeth M.: Headphone Localization of Speech, Moffet Field, NASA Ames Research Center, Human Factors, 1993

With the use of a loudspeaker array several advantages rise and some disadvantages disappear.
Using multichannel techniques the environmental influence through e.g. reflections is still present and cross talk does still appear, but the influence on the perceived 3-D sound imagery is lower. The listener still has to stay in a predefined position to experience the best impression but head tracking is no longer necessary, since the sound source's position is simulated and not the binaural cue at the ear drums. This leads to a fixed virtual source resulting in natural effects at the listener's ears when turning the head.
The lower CPU load needed makes the reproduction of more sound sources possible or enables more physical sound effects to be calculated in real-time resulting in a more realistic impression.

2.3. 3-D Sound Integration in Virtual Reality Set-ups

With the integration of 3-D sound in virtual reality set-ups the existing infrastructure actually used for virtual reality and other real-time applications can be enriched by one dimension, the sound.
As shown in the previous chapter, the real-time computation of 3-D sound imageries causes great CPU load. Methods for including a computational expensive real-time 3-D sound system into an other computational high cost system needs a lot of aspects and possibilities for adjustment to be taken into account.
"Perceptual requirement must be balanced against available resources. This process not only involves realistic limits imposed by time and monetary budgets, but also the technological aspect of the audio system as a whole. A designer must envision the nature of the sound system integration requirements, in terms of hardware and software. This task is partly on inventory of computer resources (memory, available instructions) and external hardware, such as the effectors and sensors already in use for the visual displays."[30]

[30] cp.: Begault, Durand R. : 3-D sound For Virtual Reality And Multimedia, Moffet Field, NASA Ames Research Center, Academics Press Professional, 1994, Reprint 2000, p.95

Two systems, the virtual reality system and the 3-D sound subsystem have to be connected. The virtual reality system involves software calls to the operating system and graphical displays as well as the reception of user input calls from any input device.
It holds the representing data structures of virtual worlds and is also responsible for distributing software calls to other designated subsystems involved in the representation of the virtual world. A 3-D sound system is a subsystem designated to the reproduction of audio in general and 3-D sound in special.
The 3-D sound subsystem responds to software calls from the scenario control system software. The scenario control system can include data from user input devices like head tracking systems for example and is part of the virtual reality system. The 3-D sound subsystem's task is then to manage internally the input and output of audio, to control data flow to other audio components, like specialised audio cards for instance, and to distribute and route audio source signals to one of any number of components as well as combining and mixing signals.[31]

The inclusion of any subsystem, respective a 3-D sound system, can be implemented either in a integrated, a distributed or as a mixture of both, in a hybrid way.
Integrated systems consist of all of the hardware and software contained within the same computing platform. The virtual reality application and the 3-D sound subsystem is then running on the same machine, shortening and easing the communication between them.(Fig. 9)

[31] cp. :Begault, Durand R. : 3-D sound For Virtual Reality And Multimedia, Moffet Field, NASA Ames Research Center, Academics Press Professional, 1994, Reprint 2000, p.97

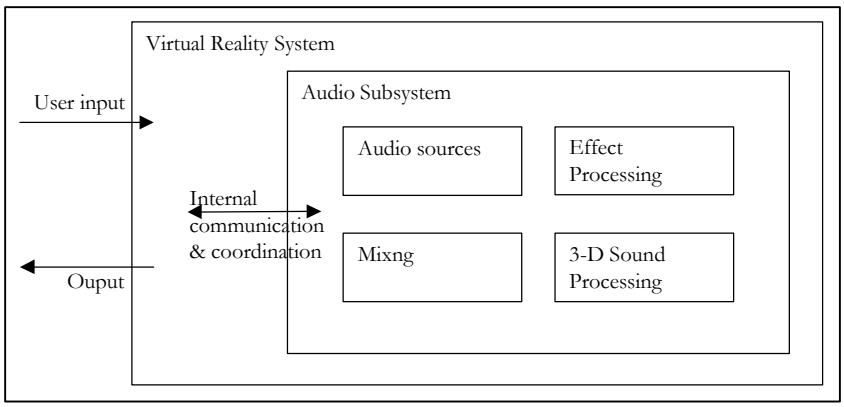

Figure 9 : Integrated System approach for 3-D sound system integration

In distributed systems the virtual reality system is running on a different machine than the 3-D sound system. In general every service participating in the virtual reality application can be distributed to other platforms. There could be several subsystem combined, like a graphical subsystem for visual representation, an input subsystem for receiving and processing user input and others.

This approach reaches in high effort for co-ordination and communication. Although the subsystem itself is optimised for one operation, performing better than in an integrated approach, the system can be slowed down through the immense overhead.(Fig. 10)

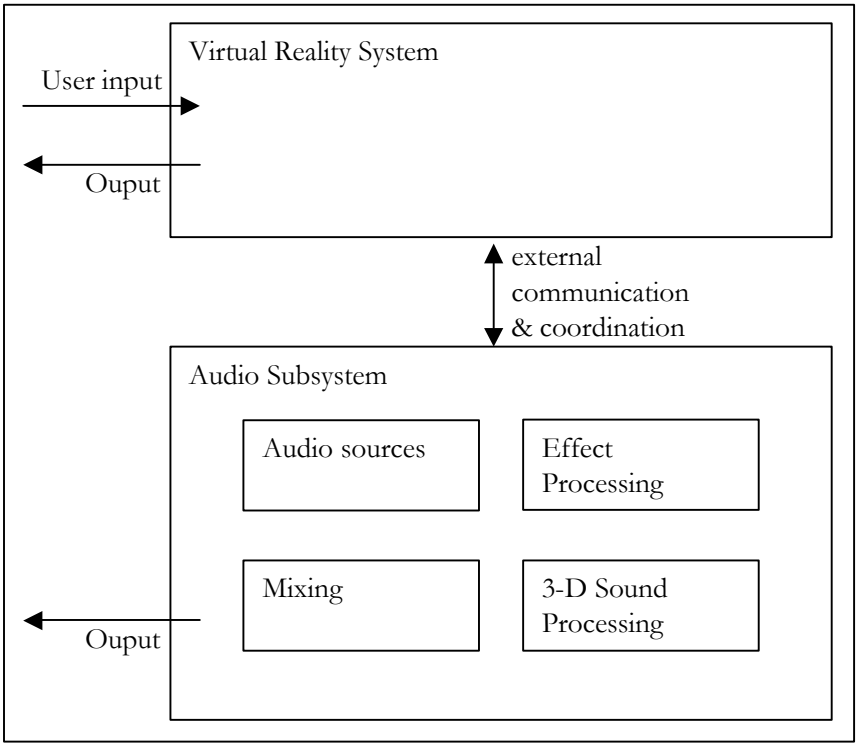

Figure 10 : Distributed System approach for 3-D sound system integration

A precise investigation of the effort necessary for communication has to be done and to be compared against the release of CPU time gained through outsourcing part of the computation.
Whereas some subsystem can suffer from a distributed approach, other subsystems can profit from it. Systems where the speed up in processing through the optimisation of a subsystem overcomes the overhead, profit from outsourcing the corresponding subsystem.

Hybrid systems combine a virtual reality system with outsourced subsystems that cause high CPU load. While minimising the loss through co-ordination and communication the win through outsourcing services is maximised.

3. Real-Time 3-D Sound API requirements and Related Approaches

Defining minimum requirements for real-time 3-D sound APIs helps to find a common base for developers to work on. Minimum requirements can further be used for estimating existing APIs and classifying their quality.
In the first part requirements for 3-D sound implementations are carried together whereas in the second part existing approaches are compared against these minimum requirements.

3.1. Real-Time 3-D Sound API Requirements

Beside the demands for the access points, services and functions an API offers the ease of use and the dependency on special operating systems and hardware set-ups has to be taken into account when composing minimal requirements for an API.

3.1.1. Hardware Dependencies

A 3-D sound API should be designed to support any hardware set-up available for real-time audio computation and audio reproduction. The key is not to rely on any specific sound hardware components, the API has to rely on access points offered by the operating system to access the sound hardware, neglecting the details of the hardware. An API that does rely on specific hardware will hardly be adjustable if further hardware becomes available. The frequency and volume of change would be too high to adjust the API simultaneously with the progress in hardware development.
Shifting this task to the developer of audio hardware, keeps the API clear of any specific hardware implementations and leaves the updates to the operating system and driver development.

3.1.2. Operating System Dependencies

Opening the field of 3-D sound for a great number of developers is one of the targets the 3-D sound API pursues. To serve all sorts of developers and application users working on all kinds of workstations, an API must not be operating system dependent.

Operating system independence can only be reached using functions and software calls that do not use any of the operating system features or services and does not rely on programming language features the are connected to any operating system services.. Since it is a good idea to delegate the hardware access to the operating system as it was stated above, the 3-D sound API is forced to do rely on it in the same place.
To keep the reliance on the operating system as low as possible to enable an easy and quick translation from one operating system to others available, encapsulating the operating system reliance in one segment of the API is desirable.

3.1.3. Access Points, Services, Functions and Components

Requirements necessary to represent naturally appearing 3-D sound processed in real-time are reviewed.
The requirements should satisfy the expectations of most developers and consumers.

A group of hardware and software vendors and application developers founded the "Interactive Audio Special Interest Group" (IASIG) to influence and improve the development of platforms for interactive multimedia, in the area of 3-D sound playback. The IASIG published a paper defining the minimal expectations to a 3-D sound API and the sound system that the API is based on. The functionality is based on what is possible with technology today at acceptable price points. A second paper relying on future and high-cost technologies does widen the expectations. Although the group does not meet any more, the expectations are still up to date.
The expectations where set-up with producing naturally experienced 3-D sound in mind. Originating from that goal the minimum requirements have been investigated.

In a first step, only the requirements based on today's technology are taken into account. These are fixed in the "IASIG Interactive 3D Audio (Level 1) Requirements".[32]
In order to represent the listener in the virtual world, there must be a listener object. The listener object must be positionable in a Cartesian coordinate system.

---

[32] cp.: IASIG, IASIG Interactive 3D Audio (Level 1) Requirements, 1998

The audio environment should be viewed from the perspective of the listener and correspond with the view being depicted on the graphical display. The location of the user's head and the head's orientation are key to locating the ears, needed for proper rendering of the audio content. Hence the orientation of the listener must be settable together with his position. For effect calculations like the Doppler shift, knowing the velocity of the listener is important, so it has to be determinable, too.
Sound source objects then have to hold at least the same attributes as the listener, position, velocity and orientation for directional sounds. Having the attributes of both, the listener and the source, the relative distance and velocity between them can be estimated and exploited in further calculations. Additionally the radiation pattern of the sound have to be setable - the algorithm for radiation of sound is different for directional sounds, emitting sound primarily in one direction, and omni-directional sound.
The more of the sound sources that can be rendered in real-time at once, the better the interactive rendering engine can produce the illusion of a realistic sound environment. A minimum of eight sound sources processed at once should be reached by every 3-D sound API architecture.
The object's attributes have to be exploited – effects like Doppler shift, translating the sound radiation model or distance attenuation have to be rendered in real-time for each source marked for processing.
To guaranty a minimum quality standard the API rendering must be done all in real-time at a minimum sample rate of 22050Hz using a 16bit coding scheme. There must not be any audible latency or artefacts.

3.2. Related Approaches

Some APIs concerning the implementation of 3-D sound available for varying system set-ups have been developed. The different approaches are introduced and the underlying concepts are shortly explained.
This section ends with a conclusion part, summarising the obtained insight in the approaches confronting the advantages and disadvantages of the approach and explaining the conditions making an other API desirable and necessary.

3.2.1. Java 3DÔ API[33]

The Java™ 3D API was originally designed to be used for 3-D graphic applications. Following the philosophy "write once, run anywhere" the API is designed to be platform independent, supporting the total set of the Java™ class structure. The ideas behind Java 3D™ were derived from existing low-level graphic bibliographies like Direct 3D™, Quick Draw 3D™ and others.

For audio device support first a "Audio Device Interface" were added to the Java™ 3D API. Later the audio interface was completed with the introduction of the "Audio Device 3D Interface".

With the introduction of the audio interfaces, accessing the hardware audio devices is possible. For audio reproduction one or all of the offered audio devices on one machine can be chosen as well as choosing headphones or loudspeakers for reproduction is possible.

Details for reproduction can be set after having initialised the hardware device. Determining the amount of loudspeakers as well as their positions is possible allowing to take individual audio set-ups into account. To make these features available, the abstract audio interfaces have to be implemented for every audio hardware device to be assessable through Java 3D™.

The "Sound Node Object" used for sound source representation is defined as an abstract class defining attributes all sound source types have in common – including attributes like the reference to the binary data, amount of repetitions, the current state and the priority.

A "Background Sound Node" representing environmental sound and "Point Sound Node" as well as "Cone Sound Node" for representation of omni-directional and directional sound sources are also valid.

Effects available are at present limited to distance attenuation, reverberation and calculating the Doppler Shift.

---

[33] cp.: Sun microsystems, The Java 3D API specification, http://java.sun.com/, 11.02.2002

Since the Java 3D™ API does rely on Java™ technologies it is evenly designed to be platform independent. The platform independence of the Java™ approach has been proven in practical applications over time.

3.2.2. DirectXÔ Audio API[34]

DirectX™ Audio is part of the complex DirectX™ API. DirectX™ Audio offers a complete system for integration of a dynamic soundtrack using the advantages of hardware acceleration, downloadable sounds, DirectX™ media objects and advanced positioning effects.

With the DirectX™ Audio API simultaneously playing back several sound sources, that can be positioned in 3D space, is possible. Sound effects like pitch changes, reverberation or the Doppler shift are adjustable to the sounds.

For developers the API offers easy access to low-level libraries and is extensible in function. It is available for operating systems Microsoft Windows95™, Microsoft Windows98™ and Microsoft Windows2000™.

Hardware devices are accessed through DirectSound™ or DirectSound3D™ drivers.

Audio data is represented mainly using two kind of objects -the "DirectSound 3-D Buffer Object" and the "DirectSound 3-D Listener Object".

The first is used to represent a sound source in 3-D space. It holds a sound source's attributes like position, orientation, processing mode, minimum and maximum distance, gain and others.

The later object holds attributes related to the listener involved in the 3-D sound scenario. Attributes like the listener's velocity, his position and orientation are of great significance for 3-D sound effect processing.

Further attributes influencing the effects being calculated can be set.

---

[34] cp.: Microsoft, MSDN Library – DirectX, http://www.microsoft.com/directx/, 11.02.2002

3.2.3. OpenAL[35]

OpenAL stands for "Open Audio Library". It is an approach to set-up an open library for implementing real-time 3-D sound. The initiative did not rise out of the interests of only one company, several companies are involved in the initial declaration of goals, features and implementation.
OpenAL is a software interface for audio hardware access, designed to be easy to use and to be platform independent. It fulfils the minimum requirements of the "IASIG Level One Requirements" as well as the "IASIG Level Two Requirements".(cp. 3.1.3)
For data representation OpenAL uses two basic objects, the listener object representing the listener in a virtual world and the source object representing sound sources in 3-D space, which refer to a sound buffer object, holding the binary stored data.
The listener is a unique object, whereas several sources can exist.
OpenAL offers a set of commands making positioning of the listener and the sound sources in 3-D space as well as controlling over them possible. Since OpenAL leans on the OpenGL specifications similar implementation techniques used there.

3.2.4. VRML

VRML as an abbreviation for "Virtual Reality Modeling Language". It is a fileformat for describing interactive 3-D objects and virtual 3-D worlds.
VRML is designed to be used within the internet context and in locale client server systems. Moreover VRML is ment to be an exchange format for 3-D graphics and multimedia content.
Further VRML is designed to fulfil the needs of authorability, composability, extensibility, capability of implementation and performance.
While authorability and composability are primary addressed needs for building applications using the VRML format, the aspects extensibility, capability of implementation and performance are related to the low-level VRML implementation. The later needs are concerned with the ability to enlarge the capabilities of the VRML format to guaranty the executeability on every hardware system, supporting every operating system and optimising the performance of VRML based applications.

---

[35] cp.: Loki Software, OpenAL Specification and Reference, http://www.openal.org, 2000

In VRML the data is organised in a tree like structure.
The "Sound Node" is used for representing sound sources in a virtual world. It holds information on the position, the direction, location and priority. The "Sound Node" is connected to an "Audio Clip Node" which holds the link to the binary data as well as attributes concerned with the audio play back.
The positioning of sound sources is only possible in the listener's horizontal plane.

3.2.5. Conclusion

Several of the essential needs as mentioned in the first section of this chapter like high performance and platform independence are served by the introduced APIs. However every of these API s suffer from insufficiency in one or more of the areas most important for a real-time 3-D API, making their use not desirable.

Pursuing the goal of developing an easy to use API for quick development of high quality multimedia applications, with Java 3D™ an API was introduced, not optimised for performance, for usability instead. As desired it is platform independent. To implement the necessary functions for 3-D sound reproduction Java 3D™ uses calls to other existing low-level APIs, causing additional latency.
Practical applications implemented in Java™ and compared to applications written in other, low-level development languages, such as 'C', have proven, too, that Java™ suffers from not being as high performing as the mentioned low-level environment.

The DirectX™ Audio API is more complex than the Java 3D™ API.
Audio hardware vendors started to implement some of the major functions offered within the API in their digital signal processing chips used on audio cards to optimise performance. More CPU power then is available for other tasks, like graphical applications.
DirectX™ Audio suffers from being dependent on Microsoft Windows™ operating system.

These operating systems are wide spread, but referring to the part "Operating System Dependencies"[36] an API offering general approach to the whole field of applications should be available platform independent, opening the area for any kind of developer to push further developments in this field of interest.
Due to the closed source concept transporting the DirectX™ Audio API to other platforms is an time intensive, strenuous effort.

OpenAL™ was introduced to be a platform independent and high performance API. Building an application on a Microsoft Windows™ platform, OpenAL simply uses a redirect of function calls to in DirectX™ Audio. The redirect used to realise the OpenAL™ in co-operation with Microsoft's operating system causes further latency. Tests have shown that programs developed using OpenAL™ were not portable between operating systems. Platform independence is not realised caused by varying calling structures and routines within the operating system dependent implementations.

Since VRML is only able to process horizontal positioned sound sources it cannot be taken into account for a further exploration in the field of real-time 3-D sound, although the platform independence needs are satisfied and the VRML file format is designed open to be extendible.

With the spotted out shortcomings of the already existing approaches, designing a new API serving all of the needs is desirable.
Since every API introduced here has advantages comparing to other approaches and every API has special ideas on how to implement functionality, a new design will be inspired by the already existing ones.

[36] cp.: 3.2.1

4. 3-D Sound System Analysis

To realise the demands on a synthetic 3-D sound system, the natural 3-D sound system must be understood. Based on an analysis a model of the natural sound system can be visualised.
The model is then checked against applicable simplifications.
The simplified model represents the foundation for the software concept and the implementation.
To discuss all aspects of the natural listening process a data flow diagram is the first step in the analysis. Based on the data flow diagram the structure and meaning of the involved data elements are determined within the data catalogue. The interaction between the data elements is investigated and visualised in the process specification.

4.1. Data Flow Analysis

Envisioning the context of the 3-D sound system, there are at least two objects interacting through the 3-D sound system. An object is generating a sound which is somehow transferred to a listener who is perceiving it. (Fig. 11)

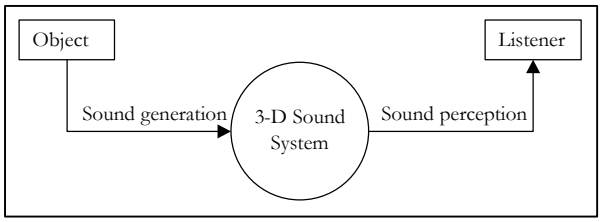

Figure 11: Context model of the 3-D sound system

Further investigating the 3-D sound system points out the environment in which the listener is in, other objects in the environment and the sound itself are interacting, influencing sound propagation.

Defining the sound object to be a property of an geometric object makes a separation of the object itself and the sound object possible. (Fig.12)
The sound object is holding audio specific information represented in it's attributes and the graphical object is containing information about the physical aspects of the object itself.

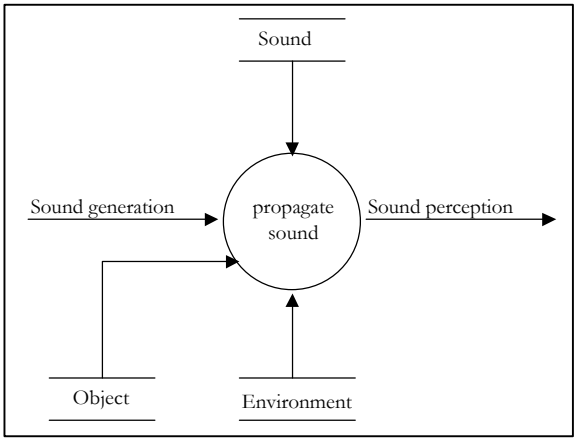

Figure 12: Refinement of the 3-D sound system

The sound object is then responsible for the creation of the sound wave. The object's physical attributes are influencing the sound wave. Additionally the sound wave is influenced by the environment the sound is propagating in.
The influence on the sound wave is impairing the sound propagation.(Fig.13)

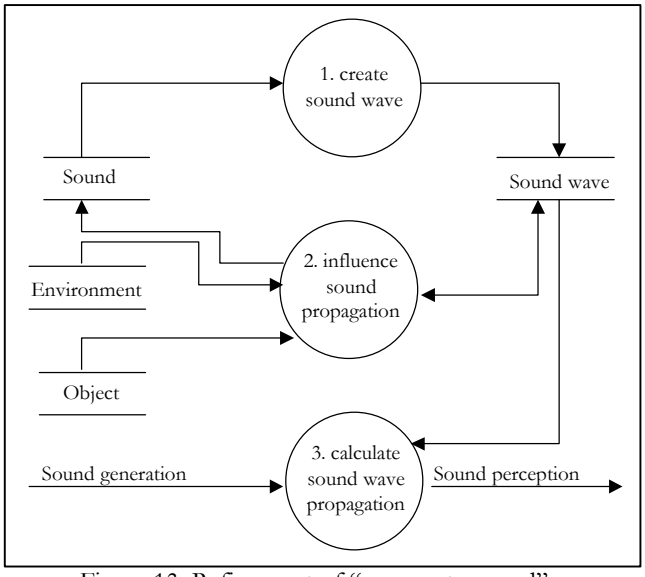

Figure 13: Refinement of "propagate sound"

For the creation of a sound wave the physical attributes of the object as well as the attributes of the sound object have to be taken into account. Based on these information the direction for sound propagation can be determined and one oscillator can be generated. Combining the oscillator with the propagation direction makes the emission of a sound wave possible. (Fig.14)

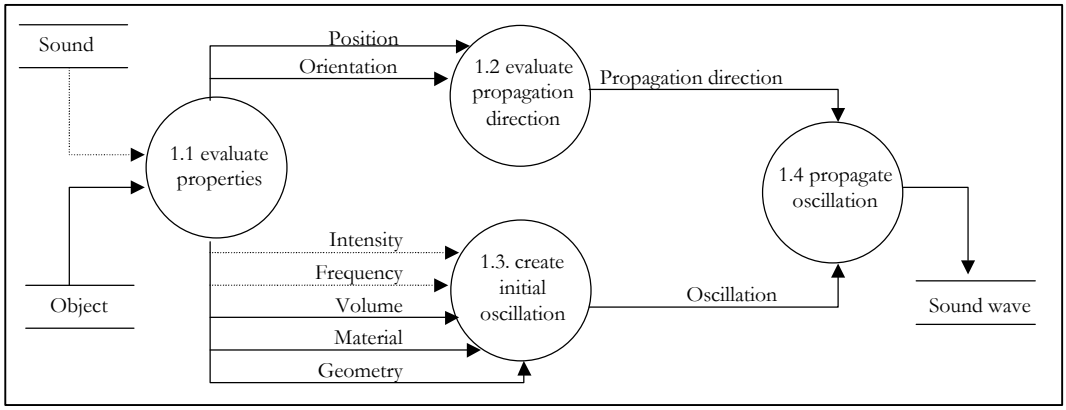

Figure 14: Refinement of "create sound wave"

After initial creation of the sound wave it is manipulated by the environment and objects in it as mentioned before. Depending on the sound wave's -, the object's - and the environment's properties the manipulation is performed.

Objects are able to damp, diffract or reflect sound waves while the sound waves is reaching the object interacting with it's surface. Sound wave reflection is also possible. When reflecting a sound wave either all of the sound wave or part of it is reflected appearing to the listener to be an extra sound source. Frequency shift through interaction between objects and sound waves is possible, too.

Through propagation in the environment the sound wave is attenuated or slowed down by the influence of the environment itself. (Fig. 15)

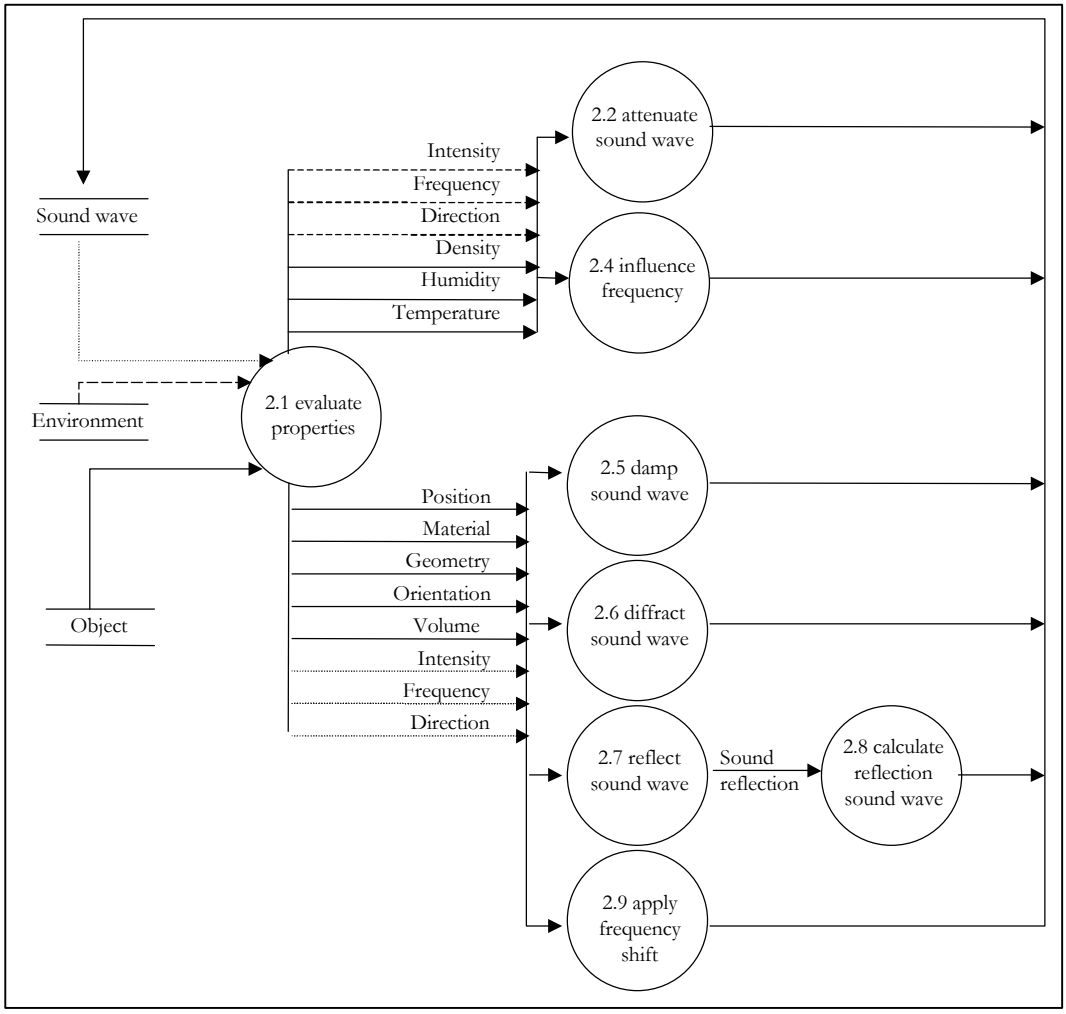

Figure 15: Refinement of "influence sound propagation"

Continuously the sound waves propagate through the environment. For every time step an image of the current sound wave can be taken, describing the sound wave.
These images can be interpreted by the listener. The sound wave's properties include psychological as well physical attributes about the sound and the object that emitted that sound. (Fig. 16)

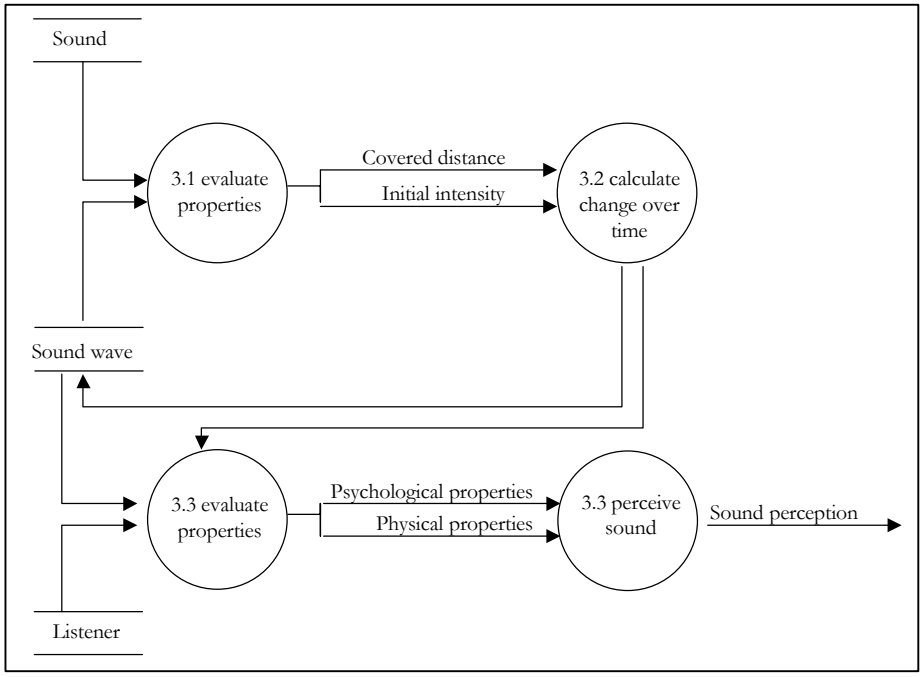

Figure 16: Refinement of "calculate sound wave propagation"

4.1.1. Data Flow Simplification

Making an API available for a wide spread variation of audio reproduction hardware set-ups, the API must consider low performance set-ups as well as professional systems for sound reproduction. This consideration makes an modular solution necessary.
The API must have a base kernel offering the minimum functionality for 3-D sound reproduction in real-time. This kernel must be designed to be executable on standard system set-ups. Additional features implemented in extension modules must be integrable on user demand for more efficient hardware set-ups.

Taking a modular implementation as a basis for a real-time 3-D sound API, simplifications of the model developed in the preceding part have to be applied to reduce the model to the minimum to find the demands on the kernel implementation. Minimising the model requires the definition of minimal features.

Although the requirements defined by the IASIG in the "IASIG Interactive 3D Audio (Level 1) Requirements" [37] are designated to be minimum requirements for an API to be considered as an real-time 3-D sound API, the requirements are beyond the minimum.

Hearing a sound in nature, the listener intuitive finds himself answers to two questions. What kind of object is emitting the sound and where is the sound coming from?
These two questions can be determined to be the questions the 3-D sound API kernel is all about.

Further questions are not directly connected to the sound in 3-D space or can be simplified through a reformation of the basis questions.
Answers to the question for the kind of object emitting the sound can easily be found. A known object produces known sounds. Identifying a known object is simple. The nature of unknown sounds on the other hand is characteristic for the object they originate from, leading to an assumption about the object's size and nature although not knowing the object in detail.[38]
Locating the object's location and thereby answering the question of determining the direction the sound is coming from is done in a second investigation.

While information about the object's characteristic can be obtained for known as well for unknown sound sources, determine the distance to the sound source is more difficult. Even for known sounds the human distance perception is weak. For unknown sounds it is almost impossible.
Human distance perception is primarily derived from the sound's loudness. The memorised original sound pressure at minimum distance is thereby compared against the actual perceived sound intensity.
Using an artificial technique for sound reproduction like done in the 3-D sound API approach, only the headphone reproduction can always guaranty to reproduce a sound with the same sound pressure as perceived under natural conditions. Nevertheless even if the natural reproduction is possible it is seldom applied.

---

[37] cp. Section 3.1.2

[38] cp. Section 2.1.1

(Hearing a plane's turbine placed close to your visual representation's ear with the same sound pressure as it appears in nature is not desirable.)
With this restriction, every sound being played back using an artificial technique must be considered to be unknown, since the listener has no reference loudness to derive the distance from, resulting in almost impossible distance perception.
Since distance perception is almost impossible, no matter what kind of hardware set-up is used, it does not have to be supported by the API kernel.

The perception of direction relies on the interaural time and intensity differences between the listener's ears. Besides the known limitations, determine a sound's direction is precise for known sounds as well as for unknown ones. [39]

Concentrating on information about the sound source itself, questions about the environment the source is in can be kept out of the kernel implementation as well as influences on the perceived sound by other objects.

Neglecting the influence by the environment and imagine the object that emits the sound to be the only one, eliminating other objects' influence in the same step, only the initial sound wave and the corresponding objects properties regarding it's position and orientation have to be taken into account.
The applied simplifications lead to a minimised system model. Thus making a high performance API kernel for low hardware resources possible.
Based on the same context model the refinements are simplified by deleting the aspects originating from the environment and other objects. Neither the influence on the initial oscillator nor the sound propagation have to be taken into account. (Fig.17 – Fig.21)

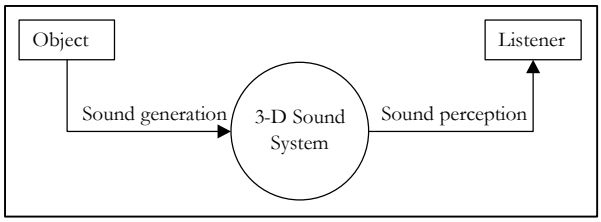

Figure 17: Context model with simplifications

[39] cp. Section 2.1.2

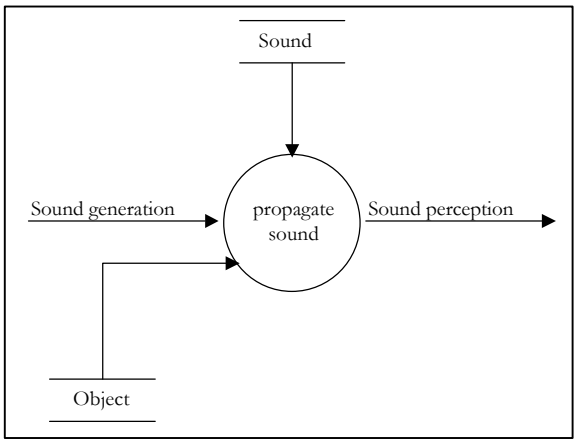

Figure 18: Refinement of the 3-D sound system with simplification

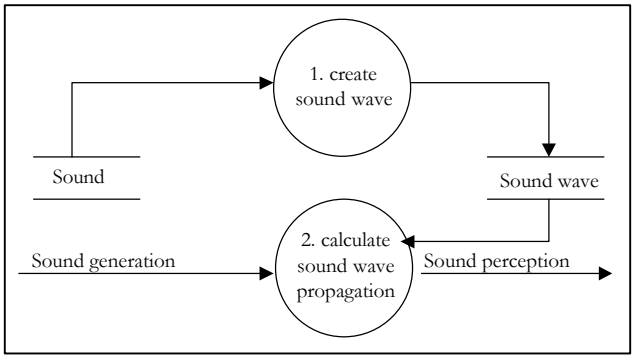

Figure 19: Refinement of "propagate sound" with simplification

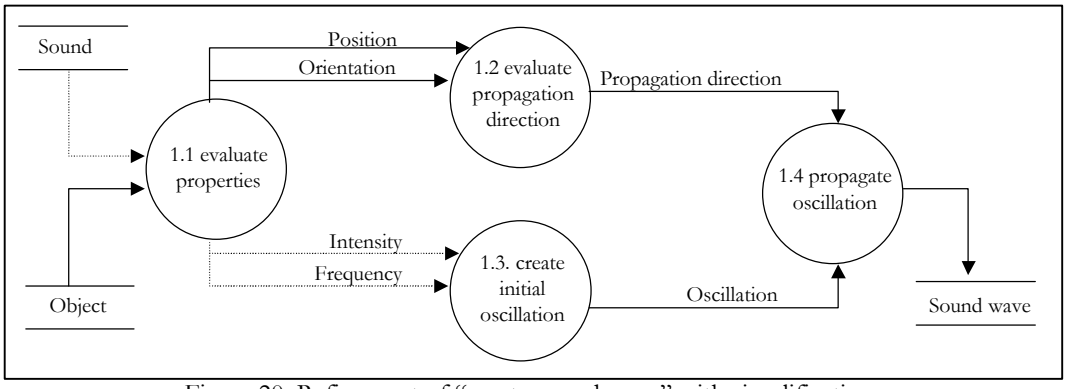

Figure 20: Refinement of "create sound wave" with simplification

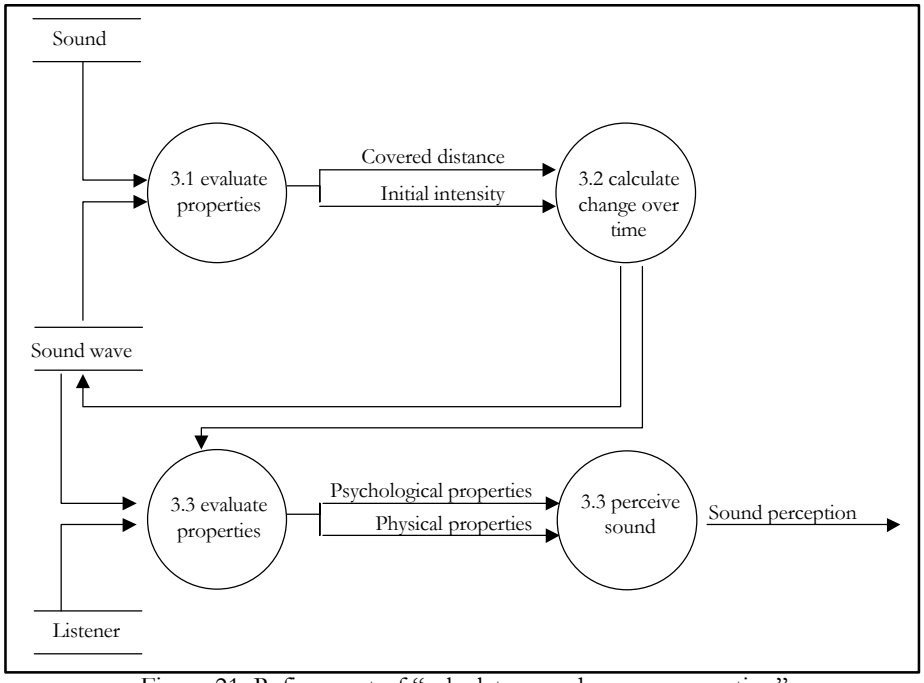

Figure 21: Refinement of "calculate sound wave propagation" with simplification

4.2. Data Catalogue

As shown in the previous section the object emitting the sound, the sound itself, the sound wave and the listener are the components influencing the perceived sound impression.

Individual data flows originating from or flowing to the involved object's and caused by other processes or extern influences contribute to the sound propagation process and the included sub processes.

Based on the simplified model the data aspects of the involved objects are investigated and abided in the data catalogue.

SoundGeneration	=	Object + Sound *The Object attributes and the sound properties are considered for sound generation*
SoundPerception	=	PsychologicalProperties + PhysicalProperties
PsychologicalProperties	=	*Individual feeling of the perceived sound*
PhysicalProperties	=	Direction + Intensity + Frequency *Attributes of the arriving sound wave*
Listener	=	Position + Orientation *Position and orientation are the essential listener cues needed for sound perception simulation*

SoundWave	=	Sound + Direction *Sound propagating in a certain direction*
Sound	=	Oscillation + Intensity *Oscillation with time dependent intensity *
Oscillation	=	Position + Frequency *Position the oscillator is placed at oscillating with a certain frequency*
Object	=	Position + Orientation *Orientation influences sound propagation*
CoveredDistance	=	Number *Distance the sound wave has progressed*
InitialIntensity	=	Number *Initial amplitude of the oscillator's vibration*
Intensity	=	Number *Amplitude of the oscillator's vibration*
Frequency	=	Number *Vibrations of the oscillator per second*
Propagation direction	=	PointIn3D *Direction the sound waves propagates in*
Position	=	PointIn3D *Position the object/listener is stationed at*
Orientation	=	PointIn3D *Point the the object/listener points to / the sound wave propagation is directed to*
PointIn3D	=	CoordinateValue + CoordinateValue + CoordinateValue *A point is specified by the x,y,z coordinate values*
CoordinateValue	=	Number
Number	=	*any number*

4.2.1. Data Catalogue Simplification

The scope of this thesis is about sound representation and reproduction of 3-D sounds. The sound syntheses including the investigation of the sound's nature, the initial oscillator creation and the sound propagation calculation is beyond the thesis' point of concern.

Sound is taken to be given in binary format, representing an already synthesised sound. The data aspects of "SoundGeneration", "SoundWave", "Sound" and "Oscillation" and the underlied data elements are externalised then.

The artificial sound generation deals with the sound reproduction at the position and in the direction given by the corresponding object.

The "Psychological Properties" are encoded in the binary sound representation.

Based on theses assumptions the data catalogue is reduced to the base aspects in a next step.

Listener	=	Position + Orientation *Position and orientation are the essential listener cues needed for sound perception simulation*
Object	=	Position + Orientation *Orientation influences sound propagation*
SoundGeneration	=	Sound + PhysicalProperties *Sound reproduction at the position and in the direction given by the corresponding object hold in the physical properties*
Sound	=	*Binary pre sampled audio* + PhysicalProperties *The physical properties are kept for quick access*
PhysicalProperties	=	Direction + Intensity + Frequency *Attributes of the arriving sound wave*
Intensity	=	Number *Amplitude of the oscillator's vibration*
Frequency	=	Number *Vibrations of the oscillator per second*
Position	=	PointIn3D *Position the object/listener is stationed at*
Orientation	=	PointIn3D *Point the object/listener points to / the sound wave propagation is directed to*
PointIn3D	=	CoordinateValue + CoordinateValue + CoordinateValue *A point is specified by the x,y,z coordinate values*
CoordinateValue	=	Number
Number	=	*any number*

4.3. Process Specification

Through the applied simplifications the kernel processes are reduced to three main processes. The assumptions made in the previous sections are valid here, too. The consideration of processes related to sound synthesis is unnecessary.
The process specification is done using structured formulation technique.
The procedure of "evaluate properties", "propagate oscillation", and "perceive sound" are described within this section. The described processes are identified to be the essential ones for the kernel.

evaluate properties :

- For Listener
  - get listener's position
  - get listener's orientation
- For Sound
  - get sound's binary sound data
  - get sound's physical properties
    - get sound's properties' direction
    - get sound's properties' intensity
    - get sound's properties' frequency
- For Object
  - get object's position
  - get object's orientation

propagate oscillation :

- get properties of the object emitting the sound
- get listener properties
- IF object is within listening range of listener
  - IF listener is within the direction the object's orientation
    - evaluate distance between listener and object
    - initialise sound with initial intensity and frequency

```
        ELSE
            do not initialise sound

perceive sound :
        get listener properties
        get sound properties
        evaluate angle between listener's orientation and the sound's direction
        FOR both ears
            evaluate time difference to other ear
            evaluate intensity difference to other ear
                IF ear's corresponding intensity > 0
                    evaluate HRTF for the ear
                    adjust sound based on the initialised sound using the
                ELSE
                    totally attenuate the sound
        reproduce the adjusted sound
```

5. Hardware Set-up and Integration

For an API implementation a hardware set-up must be available supporting 3-D sound reproduction in real – time.
Estimating the requirements on a hardware set-up for a 3-D sound system has to take into account the context in which the system integrates in as well as the used audio reproduction technique.
The current virtual reality infrastructure at the Fachhochschule Wedel where the 3-D sound system will be integrated is introduced in a first phase as foundation for further concerns.
The following investigations leading to the chosen hardware set-up are divided into two steps. The requirements on the computational unit investigates the requirements on the computer system used for calculations and audio data playback are investigated in the first step.

For best integration in the given infrastructure the usability of different audio reproduction techniques is estimated in a second step..
The system requirements and the chosen audio reproduction technique lead to the final system configuration and operating system.
The last phase visualises the system's integration and operation in the existing infrastructure.

5.1. Available Virtual Reality Infrastructure

The 3-D sound system has to be composed to fluently integrate into the available virtual reality structures. Reaching a proper integration makes the understanding of the already available system necessary. The infrastructure and communication pipelines of the virtual reality laboratory installed at the Fachhochschule Wedel are described in this section.
The 3-D sound system is planned to operate in co-operation with the display system and the underlying structures.
The centre of the available distributed virtual reality system is formed by an SGI™ Onyx2 Infinite Reality 2 engine. The system is in particular used to control a large-scale stereo projection system. The projection system, a TAN™ Holoscreen, uses two projection units projecting on two screens. A vertical screen, accessed through backside projection, and a second horizontal screen, accessed through forefront projection, are used to generate an immersive display system. (Fig.22)

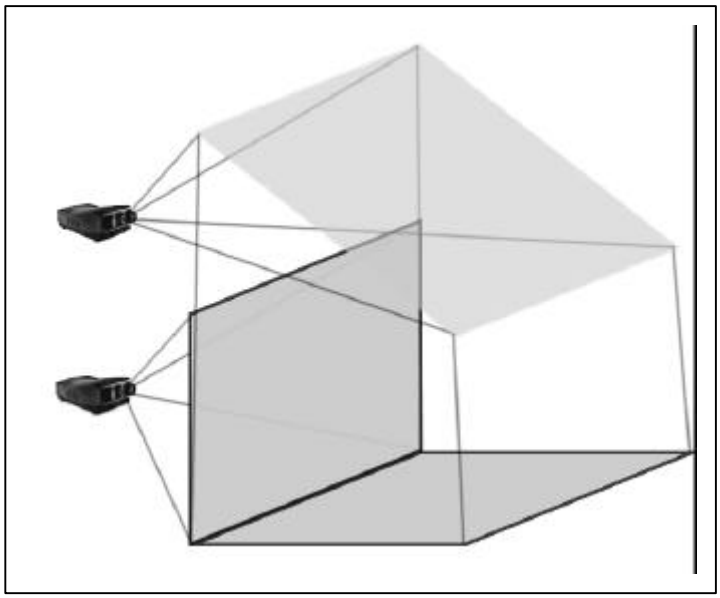

Figure 22: Immersive Display System (TAN™ Holoscreen) at the virtual reality laboratory, Fachhochschule Wedel

The main virtual reality application is executed on the Onyx2. Services offered by distributed system components factor into the application's execution.
An input device, a Polhemus tracking system, attached directly to the Onyx2 delivers position and orientation data captured through an electromagnetic tracking system.
Further a PC based system running under Windows NT™ offers motion capturing data, extracted from four installed infrared cameras attached to the PC. The PC itself communicates with the Onyx2 over ethernet. (Fig.23)

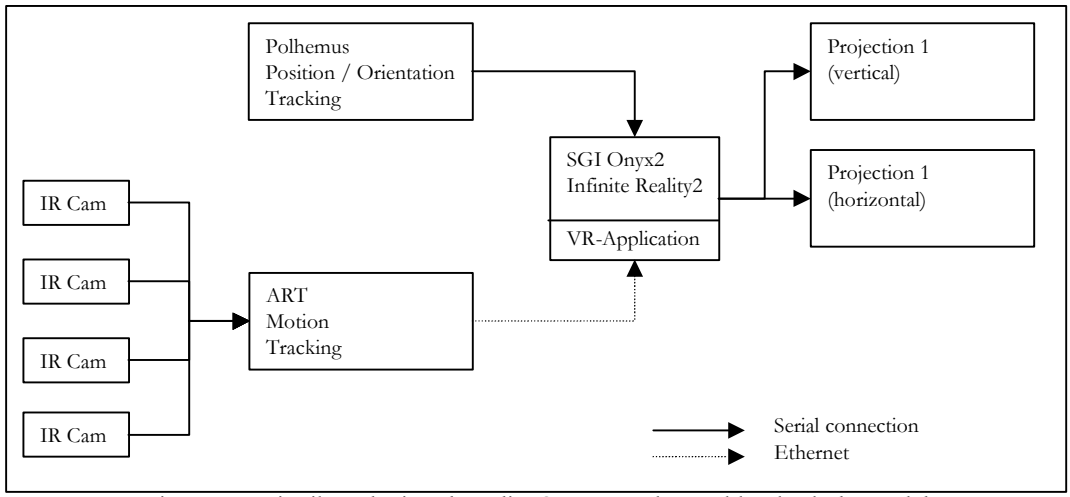

Figure 23: Distributed Virtual Reality System at the Fachhochschule Wedel

5.2. Hardware Requirements

3-D sound integration in virtual reality applications can be done based on a distributed as well as on an integrated system approach.
For future demands the system must be configurable to operate as a stand-alone integrated virtual reality system including 3-D sound functionality and it must be operateable within a distributed virtual reality system offering 3-D sound as a service.
Being adjustable to either a stand alone or a distributed system the API's functionality on both set-ups can be tested and proven using one hardware set-up.
Operating as a stand alone system the hardware must offer enough CPU power to serve the needs of the virtual reality application in addition to the needs of the 3-D sound calculations.
High performance graphic cards can contribute to release the CPU through operations encoded within the graphic card's chipset (e.g. Nvidia).
There must be no interruptions while playing back audio. To avoid disturbance during playback the binary audio data has to be present in the systems memory, leading to a great need for RAM. In case the RAM is insufficient and mechanisms of virtual memory allocation are used a fast system hard disk with large disk cache is necessary.

5.3. Audio Reproduction Evaluation

Following the path of relying on wide spread hardware, a simple two channel audio reproduction using loudspeakers or headphones would be the first choice.
Since the system developed within this project is designated to be used within the existing infrastructure of the virtual reality facilities a reproduction technique has to be chosen that fits in best.
A virtual reality application being visualised using the Holoscreen requires the user to be placed on the horizontal plane.
Within the horizontal plane the user is free to move within a certain area, exhausting the range of the Holoscreen's immersiveness.

A headphone reproduction would restrict the user's liberty of action through known reasons.[40] With a two channel loudspeaker reproduction the sound adjustment to the user's orientation and position would only be applicable to a certain point.
Only a multichannel reproduction based on a loudspeaker array can serve the demands for 3-D sound reproduction.
An array surpassing the horizontal plane must be chosen to result in a realistic 3-D sound impression.
Placing eight loudspeakers in each corner of a cube formed through six planes with the size of the horizontal and vertical plane of the installed Holoscreen results in the best 3-D sound impression.[41] (Fig.24)

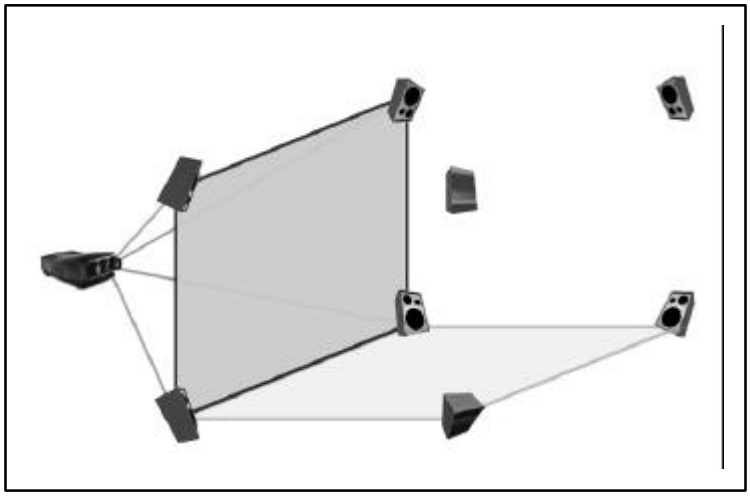

Figure 24: Integrated system set-up using an additional video projector for integrating the vertical screen and loudspeaker array of eight speakers forming a cube

The eight speaker cube makes sound adjustment through head movement or change in position possible, giving the user the liberty to move freely within the area of the horizontal plane.

[40] cp.: Section 2.2.3

[41] cp.: Section 2.2.2

5.4. System Configuration

Based on the hardware requirements and the audio reproduction evaluation the system can be composed. The configuration must be adjustable to behave like wide spread of the shelf systems on the one hand but must offer high performance for best 3-D sound impressions on the other hand if needed. The demand for being adjustable can be bypassed configuring two different systems but can easy be implemented using one set-up minimising the costs.

As PC based system can be considered to have the widest divulgence, using a PC layout composition is worthwhile.

Adjusting CPU power to simulate lower performance system can be implemented using a dual processor mainboard featuring two CPUs. Through inactivating one of the CPUs the available processing power is approximately halved but can be maximised at will with no effort.

For graphic visualisation an average graphic card supporting the Nvidia chipset is chosen to release the CPU since it can be assumed that potential users of virtual reality applications tend to have a graphicaly optimised computer system.

For audio reproduction no standard audio card can be used, since there are no of the shelf cards featuring full range eight channel audio reproduction.

The variety of loudspeakers is retrenched through the installed electromagnetic tracking system. Only loudspeakers with minimal influence on the electromagnetic field causing minimal tracking error can be taken into account. Investigations have shown that PC loudspeaker systems caused the lowest influence on the tracking system.

Other components are composed using standard equipment, completing the configuration.

System key features :

Mainboard :	MSI 694D Pro2 – IR
	Supports Dual Socket 370 for Intel® III Processors at
	500MHz~1.13GHz.
CPU :	Genuine Intel X86 Family 6 Model 8, 1.00GHz
RAM :	512MB SDRAM
Hard disk :	1 x 8GB IDE IBM (System disk)
	2 x 40GB IDE IBM ST38421A (data)
Graphic card :	Hercules 3D Prophet II GTS 32MB
	Nvidia GeForce 2 chipset
Audio card :	RME Digi 9652 + RME AEB8-O extension board
	RME Digi 9652 offering 24 digital out audio channels
	RME AEB8-O offering a eight channel DA converter
Speaker :	8 x Wavemaster Blasterbox 240
	240 Watt full range active loudspeakers

Best driver support for the chosen audio and graphic card is offered for Windows98™ second edition. Since a system using only one CPU should be simulated to guaranty the API's usability on common computer systems, Windows98™ is a good choice due to the fact that it is optimised for single processor systems, incapable of using a secondly installed CPU.

5.4.1. System Integration

Implementing the kernel functionality and proving it's usability is the primary goal of the first API development phase. The kernel functionality is designed to perform on standard systems within an integrated virtual reality system as it can be found in most implementations.

Integrating the composed system into the available virtual reality infrastructure is therefore not desirable in a first step to maintain realistic conditions, while setting up stronger restriction on the available CPU resources for 3-D sound at the same time.

For an integrated approach a video projector is attached to the system to take advantage of the Holoscreen's vertical screen and the symmetrical loudspeaker installation.(Fig.24) The virtual reality application as well as the 3-D sound generation is performed on the composed system.

Integration into a distributed virtual reality system is nevertheless applicable.

For integration into the available distributed virtual reality infrastructure the audio subsystem would be accessible over ethernet, offering 3-D sound services for the virtual reality application running on the Onyx2. (Fig.25)[42]

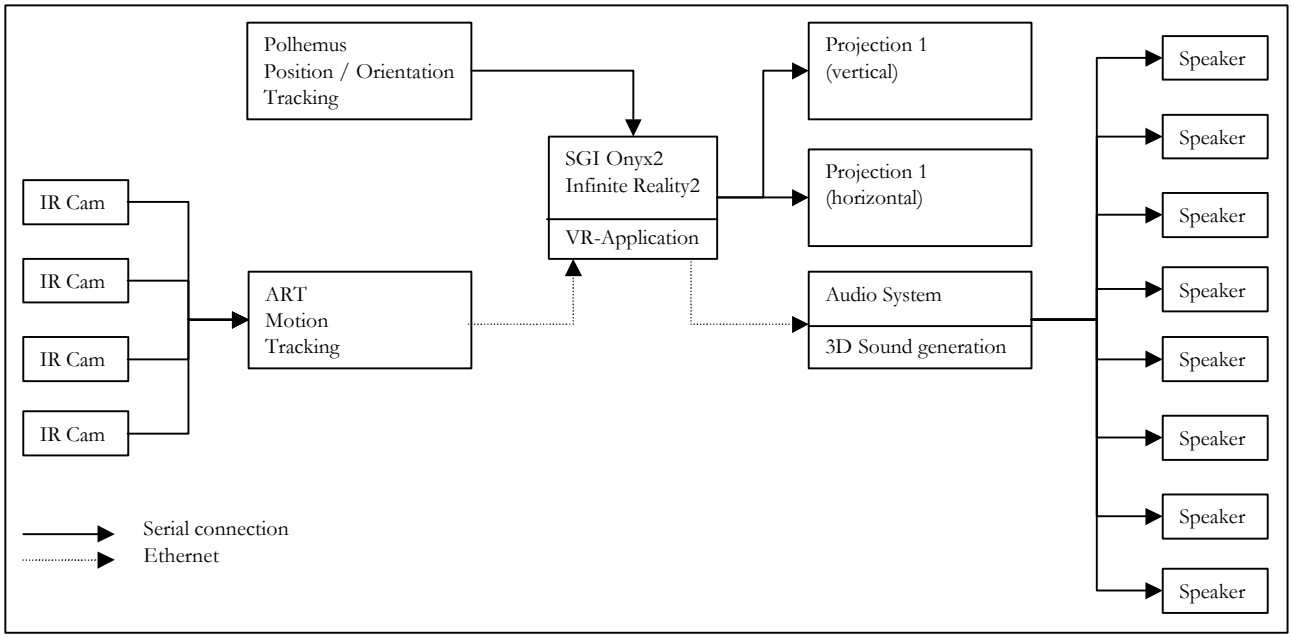

Figure 25: 3-D Sound System integration into a distributed virtual reality system

[42] cp.: Section 8.2

6. API Design

The API design works on the software engineering aspects of the API.
The preparatory work lead to the final coding of the API software realisation.
Basic communication deliberations based on the chosen hardware set-up and the operating system are performed first.
Having determined the API's position within the operating system structure, approaches to realise the real – time 3-D sound requirements are drafted and an existing approach contributing to the realisation is introduced and examined.
Rest upon the contributing approach the data aspects of the 3-D sound implementation are derived from the data catalogue and the sequence of events within a 3-D sound session is introduced.
The API design finishes with the software encoding in 'C'. Special assumptions made within coding are summarised and the major algorithms are explained in the last part.

6.1. Layer model

To guaranty an easy adjustment to varying operating systems and hardware set – ups an integration into the existing operating system, hardware and driver layer model is desirable.

The layer model describes existing modules in the system as well as the communication pipelines between them. The concept of the layer model allows the virtual reality application to be placed on top of the layers thus being independent of any restrictions and specifications except for the 3-D sound API. Additional hardware access (e.g. display devices) is possible but not necessary for audio devices.

Layers are dependent on layers at the same level or at levels below, only. A re – utilisation of modules in other contexts is thereby enabled.

The 3-D sound API is integrated on top of the operating system and hardware layers, below the virtual reality application. (Fig. 26)

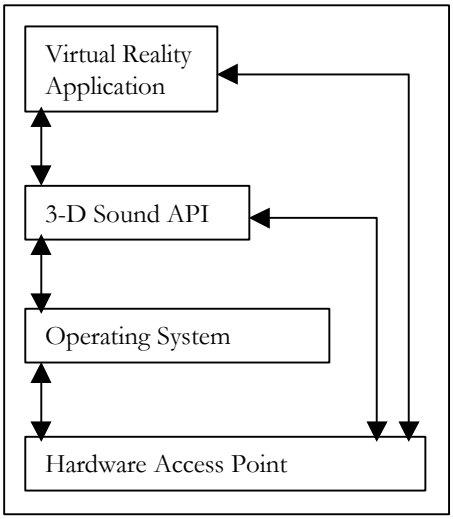

Figure 26:Layer Model 3-D sound integration

6.2. 3-D sound requirement implementation approaches

Basic requirements on a 3-D sound API determined within the 3-D sound requirements[43] are fundamental for further 3-D sound concept advisements.

To free the API from hardware dependencies the use of drivers, is to be implemented within the API implementation.
The API then is restricted to be used with hardware components that have driver support for the actual operating system available. The API implementation can rely on operating system wide access routines to control the hardware, identical for differing audio hardware components. This leads to a pure operating system dependency without hardware specific restrictions.

The operating system dependencies are to be encapsulated within a module as a part of the 3-D sound API. This approach allows easy adjustment for transportation to different systems. The implemented module is used by the 3-D sound API itself as well as it can be used from outside by other software components.
The modular approach within the 3-D sound layer allows further modules necessary for 3-D sound implementation to be added to expedite the API's adjustability and reuse.

[43] cp.: Section 3.1

Combining the modular approach and the layer model leads to a more adjustable and more open layer model.(Fig. 27)

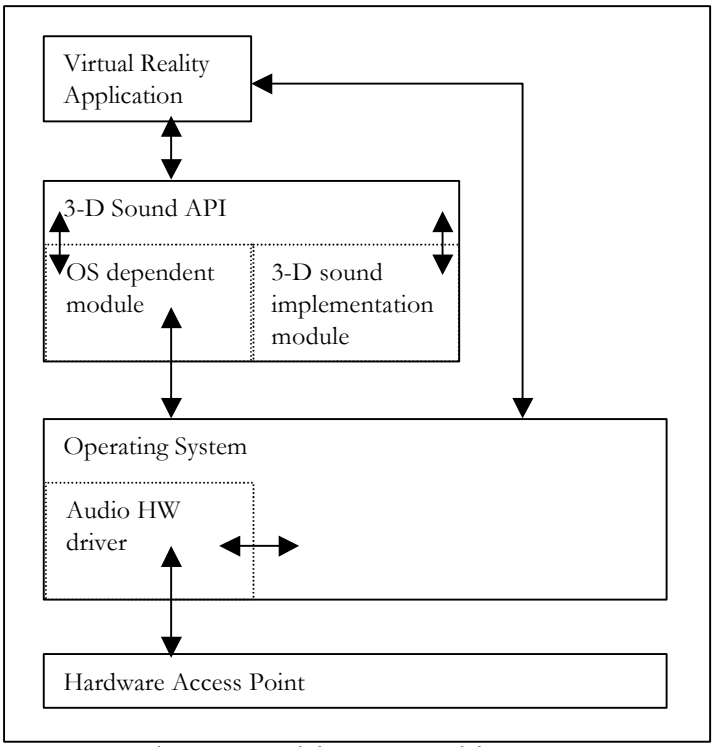

Figure 27: Modular Layer Model

6.3. PortAudio API

Researching available sound APIs and investigating their communication models, an API strictly fulfilling the requirements, restrictions and specifications of the targeted 3-D sound API has been exposed.

PortAudio (Portable Audio Library) is a cross platform, open-source, audio I/O library. It supports writing simple audio programs in 'C' that will compile and run on Windows, Macintosh, Unix(OSS), SGI, and BeOS.[44]

Portaudio can be integrated in a real – time 3-D sound API as a module implementing the hardware access and the operating system dependencies. The real – time 3-D sound API can use PortAudio and can be interpreted as a PortAudio 3-D extension(Fig.28)

(In the following the developed 3-D sound API is declared as PortAudio 3-D extension.)

[44] cp: http://www.portaudio.com, 11.02.2002

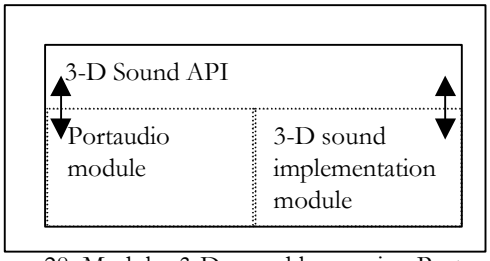

Figure 28: Modular 3-D sound layer using Portaudio

Within PortAudio hardware access is implemented taking hold on ASIO, MME and DirectSound™ for Windows, ASIO and Soundmanager for Macintosh, OSS for Unix and AL for SGI.
To build on Portaudio an analysis of the data structures and processes must be be performed.

6.3.1. PortAudio Analysis

For hardware access PortAudio offers the possibility to initialise streams connected to audio devices for audio output.
The use of several devices in parallel is possible and can be used to perform a multi device audio reproduction.
For controlling the initialised audio device streams, control functions are provided. The use of start and stop functions effectuate the corresponding audio device stream processing to be started or stopped.
PortAudio is using a callback approach to feed the audio device streams with binary audio data. Callback functions are to be defined by developers using PortAudio within their application and are called continuously until stream processing is stopped.
The callback function call is controlled using interrupt technologies or low level processes and is implemented in the PortAudio API.

6.4. PortAudio 3-D Extension Data aspects

Based on the 3-D sound system analysis including the data catalogue, necessary data structures and methods to manage and access these structures are developed.
The necessity of data objects as well as their structure in detail and service functions necessary for conveniently accessing and manipulating these objects are argued.

6.4.1. Data structures

Deriving from the 3-D sound system analysis, essentially three objects are influencing a virtual 3-D sound world.
A listener, the sound source object as well as the graphical object corresponding to the sound source object have to be represented.

The listener object is representing the application user in the virtual audio world. It has to correspond with the virtual graphical representation. The listener object is taking over the graphical representation's properties broadening them through inclusion of audio specific properties.
A sound source is connected to a graphical object and influenced through it's properties like position and orientation. Typically the graphical object is represented in the virtual reality application, thus making a redundant representation of the graphical object unnecessary. Merely the properties influencing the sound source have to be managed within the 3-D sound API. These properties are taken over to be properties of the sound source object itself.
The sound source's properties are additionally enriched through the binary audio data needed for audio playback. For known reasons binary data has to be stored within system memory thus making data structures necessary.
The complexity of binary audio data lets a management structure in terms of a audio buffer object appear suggestive. (Fig.29)

6.4.2. Data Access and Service Routines

Despite the minimal amount of object types, the total amount of objects to be managed can quickly surmount the administerable boundary leading to developer's confusion. Within application progress 1 (listener) + n (source) + m (buffer) objects have to be co-ordinated. Although the amount of objects is reduced with sound sources using the same buffer, service routines are exigent necessary to alleviate the data management through offering a central access point and object manipulating functions.
Introducing a 3-D sound system object representing the central access point reduces the application developer's efforts to managing only one object. Using offered service routines the developer can access and manipulate the listener-, any sound source- and any buffer object through one access point.
In addition to serving as central access point, the 3-D sound system is used to hold user specific environment information regarding the audio reproduction facilities and the user's preferred audio data representation and reproduction.
For modularisation purposes the environmental data is kept in an additional environment object.
The objects' conceptual structures and relations are visualised using OMT notation.
The final implementation does include additional secondary routines and properties not relevant for conceptual purposes therefore not included here[45] (Fig.29)

[45] cp.: Todt, Severin S., PortAudio 3-D Extension Dokumentation, VR4 Lab, Wedel, 2002

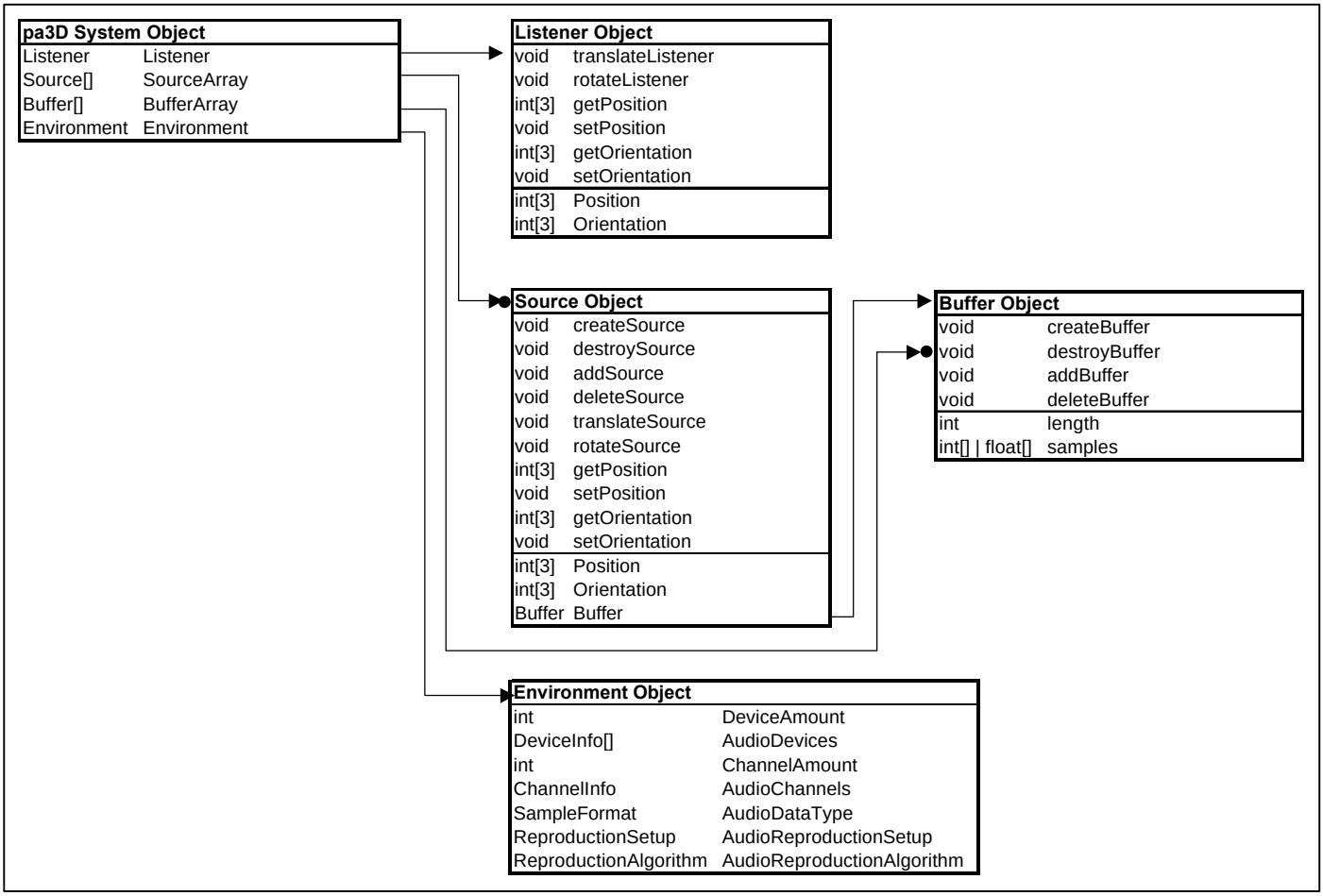

Figure 29: Conceptual OMT Diagram Pa3D System

6.4.3. Listener Object

In every generated 3-D sound system one unique listener object exists.

The listener object is generated within the initialisation phase of the 3-D sound system and is being destroyed with it's termination.

The listener object is coercive necessary over the period of the 3-D sound system's existence since it's representing the user in the virtual auditory world. Hence routines for listener creation and destruction are designed for internal use only and must be protected against application developer access to avoid undefined situations.

For manipulating the listener's properties, further service routines are offered.

The listener object can be referenced using the 3-D sound system object. Manipulation is done using set and get methods.

For listener movement the listener object is to manipulated using translate and rotate functions. The translate and rotate functions' name originate from names used within OpenGL to ease the sound implementation for OpenGL developers.[46]

6.4.4. Source Object

The great amount of possible source objects is managed within the 3-D system object in a sound source list.
Sound source object creation and destruction is done on application developer's demand using the offered service routines to add or remove sound sources to or from the 3-D sound system's sound source list.
To manipulate a selected sound source the sources can be accessed through an access function, delivering the reference to a sound source. For source identification a textual description given by the developer is used.
For sound source movement translate and rotate functions, according to the listener's functions are offered.

6.4.5. Buffer Object

Corresponding to the source objects the buffer objects are managed within a list as part of the 3-D system object.
Buffer objects are created using a creation service routine which is allocating memory and reading the binary audio data from a file. Destruction is done deleting the buffer from the buffer list.
Adding and deleting buffers is done equivalent to the sound sources' proceeding.

---

[46] cp.: Todt, Severin S., PortAudio 3-D Extension Dokumentation, VR4 Lab, Wedel, 2002

6.5. PortAudio 3-D Extension Application

A regular PortAudio (not PortAudio 3-D Extension) application requires the execution of six steps to be processed successively:[47]

1. Writing a callback function that will be called by PortAudio when audio processing is needed.
2. Initialising the PA library and opening a stream for audio I/O.
3. Starting the stream. The callback function will be now be called repeatedly by PA in the background.
4. Within callback data can be written to the output Buffer.
5. Stopping the stream by returning 1 from your callback, or by calling a stop function.
6. Closing the stream and terminating the library

Step 1 and 4 require knowledge about digital audio processing to implement the needed functionality.
To ensure best results, correct processes and minimal effort for the developer the digital processing needs are implemented within the PortAudio 3-D Extension API.
Pure audio play back is based on equal procedures for every participating device functionality, thus enabling an automation.
The developer is totally freed of any digital audio processing knowledge to concentrate on the virtual reality application's demands through applied automation in meanings of offering predefined callback functions for each of the user defined devices.
Using the PortAudio 3-D Extension API, to implement a 3-D sound application, the developer merely needs to perform four steps.

1. Initialising the PortAudio 3-D Extension library
2. Creating sound sources to be reproduced and the corresponding buffers
3. Starting 3-D sound processing
4. Stopping 3-D sound processing

[47] cp.: http://www.portaudio.com, 11.02.2002

6.6. PortAudio 3-D Extension Implementation in 'C'

Since the PortAudio 3-D Extension is based on and uses PortAudio the same programming technique is used for implementation.
The API is composed in ANSI C compilable using a C++ compiler.
Using ANSI C a ceiling of platform independence is reachable. Coding is done taking further differing C type interpretation on different machines into account.
The use of ANSI C makes the conceptual data structures impossible to implement. Objects are not implemented as objects in a object oriented programming manner. Structs holding relevant data are used to implement the data concepts.

To gain a maximum in adaptability thus pandering the transportation to different operating systems, the modular concept is continued within the API implementation. The main functionality, implemented using platform independent code is enclosed in a "pa_3d" module. It includes the service routines as well as the object data structures and the rudiment for initialisation, audio play back and termination.
Initialisation, audio play back and termination need to use operating system and hardware dependent code. The dependent code is captured in separate modules. Operating system dependencies exceeding the dependencies treated with in the Portaudio API are kept in one module. The API's kernel uses threads which are to be implemented dependent on the operating system. Since no other operating system dependency appear within the first approach, the module "pa_3d_thread" keeps hold of the code pieces dependent on the operating system.
Depending on the amount of available devices and channels, the callback functions used for audio play back have to be realised. Hence callback functions are dependent on user specifications realised within the "specifications" module. (Fig.30)

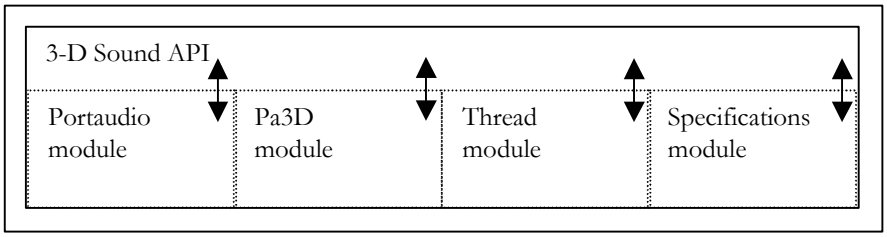

Figure 30: Portaudio 3-D Extension Modules

6.7. 3-D Functionality Algorithms

The Algorithms used within the Portaudio 3-D Extension for initialisation, sound source generation, buffer generation as well as for audio processing can be identified to be the major functions for 3-D sound implementation thus being described in detail. Furthermore the sound source and listener movement are explained in more detail.

6.7.1. Initialisation

The initialisation process determines the available devices within the system and initialises the hardware corresponding to the user defined specifications.
The hardware's data structures and values are stored within the allocated 3-D sound system object.
Further the listener object is allocated and initialised. The listener is determined to be positioned in the virtual world's origin. Within sound processing a listener oriented Cartesian coordinate system is presumed.
Sound source and buffer list are initialised as empty lists.

Initialisation progress in detail :

```
allocate SOUND_SYSTEM_OBJECT
initialise SOUND_SYSTEM_OBJECT.environment with user audio format \
    specifications and device definitions
check available devices
FOR each DEFINED_DEVICE
    get device information from system
    IF DEFINED_DEVICE available
        initialise SOUND_SYSTEM_OBJECT.environment.DEFINED_DEVICE \
            with system device information
        activate DEFINED_DEVICE
        FOREACH DEFINED_DEVICE_CHANNEL
            allocate memory for CHANNEL_BUFFER
```

initialise SOUND_SYSTEM_OBJECT.environment. \
DEFINED_DEVICE.speaker_pos with user specification
ENDFOREACH
ENDFOR
initialise spatialisation.algorithm with user specification
allocate LISTENER_OBJECT within SOUND_SYSTEM_OBJECT
initialise LISTENER_OBJECT.position with (0,0,0)
initialise LISTENER_OBJECT.position with (0,0,-1)
allocate SOURCE_ARRAY within SOUND_SYSTEM_OBJECT
allocate BUFFER_ARRAY within BUFFER_SYSTEM_OBJECT

6.7.2. Sound Source Generation

Sound source generation allocates memory for a sound source object and initialises it with centre position and non iterating play back.
The binary audio data property is initialised using a given reference to a buffer object.

Sound source generation progress in detail :
allocate SOURCE_OBJECT
initialise SOURCE_OBJECT.position with (0,0,0)
initialise SOURCE_OBJECT.position with NO_LOOP
initialise SOURCE_OBJECT.binary_data with given BUFFER_OBJECT reference

6.7.3. Buffer Generation

Buffers are generated allocating memory for a buffer object holding no samples. While reading a given file holding the audio data, memory for storing it within the buffer object is reallocated with every sample being read.

Buffer generation in detail :

allocate BUFFER_OBJECT

initialise BUFFER_OBJECT length with 0

open given audio file

WHILE not end of file

    read 1 sample

    BUFFER_OBJECT.length = BUFFER_OBJECT.length +1

    BUFFER_OBJECT.binary_data =

        BUFFER_OBJECT.binary_data + read sample

ENDWHILE

close file

6.7.4. Audio processing

With starting 3-D processing[48] all device streams are opened and thereby prepared for audio play back. With opening the devices' streams the devices' callback functions are called repetitively through PortAudio's play back routine.
The callback functions implemented within the PortAudio 3-D Extension API are mixing a device's channel buffers down to one, writing into the device stream.
Each of the channel buffers are filled by a process routine which is providing the audio data considering applicable audio effects, the source's position and the user specified audio reproduction technique.
Since the process routine is writing to the channels' buffers and the callback functions is reading the same buffers, the processes have to be synchronised.

Synchronisation is necessary for parallel processing as well as for parallel memory access. For parallel processing the processing routine is running within an own thread. The repetitive call of the callback functions, implemented within the Portaudio API is either controlled using low level processes or interrupts.

[48] cp. Section 6.5

The implementation using low level processes requires both processes to be operated with the same priority for equal distribution of CPU power.

For memory access synchronisation cyclic buffer memory is established for each channel buffer. With the start of 3-D sound processing the processing thread is filling each of the channels' cyclic buffers. In a second phase the mechanism for callback function calls can be started. The callback functions are reading as many samples as it takes to fill a device stream buffer.

With every execution of the processing thread the cyclic buffers are refilled if necessary. To synchronise the read and write mechanism, the callback functions and the processing routine keep track of their current position within the cyclic buffer. Whenever the processing routine's position is "behind" the callback's position the cyclic buffer has to be refilled. (Fig. 31)

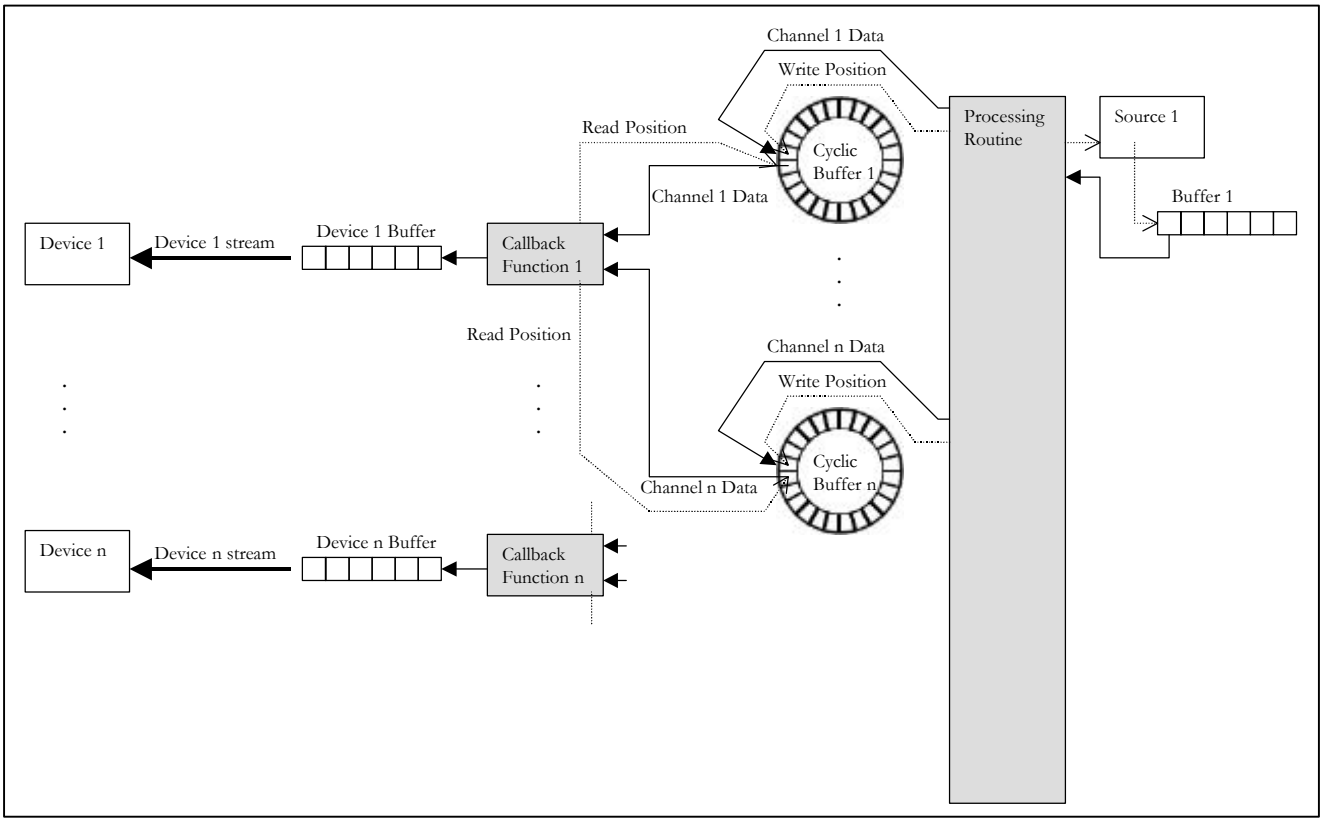

Figure 31: Memory access synchronisation

```
Callback in detail :
FOR frames_per_Buffer
    FOR samples_per_frame
        read CHANNEL_BUFFER at read_position
    ENDFOR
ENDFOR
read_position = read_position + frames_per_buffer
IF read_position >= cyclic buffer size
    read_position = cyclic buffer start
ENDIF

Process routine in detail :
WHILE 3-D processing not stopped
    IF (FOREACH device write_position <= read_position)
        FOREACH SOURCE_OBJECT
            IF enough samples left to read in SOURCE_OBJECT.binary_data to fill \
                device buffer
                FOREACH device
                    FOREACH device.channel
                        IF device.channel is active
                            call reproduction function to work up size of device buffer \
                                samples
                        ELSEIF
                            fill up device buffer with '0'
                        ENDIF
                    ENDFOREACH
                    device.channel.read_position=read_position +
                        frames_per_buffer
                    IF device.read_position >=cyclic buffer size
                        device.read_position = cyclic buffer start
                    ENDIF
                ENDFOREACH
            ELSEIF //not enough sample left in buffer
```

```
FOREACH device
    FOREACH device.channel
        IF device.channel is active
            call reproduction function to work up sample left in buffer
        ELSEIF
            write '0' to device buffer as often as samples left in buffer
        ENDIF
    ENDFOREACH
    device.channel.read_position=read_position +
        written samples
    IF device.read_position >=cyclic buffer size
        device.read_position = cyclic buffer start
    ENDIF
    IF SOURCE_OBJECT to loop repetitive
        SOURCE_OBJECT.buffer_read_position= start of  \
            corresponding BUFFER_OBJECT.binary data
        FOREACH device.channel
            IF device.channel is active
                call reproduction function to work up samples needed \
                    to fill  up device buffer
            ELSEIF
                write '0' to device buffer as often as samples needed \
                    to fill up  device buffer
            ENDIF
        ENDFOREACH
        device.channel.read_position=  read_position +
            needed samples to up device buffer
        IF device.read_position >= cyclic buffer size
            device.read_position = cyclic buffer start
        ENDIF
```

```
                    ELSEIF // no loop
                        FOREACH device.channel
                            write '0' to device buffer as often as samples needed to
                                fill up device buffer
                            device.channel.read_position= read_position +needed
                                samples to up device buffer
                            IF device.read_position >= cyclic buffer size
                                device.read_position = cyclic buffer start
                        ENDFOREACH
                    ENDIF
                ENDFOREACH
            ENDIF
        ENDFOREACH
    ENDIF
ENDWHILE
```

Reproduction function in detail :

The reproduction function being called within the process routine is reading a given length of a given source object's buffer into a temporary buffer.
Effects defined for the source object are applied to the temporary buffer.
Having all effects applied, the algorithm for sound spatialisation, dependent on the chosen reproduction technique, is used on the temporary buffer.
The resulting buffer is then written to the device stream.

```
create tempbuffer with length of device.buffer size
FOR device.buffer size
    read sample from SOURE_OBJECT.binary_data to tempbuffer
ENFOR
IF effects defined for SOURCE_OBJECT
    FOREACH SOURCE_OBJECT.effect
        apply effect algorithm to tempbuffer
    ENDFOREACH
ENDIF
```

IF SOURCE_OBJECT is to be positioned

    apply specified spatialisation algorithm to tempbuffer

ENDIF

write tempbuffer to device buffer

Sound effects in detail :

With the kernel implementation of the Portaudio 3-D Extension no sound effects are supported. Functions and control structures necessary to implement sound effects are integrated within the audio reproduction process nevertheless.

Thus broadening the reproduction with sound effects step by step is possible.

Implementing sound effects like distance attenuation, doppler shift, diffraction or reverberation contribute to the impression's reality.

Spatialisation algorithm in detail :

For the chosen audio reproduction set-up an algorithm for reproduction using a loudspeaker array must be chosen.

For improved performance the Ambisonics reproduction technique has been selected.[49]

Using amplitude panning techniques a sound signal is applied to all loudspeakers placed evenly around the listener.

The loudspeakers gain factor is evaluated using the total number of speakers and the angle between the loudspeakers and the virtual sound position.

$$g_i = \frac{1}{N} (1 + 2\cos\alpha_i)$$

g_i : gain of loudspeaker I

N : total number of loudspeakers

α : angle between virtual sound source signal and loudspeaker

The reproduction can be approved using second order Ambisonics.

$$g_i = \frac{1}{N} (1 + 2\cos\alpha_i + 1 + 2\cos 2\alpha_i)$$

[49] cp.: Section 2.2.2

The sound is still applied to all of the loudspeakers but the gains have prominently lower absolute values on the opposite side of a panning direction. Thus fewer artefacts appear. [50]

The Ambisonics reproduction technique leads to the spatialisation algorithm used within the Portaudio 3-D Extension.

The loudspeaker amount as well as their positions are given within the environment property of the sound system object. For sound sources the position is given in their properties allowing the determination of the needed operands and the gain to be applied.

6.8. Moving objects

Within the virtual world the listener object and the source objects can be replaced using the offered service routines for translation and rotation.

Within the Portaudio 3-D Extension a listener oriented Cartesian coordinate system is assumed. Thus the listener position always stays origin of the world.

Translating or rotating a source is done corresponding to translation and rotation algorithms used in computer graphics.

Listener translations or rotations have to be performed in a different way.

Since a listener oriented coordinate system is used, translating or rotating the listener is implemented as a transformation of the coordinate system.

Not the listener is changing with a transformation being applied every source within the virtual world is changing it's position when a transformation is applied to the listener object.

---

[50] cp. :Pulkki, Ville, Spatial Sound Generation and Perception by Amplitude Panning Techniques, Helsinki, Laboratory of Acoustics an Audio Signal Processing - Helsinki University of Technology, Espoo 2001, Report 62, 2001, pp.14-15

7. Application Test

For proving the workability and the rightness of the API's kernel approach a demo virtual reality application is developed including real – time 3-D sound.
To prove the API's functionality a sound source and the listener object are included within the application.
For audio play back purposes the audio reproduction set-up at the Fachhochschule Wedel is used.
Although the installed distributed virtual reality system at the virtual reality laboratory benefits a demonstration of a distributed approach, the the demo application is developed using the composed integrated virtual reality system. Using an integrated system, the API's workability in high stress situations is proven.

7.1. Virtual Reality Application Implementation

To demonstrate PortAudio 3-D Extension's functionality a simple OpenGL based virtual reality application is developed. The application is implementing a sphere placed on a plane within the virtual world.
The camera is representing the user's view.
The application user is free to move within the virtual world. Translation as well as rotation in every direction is possible. With listener movement every possible relative position between the sphere and the listener is produceable.[51]
For 3-D sound inclusion, the sphere is to be interpreted as an object emitting omni directional sound. The camera representing the user is to be taken as the listener's representation as well it is already representing the user's visual aspects.
Using the PortAudio 3-D Extension, the sound inclusion is done in four steps as mentioned.[52]

---

[51] cp.: Todt, Severin S., PortAudio 3-D Extension Demoprogramm Dokumentation, VR4 Lab, Wedel, 2002

[52] cp.: Section 6.5

The first three steps are executed within the initialisation phase of the virtual reality application.

1. Initialising the PortAudio 3-D Extension library
2. Creating sound sources to be reproduced and the corresponding buffers
3. Starting 3-D Sound processing

Moving the listener is done within OpenGL's callback function through user input application.
The graphical transformations are taken over for listener transformation resulting in corresponding situations within the virtual graphical as well as in the virtual audible world.

When terminating the graphical output the 3-D sound is stopped also.

4. Stopping 3-D Sound processing

For successful compilation the "pa_rd.h" header file has to be included and the libraries needed by PortAudio to realise the operating system dependent tasks has to be linked.

7.2. Evaluation of Integratability

Within 3-D sound integration the application developer is freed of any knowledge about digital audio processing, neither he has to care about audio file input or output. Writing an virtual reality application including 3-D sound is comfortable and easy.

Although only four steps have to be performed and the developer is freed of a lot of background knowledge he still has to take into account the synchronisation mechanisms used within the Portaudio 3-D Extension.
The first two steps of 3-D sound integration are uncritical. They can be performed practically anywhere in the virtual reality application.
Starting the 3-D sound processing and starting and the virtual reality application processing has to be done in separate threads using higher priority for the audio processing thread to avoid interruption or audible artefacts during playback.
Knowledge of multithreaded programming is therefore essential.

Since the API's specification module could not be finished within this thesis, the work to be done in advance to enable an ease of use is uncomfortable and lengthy.
Although the specification module is already prepared, encapsulating the relevant information in two header and source files, the developer still has to edit these files in order to adjust the specification to his audio reproduction set-up.

Choosing virtual reality applications other than these based on OpenGL might be an easier idea for interpreting 3-D sound at the moment.
Since OpenGL is working on a stack based transformation system, the applied graphical transformations can not be transferred as they are.
Applying a rotation followed by a translation in OpenGL in the same order using the PortAudio 3-D Extension API would result in different situations.
Within OpenGL the rotation is performed based on the position resulting from the translation.
The PortAudio 3-D Extension API does not work on a transformation stack instead applying the transformation directly.
Thus the final position is evaluated as the translation of the coordinates that result from the rotation.
The transformations must therefore be performed in opposite order which is easy for small OpenGL application but can be become impossible for mighty ones.

7.3. Evaluation of Application Execution

The demo application showed that the integration of real – time 3-D sound integration is possible even on an integrated virtual reality system approach.
The workability of the PortAudio 3-D extension API is proven with the demo application.
It could be shown that sound reproduction corresponding to a visual counterpart within the virtual reality application can be performed without noticeable delay in real - time.
The reproduction is possible for objects placed anywhere in 3-D space.

It could further be proven that the routines for translating and rotating objects are resulting in changes corresponding to the common understanding of the functionality. Performance investigation is done comparing the frame rate reachable by the OpenGL application with and without 3-D sound integration.
Integrating PortAudio 3-D Extension API calls, the frame rate lowers from approximately 30 Frames per second to a rate as low eight frames per second for the demo application running on the installed hardware set-up.
The virtual reality application's performance is decreasing with 3-D sound integration but no artefacts were recognisable on the 3-D sound application part during audio play back. The Portaudio 3-D Extension API's influence on CPU power distribution can be adjusted through change of the thread priority the 3-D sound functions are started and operated in.
Downgrading the thread's priority is not desirable since an interruption of the 3-D sound processing would result in audible artefacts.

8. Conclusion and Future Work

8.1. Summary

A real – time 3-D sound API kernel was developed and tested based on a standard hardware platform and a loudspeaker array for audio reproduction.
The development and testing was performed in the virtual reality laboratory at the Fachhochschule Wedel .
An overview over related research projects (e.g. Virtual reality Sound at the University of Aizu, Cyber Stage at GMD, NAVE at Georgia Tech) was presented. Based on those the idea for the 3-D sound system was developed, exposing their restrictions and features.
The background of human sound perception was introduced and techniques for spatial sound reproduction were presented and evaluated. Methods for 3-D sound integration in available virtual reality systems were shown and discussed.

For the software engineering process minimum requirements on a real – time 3-D sound API were composed and existing 3-D sound APIs were checked against these, ratifying the need of a new real – time 3-D sound API development.
A stand alone virtual reality system housing the virtual reality application and the real – time 3-D sound application were set up based on standard hardware components and a loudspeaker array for play back.
The kernel functionality of a real –time 3-D sound reproduction system was engineered based on a extensive analysis of the natural sound system.
The reproduction using a loudspeaker array of eight loudspeakers and Ambisonics for sound spatialisation, positioning of sound sources and listener movement are forming the kernel functionality.
The API was tested using an OpenGL application supplemented with real – time 3-D sound functionality.

8.2. Future Work

The developed real – time 3-D sound API will be developed further in a subsequent project.
Specifications module will be enlarged and an application for specifying user dependent configurations for audio processing and the available hardware set-up will be developed to ease the API's use.
An additional module realising the integration within a distributed virtual reality system will be developed. The real – time 3-D sound functionality will then be implemented using client and server stubs. The system used for audio playback in this thesis will be available over ethernet within the virtual reality laboratory as 3-D sound server.
For easier integration in virtual reality application the transformation mechanisms for translating or rotating objects will be adapted to the use of a transformation stack as performed in OpenGL.
Further audio reproduction techniques for loudspeaker arrays as well as for headphone reproduction will be supported for a wider field of use.

Sound effects, not implemented yet, will be developed to increase the grade of presence and reality of the 3-D sound reproduction. Effects like Doppler Shift, distance attenuation or reverberation will improve the natural impression.
Continuously the source code will be optimised for increase of performance.

References

Begault, Durand R. : 3-D sound For Virtual Reality And Multimedia, Moffet Field, NASA Ames Research Center, Academics Press Professional, 1994, Reprint 2000

Begault, Durand R. : Auditory And Non-Auditory Factors That Potentially Influence Virtual Acoustic Imagery, Moffet Field, Human Factors and Technology Devision - NASA Amees Research Center, 1999

Begault, Durand R. : Head-Up Auditory Displays for Traffic Collision Avoidance System Advisories, Moffet Field, NASA Ames Research Center, The Human Factors and Ergonomics Society Inc., 1993

Begault, Durand R. : Preferred Sound Intensity Increase For Sensation Of Half Distance, Moffet Field, AES 93rd Convention San Francisco, 1992

Begault, Durand R., Wenzel, Elisabeth M.: Headphone Localization of Speech, Moffet Field, NASA Ames Research Center, Human Factors, 1993

Davis, Elisabeth T. a.o. : Can Audio Enhance Visual Perception and Performance in a Virtual Environment, Atlanta, Georgia Institute of Technology, 1999

Dickreiter, Michael: Handbuch der Tonstudiotechnik, München, Saur, K.G, 1997

Faust, Simon: 3-D Akustik mit Lautsprechern, Aachen, RWTH Aachen Virtual Reality and 3D Visualization, 2000

Gardner, William G.: 3-D Audio Using Loudspeakers, Boston, Massachusetts Institute of Technology, Kluwer Academic Publishers, Norwell, Massachusetts, 1998

Groening, Matt : The Call of the Simpsons, Twentieth Century Fox, 1990

Hennig, Alexander, Die andere Wirklichkeit, München, Addison Wesley, 1996

Herder, Jens : A Sound Spatialisation Resource Management Framework, Aizu, The University of Aizu - Spatial Media Group, 1999

IASIG, IASIG Interactive 3D Audio (Level 1) Requirements, 1998

Jot, Jean-Marc, Synthesising Three-Dimensional Sound Scenes in Audio or Multimedia Production and Interactive Human – Computer Interfaces, Ircam - Centre Georges-Pompidou, 5th International Conference, Interface to Real & Virtual Worlds, Montpellier, 1996

Kendall, Gary : A 3-D sound Primer, http://www.northwestern.edu/musicschool/classes/3D/pages/3DsoundPrimer.html, 11.02.2002

Loki Software™, OpenAL Specification and Reference, http://www.openal.org, 2000

Microsoft™, MSDN Library – DirectX, http://www.microsoft.com/directx/, 11.02.2002

Pulkki, Ville, Spatial Sound Generation and Perception by Amplitude Panning Techniques, Helsinki, Laboratory of Acoustics an Audio Signal Processing - Helsinki University of Technology, Espoo 2001, Report 62, 2001

Sun microsystems™, The Java 3D API specification, http://java.sun.com/, 11.02.2002

Todt, Severin S., PortAudio 3-D Extension Demoprogramm Dokumentation, VR4 Lab, Wedel, 2002

Todt, Severin S., PortAudio 3-D Extension Dokumentation, VR4 Lab, Wedel, 2002

Tonnesen, Cindy; Steinmetz, Joe : 3-D sound Synthesis, Washington, The Encyclopedia of Virtual Environments, http://www.hitl.washington.edu/scivw/EVE/I.B.1.3DSoundSynthesis.html, 1993

http://wwwsv1.u-aizu.ac.jp/~mcohen/spatial-media/PSFC/welcome.html, 11.02.2002

http://www.gmd.de/, 11.02.2002

http://www.portaudio.com, 11.02.2002

Appendix A

CD Contents

The CD ROM included within this diploma thesis contains literature available in misc. formats.
The literature contributed to the work as well as it can be used for future research projects.

The diploma thesis and the referenced implementation documentation written by the author are included within the “diploma_thesis_doc” directory.

Sourcefiles including the PortAudio API sources, PortAudio demo programs as well as the PortAudio 3-D Extension can be found with in the “sources” directory.
The PortAudioExtension API can be found under “sources/portaudio/pa_common/pa_3d“, the referred demo programm is placed in the directory ”source/portaudio/pa_common/pa_3d”

CD ROM directory path

/literature (contains literature that contributed to the diploma thesis)

- /aachen
- /aes
- /aizu
- /ames
- /dolby
- /gerogia_tech
- /hut
- /iasig
- /icad
- /lake_products
- /mit_medialab
- /openal
- /princeton

/diploma_thesis_doc

/sources

- /portaudio
  - /docs
  - /download
  - /pa_common
    - /pa_3d
  - /pa_mac
  - /pa_tests
  - /pa_unix_oss
  - /pa_win_ds
  - /pa_win_wmme
  - /pablio

Eidesstatliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Ort, Datum Unterschrift (Vor- und Nachname)