Final Degree Project

Bachelor’s degree in Industrial Technology Engineering

Incorporation of three-dimensional audio into virtual reality scenes for an improved immersive experience

REPORT

Author: Henriksson Martí, Eric
Director: Susín Sánchez, Toni
Deposit date: September 2017

Barcelona School of Industrial Engineering
Abstract

Audio is a crucial aspect to bear in mind when designing virtual reality applications, as it can add a whole new level of immersion to this kind of experiences if properly used. In order to create realistic sound, it is essential to take audio spatialization into consideration, providing the information necessary for an individual to estimate the position of sound sources and the characteristics of surrounding spaces. This project proposes implementing spatial audio in virtual reality scenes created with a game engine, as well as providing all of the theoretical bases that explain how this can be ultimately achieved.

It first touches upon how the human auditory system is able to estimate the direction and distance to an audio source by interpreting cues such as time and level differences between ears, pinnae reflections, reverberation and general variations in loudness.

Next, the limited spatial properties present in the most common audio reproduction systems are discussed, arguing why they are insufficient for virtual reality applications. Two spatial audio recording and reproduction techniques for headphones and loudspeakers are presented as alternatives for virtual reality scenarios in which the user remains static.

As a means of acquiring the knowledge necessary to understand more advanced spatial audio systems, the concept known as Head Related Transfer Function or HRTF is introduced in great detail. It is explained how HRTFs encompass all physical cues that condition sound localization, as well as how the frequency responses that characterize them can be experimentally measured and used for artificial spatialization of virtual sources.

Several HRTF-based spatial audio systems are presented, differentiating between those that apply HRTFs as mathematical models and those that make use of experimental impulse response data sets. These advanced models are the way to go if spatial audio is to be applied to virtual reality experiences that involve user motion, as they are capable of constantly adapting to the user’s position and direction relative to the present virtual sources.

The rest of the project focuses on how some of the mentioned HRTF-based spatial audio systems can be implemented in the Unity game engine. The poor built-in spatialization options the main software offers can be complemented and greatly improved with the use of audio plugins that perform HRTF filtering and introduce features such as sound occlusion, room simulation models and sound directivity patterns.

Three demos with different levels of complexity are finally carried out in Unity in order to showcase the virtues of spatial audio in virtual reality applications.
Index

ABSTRACT .......................................................................................................................... 1
INDEX ................................................................................................................................. 3
1 GLOSSARY ....................................................................................................................... 7
2 PREFACE .......................................................................................................................... 10
   2.1 Origins of the project ................................................................................................. 10
   2.2 Motivation ................................................................................................................ 10
3 INTRODUCTION ................................................................................................................ 11
   3.1 Objectives of the project ........................................................................................... 11
   3.2 Scope of the project ................................................................................................... 11
4 SOUND LOCALIZATION .................................................................................................... 12
   4.1 Localization of sources and the human auditory system ........................................... 12
   4.2 Psychoacoustics ........................................................................................................ 12
      4.2.1 Coordinate systems ............................................................................................ 12
      4.2.2 Azimuth cues and lateral localization ................................................................. 13
         Interaural Time Difference ....................................................................................... 13
         Interaural Level Difference .................................................................................... 13
         The complementarity of ITD and ILD ..................................................................... 14
         Localization accuracy ............................................................................................ 14
         The cone of confusion and front/back ambiguities ............................................... 15
      4.2.3 Elevation cues and vertical localization ................................................................ 16
      4.2.4 Range cues and localization ............................................................................... 16
         Loudness .................................................................................................................. 17
         Motion parallax ........................................................................................................ 17
         Excess interaural level difference ........................................................................... 17
         Ratio of direct to reverberant sound ....................................................................... 17
   4.2.5 Reverberation and echoes ..................................................................................... 18
5 SPATIAL PROPERTIES IN AUDIO REPRODUCTION ...................................................... 19
   5.1 Common audio reproduction methods and systems ............................................... 19
      5.1.1 Monaural audio .................................................................................................... 19
      5.1.2 Two-channel stereo systems .............................................................................. 19
      5.1.3 Surround sound systems .................................................................................... 20
5.2 Notable spatial audio recording and reproduction methods ......................... 21
   5.2.1 Binaural recording ........................................................................... 21
   5.2.2 Ambisonics ..................................................................................... 22

6 HEAD-RELATED TRANSFER FUNCTION ............................................. 24
   6.1 HRIR ................................................................................................... 25
      6.1.1 Horizontal plane ........................................................................... 25
      6.1.2 Median plane .............................................................................. 26
   6.2 HRTF .................................................................................................. 27
      6.2.1 Horizontal plane ........................................................................... 27
      6.2.2 Median plane .............................................................................. 28

7 HRTF IMPLEMENTATION IN AUDIO SYSTEMS .............................. 29
   7.1 Listening devices ................................................................................ 29
      7.1.1 Headphones ................................................................................ 29
         Strengths and weaknesses ................................................................. 29
         Head tracking .................................................................................. 29
      7.1.2 Loudspeakers ............................................................................... 30
         Crosstalk cancelled stereo ............................................................... 30
         Strengths and weaknesses ................................................................. 31
   7.2 Modelled and measured HRTFs ....................................................... 31
      7.2.1 Modelled HRTFs ....................................................................... 32
         ITD model ....................................................................................... 32
         ILD model ....................................................................................... 33
         ITD + ILD model ............................................................................... 33
         Pinnae model ................................................................................... 35
         Complex combined models ............................................................. 35
      7.2.2 Measured HRTFs ........................................................................ 36
         The Convolvotron ............................................................................ 36
         Ambisonics + binaural rendering systems ........................................ 38

8 INCORPORATION OF SPATIAL AUDIO IN UNITY ...................... 40
   8.1 Unity’s built-in spatialization features ............................................... 40
   8.2 Unity’s native audio plugin SDK ....................................................... 42
      8.2.1 Audio spatializer SDK ................................................................. 42
         Convolution of functions or signals .................................................. 45
         Using the audio spatializer plugin in Unity ....................................... 47
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience

8.3 Google VR's SDK for Unity ................................................................. 49
  8.3.1 Google VR's spatial audio support .............................................. 49
  Using Google VR's spatial audio support in Unity .............................. 58
  Use recommendations ........................................................................ 60
8.4 Comparison chart ........................................................................... 61

9 SPATIAL AUDIO DEMOS IN UNITY .................................................. 62
  9.1 Demo 1: Direct HRTF test ................................................................. 62
    9.1.1 Concept .................................................................................... 62
    9.1.2 Spatial audio features ............................................................... 62
    9.1.3 Plugins and hardware ............................................................... 62
    9.1.4 Operation ................................................................................. 63
  9.2 Demo 2: Office cinematic scene ..................................................... 63
    9.2.1 Concept .................................................................................... 63
    9.2.2 Spatial audio features ............................................................... 64
    9.2.3 Plugins and hardware ............................................................... 64
    9.2.4 Operation ................................................................................. 65
  9.3 Demo 3: Interactive VR acoustic room ........................................... 65
    9.3.1 Concept .................................................................................... 65
    9.3.2 Spatial audio features ............................................................... 66
    9.3.3 Plugins and hardware ............................................................... 67
    9.3.4 Operation ................................................................................. 67

10 PROJECT BUDGET .................................................................................. 70
  10.1 Time dedication ............................................................................... 70
  10.2 Software licenses ............................................................................ 70
  10.3 Equipment ...................................................................................... 71
  10.4 Tuition fees .................................................................................... 71
  10.5 Total cost of the project ................................................................. 72

CONCLUSIONS .............................................................................................. 74

ACKNOWLEDGEMENTS ............................................................................. 77

BIBLIOGRAPHY ............................................................................................... 78
  Bibliographical references .................................................................... 78
  Complementary bibliography ............................................................... 79
1 Glossary

- **Ambisonics**: Surround sound technique that enables advanced three-dimensional audio reproduction.
- **Azimuth**: Angular measurement in a spherical coordinate system.
- **B-format**: Format used by the Ambisonics technique to represent a sound field.
- **Binaural hearing**: Hearing based on two ears that allows determining the direction and origin of sounds.
- **Binaural recording**: Two-channel recording method that recreates human binaural hearing.
- **C#**: Multi-paradigm programming language.
- **C++**: General-purpose programming language.
- **Cone of confusion**: Sound localization ambiguity caused by a certain disposition of audio sources relative to the listener.
- **Convolution**: Function derived from two other functions by integration that determines how the shape of one is affected by the other.
- **Crossfading**: Gradually increase the level of a sound as another is decreased.
- **Crosstalk-cancelled stereo**: Technique that allows binaural audio to be reproduced through two loudspeakers.
- **Dummy head recording**: Binaural recording technique.
- **Duplex theory**: Theory that explains how humans are able to localize sounds by interpreting time and level differences between the ears.
- **Early reflections**: First sound reflections that reach the listener and provide information about the surrounding space.
- **Echoes**: Consciously noticeable sound reflections.
- **Elevation**: Angular measurement in a spherical coordinate system.
- **Fast Fourier Transform**: Algorithm that converts a signal to the frequency domain.
- **Game engine**: Software framework meant for the development of video games and simulations.
- **Head Related Impulse Response**: Set of impulse responses that capture all of the cues that determine human sound localization.
• **Head Related Transfer Function**: Fourier transform of the HRIR.

• **Head-shadow effect**: Reduction of the amplitude of sound in one ear due to the sound obstruction caused by the head.

• **HRIR**: Abbreviation for Head Related Impulse Response.

• **HRTF**: Abbreviation for Head Related Transfer Function.

• **ILD**: Abbreviation for Interaural Level Difference.

• **Interaural Level Difference**: Difference in sound level between the listener’s two ears.

• **Interaural Time Difference**: Difference in arrival time of a sound between the listener’s two ears.

• **ITD**: Abbreviation for Interaural Time Difference.

• **Monaural audio**: Sound based on a single channel.

• **Mono**: Shortened form of monaural.

• **Motion parallax**: Cue that determines how changes in source localization perception caused by head motion are dependent on the distance to the source.

• **Native plugin**: Library of native code from which plugins allow a software to call functions.

• **Occlusion**: Frequency-dependent level decrease in incoming sound caused by an obstacle between an audio source and the listener.

• **Panning**: Controlled distribution of a sound signal into a stereo or multi-channel sound field.

• **Phantom source**: Imaginary sound source as a result of synthetic audio spatialization.

• **Pinna**: Visible part of the ear that resides in the external part of the head.

• **Precedence effect**: Conscious perception of direct sounds and early reflections as a single fused auditory image.

• **Psychoacoustics**: Study of the perception of sound.

• **Reflections**: Parts of a sound that have bounced off a surface before reaching the listener.

• **Reverberation**: Build up of a large number of reflections that result in a persistence of sound after an initial emission.
• **Room simulation model**: Acoustic simulation of an enclosed space achieved by replicating reflections and reverberation effects.

• **SDK**: Abbreviation for Software Development Kit.

• **Signal-to-noise ratio**: Relation between the sound level of a signal and the level of undesired noise.

• **Sound field source**: Audio source without a specific origin that solely depends on direction.

• **Stereo**: Method of audio reproduction usually based on two channels that creates the impression of surrounding the listener.

• **Surround sound**: Method of audio reproduction based on three or more loudspeakers surrounding the listener.

• **Unity**: Cross-platform game engine developed by Unity Technologies.

• **VR**: Abbreviation for virtual reality.
2 Preface

2.1 Origins of the project

Virtual reality, also known as VR, has quickly become one of the main topics of interest in current society. It is a fast-growing branch of modern technology and is considered by many to be the future of multimedia. Consequently, huge investments are being made by numerous companies; healthcare, education and military institutions; and many others in order for it to prosper and reach its peak performance.

The term virtual reality usually leads us to think of interactive three-dimensional visual experiences, but it is much more complex than that. As we are all aware of, most of us experience reality through our five senses: sight, hearing, touch, taste and smell. Therefore, if reality was to be replicated perfectly in a virtual world, all of the mentioned senses should be taken into account. Current virtual reality systems are a mere approximation of this ideal situation. They are for the most part based on the visual aspect of the experience, and are usually accompanied by audio support and sometimes very basic haptic feedback. The predominance of the visual part of VR has led it to reach a much higher technical complexity than the rest of the other four senses.

This project aims to focus on the audio facet of virtual reality. Even though audio is clearly second in line after visuals when it comes to complexity in current systems, it definitely still has a lot of room for improvement. Sound quality has been addressed and improved in great measure since the invention of the first recording device in the mid-19th century, but certain other audio characteristics are yet to be properly exploited. The most notorious of these characteristics is probably sound spatialization, which will be the focal point of this project.

2.2 Motivation

Despite not having coursed any audio-related studies throughout my entire academic formation, sound has always been a passion of mine. Back in 2012, I carried out a project regarding binaural audio recording, the simplest method for maintaining the spatial properties of natural sound when recorded.

This has been the perfect opportunity to take this basic knowledge to the next level. In spite of being a very effective technique, binaural recording presents certain limitations that make it practically unusable for interactive virtual reality applications. Therefore, the main challenge of this project has been finding a suitable alternative to provide VR experiences with spatial audio in the most accessible way possible.
3 Introduction

3.1 Objectives of the project

The purpose of this project has been to find a way to improve the audio-related aspect of virtual reality scenarios, consequently increasing the level of immersion within these experiences. The implementation of sound spatialization has been established as the main priority to accomplish this, and will be the primary aspect covered throughout the report. In order to achieve the mentioned objective, the following points have been set up:

• Explain how sound localization is performed by the human auditory system.
• Examine the spatialization capabilities of the most common audio reproduction methods and systems.
• Research and list the different methodologies advanced spatial audio can be attained with, understand their principles and determine the advantages and disadvantages for each of them. Select the most favourable options for virtual reality applications.
• Find a way to implement the selected methods in the Unity game engine, taking head motion into consideration.
• Consider which hardware devices the used software should be compatible with and adapt it to them.
• Implement any other audio characteristics that might help increase the quality of the resulting virtual reality experiences.
• Document the entire process mentioned in the previous phases as accurately as possible.

3.2 Scope of the project

As anticipated in the objectives, this project is mainly based around the implementation of spatial audio in the Unity game engine, and ultimately in virtual reality scenarios. Therefore, for the most part, it revolves around the use and adaptation of plugins and software development kits that complement the game engine so that advanced sound spatialization and other audio properties can be incorporated effectively. The operation of these tools is described in detail throughout the report, as well as the theoretical basis behind each of them. In addition, the project eventually comprises the creation of three different demos in Unity, in which the spatialization of audio plays a key role.
4 Sound localization

4.1 Localization of sources and the human auditory system

Sound localization refers to our capacity to locate the origin of any audio source within our three-dimensional surroundings depending on the properties of an incoming sound. We are able to estimate the direction and the distance to any source rather precisely. Localization is something we experience all the time on a daily basis. For instance, when we drop a set of keys, we are immediately able to tell they hit the ground beneath us just by hearing them collide with it; if a car is approaching us from behind, we are also capable of approximating its direction and distance from us without the need to use our sight; etc.

This might seem like a very simple and natural concept, but it functions in a more complex way than what we could expect at first glance. If humans only have a couple of ears, how are we capable of locating incoming sounds in a three-dimensional space? In theory, this should be impossible to achieve with just two channels, but certain factors that compose what is known as psychoacoustics make it attainable.

4.2 Psychoacoustics

4.2.1 Coordinate systems

It is essential to work with a coordinate system to specify the location of an audio source relative to a listener. Due to the roughly round nature of the human head, spherical coordinate systems are the most appropriate choice. These systems use azimuth (θ), elevation (ϕ) and distance to the source as coordinates. The vertical-polar coordinate system is the most frequently used one. It first measures the azimuth as the angle from the median plane to a vertical plane containing the source and the z-axis. Then, it determines the elevation as the angle up from the horizontal plane.

![Fig. 4.1 Vertical-polar coordinates system.](image)
4.2.2 Azimuth cues and lateral localization

Sound localization on the horizontal plane functions based on the Duplex Theory. This theory states that there are two different cues for azimuth: Interaural Time Difference (ITD) and Interaural Level Difference (ILD).

**Interaural Time Difference**

Considering the shape of a human head to be spherical, Interaural Time Difference or ITD can be explained as follows. If an incoming wave from a distant source travelling at the speed of sound $c = 343 \text{ m/s}$ hits a head with a radius of $a$ at an azimuth angle of $\theta$, the sound wave arrives to each ear at different times. Obviously, the sound reaches the right ear earlier than the left one. This is due to the fact that it has to travel an extra distance of $a \cdot \sin(\theta) + a \cdot \theta$ to get to the second. The ITD is simply obtained by dividing this distance by the speed of sound:

$$ITD \approx \frac{a}{c} \cdot (\sin(\theta) + \theta), \quad -\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$$

(Equation 4.1)

As it can be expected, the ITD is non-existent when the audio source is right in front of the listener and it reaches its maximum value when the source is located completely off to one of the sides. This maximum value is of about 0.7 ms for an average human head.

**Interaural Level Difference**

In addition to the described temporal difference, a notable sound level or volume difference is also present between each of the ears. The head diffracts sound waves as soon as they hit it, consequently altering the resulting sound pressure at each ear. This phenomenon is known as the head-shadow effect. In the depicted situation, the sound level at the right ear is higher than at the left one, as the left ear is in the sound shadow of the head.

The ILD greatly depends on the frequency of the sound wave. At high frequencies, the level difference between ears can be of about 20 dB or more. On the other hand, at low frequencies, the sound pressure is almost the same at both ears, due to the longer wavelength in relation to the head diameter.
The complementarity of ITD and ILD

According to the Duplex Theory, the ITD and ILD complement each other in the following way:

- **Low frequencies**: For frequencies lower than ~800 Hz, information obtained from the ILD is very poor, whereas the ITD is surely noticeable. At these low frequencies, the sound’s half wavelengths are longer than the dimensions of an average human head, making phase or time delays very clear.

- **High frequencies**: For frequencies above ~1500 Hz, the length of the waveforms is shorter than the human head’s dimensions. Therefore, the ITD is no longer reliable. On the contrary, the ILD becomes larger, and serves as the main lateral localization cue.

- **Transitional frequency zone**: For frequencies between ~800 Hz and ~1500 Hz, both interaural time and level differences have an active role in sound localization.

Localization accuracy

Human beings can perceive interaural time differences of as little as 10 ms in optimal conditions. Therefore, sound localization accuracy can be as precise as 1 degree for sources located right in front of the listener. However, for audio sources located on the sides, this accuracy decreases to about 15 degrees.
The cone of confusion and front/back ambiguities

The described ITD and ILD can be ambiguous in certain aspects. For audio sources located along the circumferences of any circular cross-sections of a cone, such as the one shown below, time and level differences are nearly the same. The listener is incapable of perceiving any lateral localization differences between sound waves coming from any of these points. Following the same principle, it is also difficult for the listener to discern between audio sources located in front or behind him/her.

![Cone of confusion](image1)

Fig. 4.4 Cone of confusion.

These ambiguities can be easily resolved with some slight head motion. As soon the listener turns his/her head, the time and level differences change in opposite directions for each ear, making localization clear again. In the example of the figure below, the listener is first experiencing a location ambiguity and cannot tell if the audio source is in front or behind him/her, due to the ITD and ILD being exactly the same in the two depicted possibilities. By rotating his/her head slightly to the right, the listener now notices how the sound reaches the left ear earlier than the right and/or how the level is higher in the left ear, and is consequently able to identify the true location of the audio source.

![Resolution of location ambiguities](image2)

Fig. 4.5 Resolution of location ambiguities with head motion.
4.2.3 **Elevation cues and vertical localization**

Elevation, contrary to lateral localization, is determined by monaural cues instead of binaural ones. The outer visible part of the ear, known as pinna, is responsible for adding the necessary information to incoming sound waves for humans to interpret elevation efficiently. The geometry of the outer ear leads to the amplification of certain frequencies and the attenuation of others. This frequency response is highly dependent on the direction of the incoming sound.

The figure to the right presents an example with two different directions of arrival, with their respective frequency responses.

As the image tries to illustrate, part of the incoming sound wave reaches the inner ear following an almost direct path, whereas the rest of it reflects off the pinna before doing so. At low frequencies, the pinna simply absorbs part of the sound’s energy and lets the different paths reach the inner ear at the same time. On the other hand, at high frequencies, the signal that reflects is out of phase with the signal that follows a more direct path, resulting in interference. This interference is highest when the difference between path lengths is about half of the wavelength, and produces a pinna notch that usually sits within the 6 kHz to 16 kHz range. It can be appreciated at a frequency of about 10 kHz in the figure.

The pinna notch is always more pronounced for sounds coming from the front than from above, due to the pinna’s higher reflectivity from this direction. Additionally, the difference between path lengths varies with elevation, so the frequency of the notch does so as well.

4.2.4 **Range cues and localization**

Human sound localization capabilities are not as precise for range determination as for the previously explained azimuth and elevation estimations. Range is interpreted by the listener using the following aspects: Loudness, motion parallax, excess ILD and ratio of direct to reverberant sound.
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience

Loudness

The most intuitive cue for sound range localization is probably loudness. As we are all aware of, the sound level or energy we perceive from a certain audio source decreases as we move away from it, and vice versa. It does so falling off inversely with the square of range [1].

\[ p \approx \frac{1}{r^2}; \quad p = \text{sound pressure} \]

\[ r = \text{range} \]

(Equation 4.2)

Loudness by itself, however, is sometimes not enough to provide the listener with any effective range cues at all. If all audio sources emitted a constant sound level and were familiar to the individual, the relation between loudness and range would be enough for range localization, but this is certainly not the case in the real world in many cases. Sound energy that reaches the listener is proportional to the energy emitted by the audio source. Therefore, if loudness is to be used as a cue for range, some characteristics of the source must be known. For instance, we are generally able to approximate the distance that separates us from another person that is talking, as we are familiar with the average sound level of normal human speech and its general qualities. The use of loudness as a cue for range is, ultimately, an ability that is developed and perfected with subconscious learning for the most part.

Motion parallax

The concept of motion parallax basically establishes that the perception of changes in azimuth caused by head motion is dependent on range. For nearby sources, moving the head slightly results in a large change in azimuth, whereas for distant ones, the azimuth appears to stay practically the same.

Excess interaural level difference

As the range between the listener and an audio source decreases, the ILD becomes more prominent, especially within distances of about one meter or less. Sounds that are almost completely focused on just one of the ears, such as when someone whispers in one of our ears, tend to be uncomfortable to listen to. This kind of sounds should be approached with care when designing audio, especially if it is meant for headphone use. It is never necessary to place all reproduced sound in one of the ears and leave the other one in silence, even if we intend to make the listener feel as if the source is on one side and really close.

Ratio of direct to reverberant sound

In most real world scenarios, the previously described relationship between loudness decay and range is not that simple. Sound reflects and scatters all the time from all the different
surfaces that surround us. A considerable part of the sound we receive comes from these reflections, to a greater or lesser extent, depending on the characteristics of the environment. For example, if the listener finds him/herself in an enclosed room with moderately reflective walls, emitted sound within the room gets reflected multiple times around the surrounding space, and the energy that reaches the ears becomes almost independent from the distance between the listener and the source. That being said, any acoustic scenario has a certain ratio of direct to reverberant sound, which acts as a major cue for range determination. At short distances, this ratio is usually quite high, while at long ranges it tends to present a low value.

The first reflected waves that reach the listener’s ears are known as early reflections. These usually provide enough information for the listener to be able to estimate the size and shape of his/her surroundings.

4.2.5 Reverberation and echoes

As mentioned in the previous point, our surroundings are full of all sorts of reflective surfaces of different characteristics, and a great deal of the sound we perceive comes from environmental reflections, even in locations out in the open. However, we are not consciously aware of a major part of these reflections, only those in which the delay surpasses a threshold of about 30 to 50 ms. This threshold is called the echo threshold, and marks the point from which we start to consciously notice them, calling them echoes.

Surprisingly enough, reflections do not interfere with human sound localization capability. Our mind learns and adapts to new acoustic environments constantly. As a result of this learning process, our auditory system manages to suppress the negative effects of reverberation in most cases. We are able to locate sounds just by taking into account the signals that first reach our ears. This is known as the precedence effect. Still, as previously explained, we also make a subconscious use of the information that reflections provide us with when it comes to range estimation.

In a common acoustic scenario with some reverberation, and for medium to high frequencies, reflections usually reach the listener some milliseconds after the original sound. On the other hand, when the sound’s period is longer than the time it takes for the reflections to arrive, the situation is quite different. For these low frequencies, at about ~250 Hz or below, the sound reflections start to arrive to the listener before one full cycle is even completed. This makes it impossible for our auditory system to estimate ITDs, and consequently, localization cannot be accomplished. In general, high-frequency sounds are the responsible ones for providing the cues for source localization in reverberant spaces.
5 Spatial properties in audio reproduction

As simple and natural as sound localization is in our perception of the real world; it is a characteristic that is barely ever applied to any audio reproduction systems in a sufficiently realistic way. Almost all recorded or artificial audio we have access to is one or two-dimensional. What this means is that the localization information it provides the user with is non-existent or limited to somewhat of a 2D plane.

5.1 Common audio reproduction methods and systems

5.1.1 Monaural audio

Mono is the most basic and primitive form of audio. As its name implies, it simply consists of sound that is recorded or played through one individual audio channel. This method does not provide the listener with any effective spatialization cues at all.

Monaural audio can be reproduced with any sort of setup, but playing it through more than a single speaker is pointless for any purpose other than a better spread.

5.1.2 Two-channel stereo systems

Most current audio setups are based on two channels, a left one and a right one. This allows sound to be distributed between the two, in equal or different levels, creating a phantom source that can be virtually placed anywhere in between them. This way, the listener can get the impression that the audio is coming from a straight direction or from a certain angle. This technique is known as crossfading, and is present in the majority of music files and multimedia in general. Simple crossfading, however, limits the positioning of the phantom source to the line segment between the speakers. In order to be able to position the virtual source outside of this line, a technique referred to as crosstalk-cancelled stereo is used, which will be explained later in the report.

The movement of the phantom source can also be accomplished by taking advantage of the precedence effect. If the sound that is played in one of the speakers is delayed a few milliseconds relative to the sound played in the other speaker, the phantom source will appear to come from the second. It is important not to overdo this delay, as it could end up in the listener being able to consciously notice it as an undesired echo.

Two-channel stereo systems are meant to be implemented with a couple loudspeakers or standard headphones, which makes them very versatile.
5.1.3 Surround sound systems

Surround sound audio is designed to be played through an array of multiple loudspeakers (a minimum of three). The idea behind these systems is to assign individual channels or loudspeakers to different desired directions. They are usually named after the number of standard small loudspeakers and subwoofers they incorporate, separated by a dot (e.g., 7.1).

Sound localization in surround sound systems depends completely on the amount of speakers that compose the system, their type and how they are arranged. The small loudspeakers take care of the medium-high frequencies, and are therefore responsible for sound localization. The subwoofers, on the other hand, play the non-directional low frequencies, so their position is not critical.

There are no boundaries that define what the maximum amount of speakers for this kind of systems is; possibilities are endless. However, surround sound systems tend to be expensive and require quite some space, making them a non-viable option in many cases.
5.2 Notable spatial audio recording and reproduction methods

5.2.1 Binaural recording

In the earlier section regarding the localization capabilities of the human auditory system, it has been determined how we are able to locate sound sources only by listening to sounds with our two ears, employing our binaural hearing. Following the same principle, it is also possible to record and reproduce three-dimensional audio using just a couple of channels.

The most notorious and accessible technique for recording 3D sound with only two channels, keeping most of its spatial properties, is known as binaural recording. This method is based on the recreation of the sound pressures that the listener would experience in each of his/her ears in a certain situation. In order to successfully achieve such a replication, a specific technique referred to as dummy head recording is normally used due to its simplicity.

An acoustic mannequin head fitted with a microphone inside of each ear canal records the desired audio, and the resulting recording is listened to using headphones. If the dummy head’s dimensions and reflectivity properties are close enough to the ones of the listener, the recorded ITD and ILD will be almost identical to the ones the listener would experience naturally. Likewise, if the size and shape of the mannequin’s pinnae are similar to the listener’s, the elevation cues will also be the appropriate. The recreation of all of these audio localization cues provide the listener with an incredibly realistic three-dimensional sound experience, almost as if being present in the original recorded sound field.

Fig. 5.3 The binaural audio recorded with an acoustic dummy head (right) is later listened to with headphones (left). The listener experiences an illusion that provides a very accurate 3D recreation of the originally recorded audio scenario.
Regardless of the unquestionable effectiveness of binaural recordings, it must be noted that they present some significant disadvantages:

- Due to the variability in the size and shape of the pinna in different individuals, vertical sound localization can sometimes be a bit inaccurate. The pinnae of the dummy head’s ears can be customized for a particular individual, but this is a laborious and costly procedure and is ultimately not feasible for a larger public. Instead, the pinnae of these acoustic heads are usually created with average sizes and shapes.

- They must be prerecorded. It is not possible to apply binaural recordings to interactive experiences, as the audio files have already been previously recorded and are not modifiable from that point onwards.

- As soon as the listener moves, the immersive effect is lost. The audio that is being reproduced does not adapt to any movements whatsoever, which is a huge inconvenient in potential virtual reality applications, amongst others.

- The use of headphones is indispensable. Playing binaural recordings through any kind of loudspeakers eliminates all of its spatial properties.

### 5.2.2 Ambisonics

Ambisonics is a surround sound technique for recording and reproducing spatial audio, developed in the early 1970s. In contrast to other surround sound systems, this technique is based on source directions instead of loudspeaker positions. This allows the number and the layout of the loudspeakers to be quite flexible.

In order to work with source directions, Ambisonics uses what is known as the B-format. This format is basically a loudspeaker-independent representation of a sound field. The transmission channels for systems that use this technique carry this B-format, which is later transcoded to function with the used output setup. It is possible to transcode the B-format for the output in numerous different ways, from a monaural source to multi-channel surround sound arrays. In addition, the panning, tilting, positioning and rotation of virtual sources can all be modified in post processing.

The simplest form of the B-format, used in first-order Ambisonics, consists of a matrixed four-channel signal. These four channels are labelled as follows:

- $W$: Omni-directional sound pressure
- $X$: Front minus back sound pressure gradient
- $Y$: Left minus right sound pressure gradient
- $Z$: Up minus down sound pressure gradient
First-order B-format surround signals can either be artificially created by manipulating monaural sources with software, or recorded with a specific microphone composed of a tetrahedral array of capsules very close to each other.

In short, the general purpose of Ambisonics is that of allowing almost any loudspeaker array to replicate the spatial sound effects that a particular audio source in space, either created artificially or recorded, produces on a listener.

The main disadvantage when using Ambisonics is the fact that the effectiveness of its spatial effects is limited to a certain area, referred to as the sweet spot. This can be a problem when applying it to virtual reality scenarios in which the user is meant to move freely in a relatively extensive space.

First-order Ambisonics usually present quite poor spatial resolution and a rather small sweet spot. Both the spatial resolution and the size of the sweet spot can be gradually improved by increasing the order of the used Ambisonics, or in other words, by adding more components to the B-format. The amount of signal components $a$ required for a particular $n$-order Ambisonics is established in the following way:

- Horizontal-exclusive systems: $a = 2n + 1$
- Full-sphere systems: $a = (n + 1)^2$

Obviously, using more complex loudspeaker arrays can also increase the resolution of the resulting spatial audio.
6  Head-Related Transfer Function

In order to determine the sound pressure that an audio source produces at each of the listener's eardrums, it is first necessary to know the impulse response $h(t)$ that it causes in them. This response is known as Head-Related Impulse Response or HRIR, although it is more common to refer to its Fourier transform $H(f)$: the Head-Related Transfer Function or HRTF. The HRTF encompasses all of the physical cues that condition the localization of sound. If the HRTF is known for both ears, it is possible to convert any mono audio source positioned in a certain point in space to a set of two binaural signals that provide the according localization information to the listener.

The HRTF is a very complex function of frequency and the three spatial coordinates. When working with spherical coordinates, sources located further than one meter away from the receiver are considered to be in what is referred to as the far field. From this distance and beyond, the HRTF attenuates inversely with range. Most measurements are carried out in this far field, where the HRTF is reduced to just three variables: $H(f, \theta, \phi)$.

When measuring HRTFs, it is advisable to perform the procedure in an anechoic chamber so that the influence of early reflections and reverberation is minimized. It is also important for the generated impulses to be of high volume, to maximize the signal-to-noise ratio. The measurements are usually done at small azimuth increments of as little as 5°, up to 30°, for different elevation angles, and interpolation is used to synthesise the gaps in between these points.

![Diagram of HRTF measurement](image)

Fig. 6.1 Example of a possible set-up for HRTF measurement.
For the purpose of getting a better understanding of what kind of results are obtained when measuring HRTFs, we are going to be examining the measurements carried out by the University of California in Davis using an acoustic dummy head named KEMAR (Knowles Electronics Manikin for Auditory Research) [2]. More specifically, we are going to be focusing on the HRIR and HRTF evaluations for the horizontal and the median planes.

### 6.1 HRIR

#### 6.1.1 Horizontal plane

The following graph displays the impulse response of the mannequin’s right ear to an audio source moving around the head in the horizontal plane.

![Fig. 6.2 Graph showing the HRIR at KEMAR’s right ear for sources located in the horizontal plane.](image)

It can be observed how the time of arrival is the earliest and the strength of the response is the highest when the source is located at the right side \((\theta = 90^\circ)\). Contrarily, the sound arrives latest and weakest when the source is at the left side \((\theta = 270^\circ)\). The evolution of the arrival time for different azimuths actually seems to follow the sinusoidal ITD equation established earlier in the report (Equation 4.1) quite accurately.

The initial changing response intensity bands (alternating yellow and black lines) appear due to the reflections in the pinna. Shoulder reflections can also be spotted at about 0.4 ms after the initial peaks.

---

1 An online version of this full database is available at the Boston University EarLab’s website [3]. It also includes measurements for 45 different subjects apart from the KEMAR mannequin.
It is also worth mentioning how similar the responses for the front and back audio source positions are. This explains why humans find it difficult to distinguish between sources located in these positions without using head motion. However, differences do exist, and can be discerned as an absence of ideal symmetry about a horizontal line at $\theta = 90^\circ$.

### 6.1.2 Median plane

This second graph below shows KEMAR’s impulse response to an audio source moving around it in the median plane. The results here are a lot more subtle than those of the horizontal plane, which gives a hint of why humans are better at lateral sound localization than at estimating elevation.

![Graph showing KEMAR's impulse response for sources located in the median plane.](image)

Fig. 6.3 Graph showing KEMAR’s impulse response for sources located in the median plane.

The arrival time appears to be approximately the same for every elevation angle of the source, which is to be expected. The only differentiating traits occur in the relative arrival times and sound strengths that originate from the reflections in the pinna. These changes are quite hard to spot in this HRIR representation, but become much clearer in the HRTF graph shown later in this section.

Once again, the responses for the front and back positions of the audio source seem very similar, but a lack of perfect symmetry is still present about a horizontal line at $\phi = 90^\circ$. 

6.2 HRTF

6.2.1 Horizontal plane

The next graph displays the frequency response of the mannequin’s right ear to an audio source moving around the head in the horizontal plane.

Just as seen in the corresponding HRIR graph, the strength of the response is the highest when the audio source is positioned completely to the right (θ = 90°), and lowest when it is located on the other side (θ = 270°).

The similarity between front and back configurations is still noticeable in this HRTF representation. When examining the responses for these two source positions in greater detail, it becomes apparent that the one for the back positioning of the source is slightly lower than the one for the front configuration in the frequency range between ~4 Hz and ~8 kHz.

The pinna notch described in section 4 can also be clearly seen at a frequency of around 10 kHz in this particular graph.
6.2.2 Median plane

The following final graph presents KEMAR's frequency response to an audio source moving around it in the median plane.

For all possible different elevation angles to the source, the response at around 4 kHz stays nearly the same. This area corresponds to broad ear canal resonances.

The frequency at which the previously mentioned pinna notch appears depends on the elevation angle. At low elevations, it sits somewhere around 6 kHz, and it moves up to 10 kHz or more as the source elevation increases. There are certain elevations for which the frequency response can be practically flat, making the notch almost unnoticeable. This is usually the case for sources positioned right above the listener.

It is important to remind that the described characteristics for the frequency response are extremely dependent on the pinna size and shape, and can therefore present some substantial differences between different individuals. Due to these variability issues, implementing effective sound localization when it comes to elevation is a lot more complicated than doing so with azimuth.
7 HRTF implementation in audio systems

7.1 Listening devices

The first decision that needs to be faced when designing an audio system that is meant to incorporate HRTF is to determine whether the audio is going to be played through headphones or loudspeakers. Each of these two alternatives presents a set of advantages and disadvantages that must be taken into account when making this choice.

7.1.1 Headphones

Strengths and weaknesses

The use of headphones results in a wide range of conveniences:

- They simplify the issue of distributing sound to each ear individually, without the problem of one channel interfering with the other at all.
- The applied HRTFs are very accurate, as they do not suffer from additional unwanted HRTF effects. In other words, the fact that the speakers are placed in the user’s ears prevents the body geometry of the user him/herself from modifying the sound any further.
- A greater degree of acoustic isolation from the listener’s surroundings can be achieved, improving realism and immersion.
- The played audio is free from undesired reverberation and echoes.

However, the use of headphones as listening devices also presents certain disadvantages:

- The speakers are situated very close to the listener, which sometimes makes the reproduced sounds seem too close.
- Headphones can include notches and peaks in their frequency responses that can interfere with the pinna responses and compromise elevation effects if not addressed.
- It can become uncomfortable to wear headphones during long periods of time.

Head tracking

If a listener who is using headphones rotates or translates his/her head, the headphones obviously move the same way. Therefore, if the audio sent to the ears remains the same, the spatial effects follow the user’s motion and become useless. This is totally unacceptable for virtual reality applications, in which the user is allowed to move freely, but it is an issue that can be corrected with head tracking.
A device known as head tracker keeps track of the position and orientation of the head at all times. This information allows the relative position of the virtual audio sources to be recalculated constantly so that the HRIRs can be modified appropriately. It is important to bear in mind the following two aspects when using head tracking:

- **Transients.** The transition between different switching HRIRs should be as smooth as possible to prevent any clicking sounds. This can be accomplished with crossfading techniques.
- **Latency.** In order to avoid noticeable amounts of lag, the time between when a head movement occurs and the HRIR is corrected should not be longer than 50 ms.

### 7.1.2 Loudspeakers

Before establishing the advantages and disadvantages of loudspeaker use for HRTF implementation, it is necessary to explain how it is made to work.

**Crosstalk cancelled stereo**

If binaural audio is to be played through loudspeakers in an effective way, a technique called crosstalk cancelation needs to be used.

In the figure to the right, two loudspeakers are controlled by signals $S_1$ and $S_2$. The signals $Y_1$ and $Y_2$ that reach each of the ears are the resulting combination of the original signal from the speaker on their side and the crosstalk signal of the other speaker, taking into consideration their respective HRTFs ($H_{ij}$):

\[
Y_1 = H_{11}S_1 + H_{12}S_2 \quad \quad Y_2 = H_{21}S_1 + H_{22}S_2
\]

(Equation 7.1)

To establish the $S_1$ and $S_2$ speaker signals that are needed to achieve the wanted $Y_1$ and $Y_2$ signals in the listener's ears, the previous equations can be inverted:

\[
\begin{bmatrix}
Y_1 \\
Y_2
\end{bmatrix} = \begin{bmatrix}
H_{11} & H_{12} \\
H_{21} & H_{22}
\end{bmatrix} \begin{bmatrix}
S_1 \\
S_2
\end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix}
S_1 \\
S_2
\end{bmatrix} = \begin{bmatrix}
H_{11} & H_{12} \\
H_{21} & H_{22}
\end{bmatrix}^{-1} \begin{bmatrix}
Y_1 \\
Y_2
\end{bmatrix}
\]

(Equation 7.2)

Crosstalk cancelled stereo can be reasonably effective at creating spatialization effects for both azimuth and elevation, as it is able to locate phantom sources at a considerable distance away from the line between the two loudspeakers if done right.
Strengths and weaknesses

Using loudspeakers for HRTF implementation basically presents the two following advantages over the use of headphones:

- The sound generated with loudspeakers generally feels more natural to the user due to the speakers not being as close to the listener and the fact that it is not necessary to wear any sort of device.
- They usually do not present irregularities such as notches and peaks in their frequency response as pronounced as those typically found in headphones.

On the other hand, the use of loudspeakers also presents some major weaknesses:

- The listener can only experience accurate localization effects if positioned in a particular sweet spot; the location considered when inverting the previously described equations (Equation 7.2).
- Sound localization effects tend to be less accurate than those accomplished with headphones.
- The user is not acoustically isolated from his/her surroundings, which can negatively affect the level of realism and immersion of the experience.
- Room effects like reverberation and echoes can alter the sound field in an undesired manner.

7.2 Modelled and measured HRTFs

HRTFs can be implemented in an audio system as models or by using experimentally measured data such as the explained in section 6.

**Modelled HRTFs**

- **HRTF Model.** Model containing parameters adaptable to the user.
  - **Sole standard HRTF.** The use of a single standard set of HRTF measures belonging to a particular body geometry. It is the most straightforward and inexpensive approach, but it can lead to somewhat poor elevation effects for part of the user base.

**Measured HRTFs**

- **Set of standard HRTFs.** Using several HRTF measures from different distinct people, allowing the final user to select the one that fits him/her best.
  - **Customized HRTF.** Measuring the user’s own HRTF. It gives excellent results but is also a laborious and expensive procedure.
7.2.1 Modelled HRTFs

Several approaches to HRTF modelling have been proposed by researchers over time. The most intuitional ones are probably the structural models, which intend to design transfer functions that replicate the effects of physical phenomena such as the head-shadow effect. Some of these structural models are presented below, arranged from lower to higher complexity.

**ITD model**

The following model is simply meant to adjust the ITD experienced by the user in response to the current azimuth angle \( \theta \). It does so by adding different time delays to each ear that depend on this angle.

\[
T_d(\theta + \frac{\pi}{2}) \quad \text{and} \quad T_d(\theta - \frac{\pi}{2})
\]

If the assumption of a spherical approximation of the head shape is made, like the one considered when determining the ITD equation in section 4, the time delays are established with the following function:

\[
T_d(\varphi) = \begin{cases} 
\frac{a}{c} \cdot (1 - \cos(\varphi)), & \text{if } |\varphi| \leq \frac{\pi}{2} \\
\frac{a}{c} \cdot (|\varphi| + 1 - \frac{\pi}{2}), & \text{if } \frac{\pi}{2} < |\varphi| \leq \pi \end{cases}
\]

\( a = \text{head radius} \quad \frac{a}{c} = \text{speed of sound} \quad (\text{Equation 7.3})\)

This model obviously lacks many of the features that comprise a complete spatial audio experience. It does not differentiate between front and back directions to the source and does not provide any elevation, externalization or head-shadow effects at all. In addition, the fact that the ILD is null creates a conflicting cue that can lead to the listener hearing an undesired simultaneous second source located in the centre.
ILD model

The next model is focused on adjusting the ILD between the ears in relation to the azimuth angle $\theta$. It achieves this by applying a fairly simple approximated transfer function that mimics the head-shadow effect with some decent accuracy.

The mentioned transfer function $H(s, \varphi)$ includes one pole and one zero, and looks like this:

$$H(s, \varphi) = \frac{(1 + \cos(\varphi)) \cdot s + \frac{2c}{a}}{s + \frac{2c}{a}}; \quad a = \text{head radius} \quad c = \text{speed of sound}$$  \hspace{1cm} (Equation 7.4)

This function increases high frequencies for virtual audio sources located in front of the listener, and limits them for those located somewhere behind.

Just like the previous model, the ILD structural model is also very primitive and lacks some important features needed in a proper spatial audio experience. It does not discern between front and back positionings of the source and is only effective for a certain frequency range. Externalization effects such as room reverberation and echoes are also non-existing in this model. Additionally, the inaccurate time delays it applies to each ear create a conflicting localization cue that once again leads to the listener getting the impression of hearing an undesired simultaneous second audio source.

ITD + ILD model

In order to get rid of the conflicting cues that cause the previously described double-source problem in the individual models above, these ITD and ILD spherical head models can simply be joined creating an improved larger structural model that takes both ITD and ILD into account.
This combined ITD + ILD model still does not incorporate any kind of elevation or externalization effects. However, some basic externalization effects can be achieved with rather simple modifications. The simplest of these modifications could be to add a general constant echo effect for both ears, independent of the direction to the audio source.

\[ T_d(\theta + \frac{\pi}{2}) \]

\[ T_{\text{echo}} \]

\[ K_{\text{echo}} \]

\[ H(s, \theta + \frac{\pi}{2}) \]

\[ H(s, \theta - \frac{\pi}{2}) \]

\[ T_d(\theta - \frac{\pi}{2}) \]

*Fig. 7.4 ITD + ILD structural model, including simple echo effect.*

As seen in the figure above, the echo effect is implemented with a couple of gain and delay blocks, \( K_{\text{echo}} \) and \( T_{\text{echo}} \) respectively. For the externalization effect to work properly, and create the impression of a natural room echo, the values for these two filters should be situated within the following intervals:

\[ 0 \leq K_{\text{echo}} \leq 1 \]
\[ 10 \text{ ms} \leq T_{\text{echo}} \leq 30 \text{ ms} \]

(Equation 7.5)
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience

Pinnae model

When it comes to creating a model to represent the effects the pinna has on incoming sound, there is no general consensus on how to do so. Some researchers have proposed replicating the reflection effects on the different areas of the pinna with multiple echoes [4].

![Diagram of Pinnae Model]

Each of the depicted gains and delays would vary according to the elevation and azimuth angles to the audio source. Establishing these relations is a very complex process and perhaps the principal weakness of this structural model. Besides, determining parameter values for a specific listener’s pinnae is a procedure that has not really been optimized yet.

It has been proved that the values of the time delays are considerably more decisive than those of the gains, and that decent elevation effects can be accomplished with just \( n = 6 \) echo branches for each pinna [5].

Complex combined models

If all of the previous models are combined into a single more complex structural model, the resulting system acquires the individual functional qualities of each of these simpler models and becomes a much better approximation of a real HRTF.
Other structural models representing extra physical mechanisms, such as room effects, shoulder reflections, torso diffraction, ear canal resonance, etc., can also be added into this larger combined model, improving it gradually.

7.2.2 Measured HRTFs

Due to the highly complex nature of HRTFs, most spatial audio systems opt to implement them using experimentally measured data instead of adjustable models. Relying on measured data makes HRTFs a lot simpler to accomplish, but as explained earlier in the report, it also limits the adaptation capabilities to different subjects with distinct body geometries.

The Convolvotron

The device known as Convolvotron, developed by Cristal River Engineering in the late 1980s, introduced what is arguably one of the most effective concepts for achieving HRTF audio using measured data up to date. This apparatus basically consists of a couple of convolution
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience

Engines, each able to convolve\(^2\) a monaural input signal with part of a HRIR contained in a table of experimentally measured values. The HRIR values are selected depending on the real-time azimuth and elevation angles to the source, which are usually determined with head tracking. The amount of impulse response values stored in the tables is obviously limited to a certain number of directions; the rest of missing values for other directions are established by interpolating between the closest existing data points. This necessity of constantly performing high quality interpolations between different HRIRs is probably the Convolvotron’s biggest disadvantage, as it usually involves a rather high CPU usage.

An accurate set of localization cues can be attained as long as the measured HRIRs used by the convolution engines are similar to those of the user.

Range effects can be added modifying the amplitude of the signal in relation to the distance from the source to the ears. In addition, a room simulation model can be introduced in order to reproduce reverberation and echo effects.

It is only possible to obtain additional audio sources by summing the outputs to each of the listener’s ears from several systems like the one depicted above. Therefore, using this sort of system to create acoustic experiences that involve many audio sources and reflections might not be the best option, due to the high CPU usage it can entail.

\(^2\) The concept of signal convolution is explained later in the report, on page 45.
Ambisonics + binaural rendering systems

Some systems provide an alternative to the concept behind the Convolvotron, with the purpose of avoiding the mentioned need to constantly perform interpolations between measured impulse responses. This alternative consists essentially in encoding signals into the Ambisonic domain so that time-independent HRTF filters can be used, resulting in a more computationally efficient system.

Fig. 7.8 Example: Ambisonics + binaural rendering system, as proposed by the Institute of Electronic Music and Acoustics in Graz [6]. It includes a simple room simulation model comprised of first and second order early reflections, as well as reverb.
The signal emitted by the audio source, encoded in an $n$-order Ambisonics' B-format, is decoded to an array of $N_{vl}$ virtual speakers that surround the listener forming a sphere. This array is capable of replicating the original sound field with some decent accuracy, especially if the used Ambisonics are of a higher order and the number of virtual speakers is relatively high. The virtual speakers themselves are basically a set of $N_{vl}$ different HRTF filters that alter the time and frequency domain aspects of the original signal using convolution, each of them in its own way. The measured impulse response used in each filter is determined by the direction of the virtual speaker relative to the listener, with the corresponding azimuth and elevation angles.

Head rotation is obviously taken into account. Whenever the user rotates his/her head, the played sounds are rotated around the virtual loudspeaker array in the opposite direction, applying time-variant rotation matrices in the Ambisonics domain. This allows the virtual audio sources to stay in a fixed location and not follow the user’s movements.

Contrary to the Convolvotron, this kind of system is ideal to create acoustic experiences that involve many audio sources and reflections, as the number of Ambisonic channels is totally independent of the amount of virtual sources to encode. As seen in the previous figure, it is possible to easily incorporate early reflections and reverb into the system before the encoding process with a simple room simulation model. Such a model can be implemented by applying unique successive delays and attenuation gains to the direct sounds, the considered early reflections and the reverb in order to differentiate them appropriately. The values for these delays and gains are determined by the distance between the origin of the corresponding sounds and the listener’s position. All of the reflection and reverb signals also go through low-pass filters, which intend to emulate the acoustic properties of the reflecting surfaces.
8 Incorporation of spatial audio in *Unity*

As previously explained in the report’s objectives, the ultimate purpose of this project is to implement spatial audio in the *Unity* game engine, especially in virtual reality scenarios. This section describes the built-in audio spatialization capabilities of the basic *Unity* software and the different available plugins that allow improving this aspect, as well as the modifications that have been carried out on them.

Several specific terms related to *Unity* might start to appear in the following part of the report. It should still be possible to understand the majority of it without the need to possess any knowledge about the game engine at all, although it is advisable to get acquainted with its basic concepts. A good way to do so is either to read through the *Basics* part of the latest version of the *Unity* manual [7] or fellow student Ari-Pekka Laiho’s report on the use of a VR headset in *Unity* [8].

8.1 *Unity*’s built-in spatialization features

The built-in spatial audio capabilities of the *Unity* game engine are quite primitive. They are practically limited to simple panning between the left and right channels. In other words, the software adjusts the sound gain for each channel in correspondence with the distance and direction between the *AudioListener* and the *AudioSource* components. As explained in section 4, by exclusively regulating the level difference between channels, sound localization becomes limited to the horizontal plane. Also, the lack of time differences makes this lateral localization restricted to a certain range of frequencies. In short, *Unity*’s built-in basic panning is certainly a quite poor spatialization feature.

Apart from panning, the software also incorporates a few other adjustable 3D audio features:

- **Doppler effect level**: The effect of the relative velocity between the *AudioSource* and the *AudioListener* components on the pitch can be adjusted with the *Doppler Level* slider, from a minimum value of 0 (deactivated) to a maximum one of 5.

- **Spread**: It is possible to adjust how much the 3D localization of the source influences the panning of stereo or multichannel sounds. This is done using the *Spread* slider, which can be regulated from 0 to 360. A value of 0 cancels the sound file’s original panning, imposing the one originating from the source’s position in space. If set to 180, the sound uses its original panning, and position is only relevant for the volume level. A value of 360 has the same effect as one of 0, but inverts de panning.

---

3 The version used for the execution of the practical part of this project has been the *Unity 5.6.0*, released on March 31, 2017.
• **Volume rolloff**: The type of curve that defines how the sound level decreases with the distance to the source can be selected in the *Volume Rolloff* field. It is possible to choose between logarithmic, linear and custom rolloff curves.

• **Minimum and maximum distances**: Minimum and maximum distance values can be adjusted as limits to the rolloff curve in the *Min Distance* and *Max Distance* fields. For distances to the source shorter than the established minimum or longer than the maximum distance value, the volume stops increasing or decreasing respectively and stays at the same level.

All of these 3D audio features can be activated or partially activated by sliding the *AudioSource* component’s *Spatial Blend* slider from a value of 0 (2D) up to 1 (3D). If the slider is set to a value of 0, spatial attenuation is completely ignored.

Fig. 8.1 *Unity’s 3D Sound Settings.*

Fig. 8.2 *The Spatial Blend slider.*
8.2 *Unity’s native audio plugin SDK*

*Unity Technologies* offers a separate native audio plugin software development kit⁴ that includes a wide range of extra audio features to complement the mentioned built-in functionalities of the game engine. Some of these audio features include various sound effects, equalizers, granulators and many more. However, the most relevant feature to this specific project is the audio spatializer SDK extension that is also included.

8.2.1 Audio spatializer SDK (🎧)

The audio spatializer SDK is based on the implementation of direct HRTF, which only takes into account the direct path of audio from the source to the listener and does not consider its transmission through space. This lack of interaction with the surroundings is obviously an important limitation, but the introduction of HRTF by itself it is still a considerable improvement in relation to *Unity’s* original built-in spatial audio capabilities.

This plugin operates using the same principle as the Convolvotron, explained in section 7. It makes use of experimentally measured impulse response values taken from a data set elaborated by the MIT Media Lab using a KEMAR dummy head [10]. This data set consists of measures taken between -40 to 90 degrees in elevation, for different azimuth angles from 0 to 355 degrees, and has been executed with a sample rate of 44.1 kHz. When downloading this data set, the measures for each of the azimuth, elevation and left/right ear configurations are provided as 11.6 ms long sound files containing 512 samples each.

![Waveform representation fragment of the impulse response at KEMAR's left ear](image)

*Fig. 8.3 Example: Waveform representation fragment of the impulse response at KEMAR's left ear for a source at an elevation of -40° and an azimuth of 0°, based on one of the MIT Media Lab's data files. The small dots represent each of the 512 samples.*

⁴ The latest version of this plugin is available to download for free from *Bitbucket* [9].
The spatializer mainly runs via two C++ scripts, included in the general native audio plugin SDK, named `hrtftable.cpp` and `Plugin_Spatializer.cpp`. A brief description of this couple of files is presented below, summarizing their respective roles and their most significant traits and variables:

- **hrtftable.cpp**: This script is responsible for storing all of the mentioned impulse response experimental values in a single list named `hrtfSrcData[]`, as well as the information regarding the considered elevation and azimuth angles. The arrangement of the information contained in the mentioned list is detailed at the beginning of the script as a concise annotation:

```plaintext
struct Channels // left and right ears
{
    struct Elevations // 14 elevation angles from -40..90 degrees
    {
        float numangles; // small integer stored as float
        float angles[numangles];
        struct Coefficients
        {
            float impulse[HRTFLEN];
            } coeffs[numangles];
    } elevations[14];
} channels[2];
```

The `numangles` variable refers to the amount of azimuth angle configurations included in the data set, which is of 56 in this case, and the list `angles[numangles]` specifies the particular values for these 56 azimuth angles. Both of these variables are determined at the beginning of the `hrtfSrcData[]` list as follows:

```
```

After these 57 initial values, the `hrtfSrcData[]` list stores the impulse responses for all of the measured source position configurations. The response for each configuration is represented as a series of amplitude sample values, in the form of float numbers. The number of samples per measure is determined in the variable `HRTFLEN`, which is accessible in the `Plugin_Spatializer.cpp` script described later. As previously explained, this number of samples per measured source position configuration is of 512 in the data set used by this spatializer plugin.

---

5 It must be noted that the two mentioned scripts are designed to work together with the `AudioPluginUtil.h` and `AudioPluginInterface.h` header files.
The waveform illustrated in Fig. 8.3, for example, is stored in the hrtftable.cpp script as the following list of float values:

```
0.000061f, 0.000061f, 0.000061f, 0.000031f, 0.000031f, -0.000031f, -
0.000031f, -0.000031f, -0.000031f, -0.000061f, -
0.000092f, -0.000092f, -0.000061f, -0.000031f, 0.000061f, 0.000092f,
0.000092f, 0.000061f, 0.000061f, -0.000061f, -0.000092f, -0.000153f,
-0.000153f, -0.000092f, -0.000122f, -0.000183f, 0.000122f, -
0.000061f, 0.000305f, -0.000183f, 0.000336f, -0.000519f, 0.000275f,
-0.000336f, 0.000153f, -0.000366f, 0.000366f, -0.000763f, 0.050140f,
0.173004f, -0.010376f, -0.226837f, -0.088257f, -0.025085f,
0.047363f, 0.222260f, 0.255096f, 0.112366f, 0.086121f, ...
... -0.002228f, -0.002167f, -0.000641f, 0.001526f, 0.001770f, -
0.000793f, -0.003479f, -0.004517f, -0.002655f, 0.000854f, 0.004028f,
0.004456f, 0.002594f, -0.000793f, -0.003510f, -0.004395f, -0.002899f
```

The only purpose behind explaining the structure of this particular script in this report is to facilitate any modifications that could be possibly done to it if desired. As pointed out at the beginning of section 7.2, HRTF systems that rely on measured data can either be based on a single standardized set of data for a certain body geometry, on several sets of data for various distinct body geometries, or even on a customized one. This spatializer plugin is designed to work using the standard set of KEMAR measurements, but it could perfectly be adapted to function with other sets of data without excessive effort. Doing so would only require downloading any other HRTF data sets, such as the ones mentioned in section 6 [3], and adjusting the supplied data to the described hrtftable.cpp script's structure. Most of the HRTF data sets available for download provide the data in the form of multiple short audio files. However, they usually offer the alternative of downloading the full database in a Matlab version, which makes it quite easier to obtain the sample values in a numeric format.

- **Plugin_Spatializer.cpp**: This other script takes care of constantly modifying the shape of the input signal from the audio file that is to be spatialized in accordance to one of the impulse responses stored in the hrtftable.cpp script. The appropriate impulse response signal is selected depending on the azimuth and elevation angles to the audio source’s position within the local space of the listener. Once it is selected, the script performs fast convolution on the input and the selected impulse response signals with the Fast Fourier Transform.

The structure and the particular objects and variables of this Plugin_Spatializer.cpp script will not be detailed in this report, as it is a quite complex file and there is no
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience

point in modifying most of its content. Nonetheless, a brief explanation regarding what the convolution of signals actually is, and how it is performed with the Fast Fourier Transform method, is presented below:

**Convolution of functions or signals**

Convolution is a mathematical operation that is done on two functions or signals, creating a third function that basically represents how one of the two original ones is altered by the other (their magnitude of superposition).

The convolution of two functions $f$ and $g$ is determined as the integral of the product of the two, having previously reversed and shifted one of them:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

(Equation 8.1)

It is much easier to understand what this operation actually does by visualising a graphical example step by step. Consider two functions: $f$ (square pulse) and $g$ (reverse saw pulse). The “manual” convolution of these two functions would consist of the following steps:

1. Express $f$ and $g$ in terms of a bound variable $\tau$.

![Graphical examples](image)

2. Reverse one of the two functions. In this example function $g$ will be reversed: $g(\tau) \rightarrow g(-\tau)$

![Graphical examples](image)
3. Allow function $g$ to slide along the $\tau$ axis by adding an offset: $g(t - \tau)$

4. Slide function $g$ from $-\infty$ to $\infty$, calculating the integral of the product between $f$ and $g$ whenever they intersect. Obtain the resulting function.
**Linear Convolution via Fast Fourier Transforms**

When convolving two finite length signals, linear convolution via Fast Fourier Transforms or FFTs is probably the most efficient method of doing so. It can be easily performed with just a few steps. Given a signal \( f \) with \( N_1 \) samples and a signal \( g \) with \( N_2 \) samples:

1. Pad each of the signals by adding zeroes at their end in the following manner, so that their lengths become equal:
   - Pad function \( f \) with \( N_2-1 \) samples
   - Pad function \( g \) with \( N_1-1 \) samples

2. Compute the FFT for each of the signals in order to obtain their frequency domain versions \( F_r \) and \( G_r \).

3. Calculate the product of \( F_r \) and \( G_r \). Multiplication in the frequency domain is equivalent to convolution. This is why applying FFTs supposes such an advantage in computing time in comparison to other methods that do not make use of the frequency domain.

4. Apply the Inverse Fast Fourier Transform or IFFT to the resulting product.

5. Obtain the resulting signal.

**Using the audio spatializer plugin in Unity**

Making use of this particular audio spatializer plugin in the Unity game engine is a very straightforward process. It is just necessary to follow five simple steps in order to get it to work:

1. Download Unity’s native audio plugin SDK from the Bitbucket website [9].

2. Drag the downloaded Assets folder to the project folder in Unity. The downloaded contents include demos and many extra files that do not have any relation to the spatializer feature, as the download is for the entire native audio plugin and not specifically for the spatializer. It is also possible to just import the Plugins folder to avoid all of these extra files.
3. Go to the AudioManager menu (Edit -> Project Settings -> Audio) and select Demo Spatializer in the Spatializer Plugin field:

![Fig. 8.4 Demo Spatializer selection.](image)

4. Enable the Spatialize checkbox in the AudioSource component of the game object whose audio is to be spatialized. Once this checkbox is ticked, the one under it becomes available. This other option, named Spatialize Post Effects, determines whether the spatialization is applied before or after any effect filters attached to the AudioSource. Make sure the Spatial Blend is set to full 3D.

![Fig. 8.5 Spatialize and Spatialize Post Effects checkboxes.](image)

5. Insert a monaural audio file into the AudioClip field. If the file is in stereo, it can be easily converted to mono using any kind of audio editing software. Audacity is an effective free software option to do so.
Use recommendations

Based on the experience gathered from experimenting with Unity’s audio spatializer, it is recommended to bear in mind the following advices:

- Do not use this plugin for scenes in which the user’s surroundings play an important role, as sound occlusion and reflections are not considered. The plugin works best for scenes that take place outdoors or in a space with sound absorbing surfaces.
- Use sound files that are relatively long or that repeat themselves\(^6\). The spatializer is not as effective for very short sounds, just like in natural sound localization.
- Use complex sounds. It is important to avoid excessively quiet sounds, simple tones such as sine waves and sounds that lack in high frequencies.
- Avoid using audio files that include reverberation characteristics.
- Animate the sound sources if possible. Moving sources are much quicker to locate than static ones.

8.3 Google VR’s SDK for Unity

Google offers a software development kit\(^7\) as an aid to building virtual reality applications for iOS and Android operating systems in Unity, especially meant for their Cardboard and Daydream VR platforms. Even though the general SDK is oriented towards these platforms, the spatial audio features it includes are perfectly suitable for any other VR device.

8.3.1 Google VR’s spatial audio support

Just like Unity’s audio spatializer presented in the previous section, Google VR’s spatial audio support is mainly based on the use of HRTFs. However, contrary to Unity’s plugin, Google VR’s spatializer does take the transmission of sound through space into consideration, which is certainly a huge advantage when it comes to creating realistic audio scenes.

To be able to manage all of the numerous virtual sound sources that result from considering early reflections and reverberation, amongst other features, this plugin uses the Ambisonics + binaural rendering method described at the end of section 7.2.2.

Unlike Unity’s audio spatializer SDK, the download for the Google VR’s plugin does not include the different files that comprise the native plugin separately; only the general library file and the accompanying assets are provided. It is therefore not possible to access the

\(^6\) The looping of a source can be achieved by enabling the Loop option in the respective AudioSource component.

\(^7\) The latest version of this SDK is available to download for free from the Google VR’s developers website [11].
native code, so aspects such as the measured impulse response values table used for the HRTF filtering process cannot be edited. Still, the plugin offers quite a lot of customization options via five different classes that communicate with the native code implementation of the audio system and with each other. These classes are defined in five different C# scripts, named GvrAudio.cs, GvrAudioListener.cs, GvrAudioSource.cs, GvrAudioRoom.cs and GvrAudioSoundfield.cs. The mentioned scripts can be easily accessed at the GoogleVR -> Scripts -> Audio folder once the plugin is imported to the project’s assets. A brief description for each of the classes is presented below, summarizing their respective roles and their most relevant traits and variables:

- **GvrAudio**: This is the main class of the Google VR audio system. It is the one responsible for communicating with the native code, and the only class through which native functions can be called. It basically holds most of the different functions, attributes, types and properties that condition the audio spatialization features. Below are the most relevant ones for adjustment purposes:

  - **Quality** (type): It determines the quality of the binaural rendering, or in other words, the amount of virtual loudspeakers or HRTF filters that surround the listener. It can be set to three different levels: Stereo, Low or High. The higher the quality, the better the audio spatialization effects will be. Selecting a higher rendering quality also increases the CPU usage significantly.

```csharp
public enum Quality {
    Stereo = 0, ///< Stereo-only rendering
    Low = 1, ///< Low quality binaural rendering (first-order HRTF)
    High = 2  ///< High quality binaural rendering (third-order HRTF)
}
```

The quality type can be quickly selected through the GvrAudioListener script component.

- **ComputeOcclusion** (function): This function calculates the decrease in intensity that an incoming sound from an audio source suffers when it is occluded by some interfering physical game object. It does so by using point source detection with the RaycastAll function. This function simply counts the number of times the travelling sound hits physical objects before arriving to the listener, and dampens the resulting sound proportionally.

  Even though this function is not really designed to be modified, the magnitude of the applied occlusion can be easily adjusted simply by changing the value

---

8 A full list of the contents for each of the classes can be found in the reference documentation in the Google VR website [12].
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience

marked in turquoise in the code below. The higher the value is set to, the more noticeable the occlusion effect will be. This value is set to 1 as default, but it can be adjusted to any value at all (preferably within the range between 0.6 and 1 if a decent level of realism is to be achieved).

```csharp
public static float ComputeOcclusion(Transform sourceTransform) {
    float occlusion = 0.0f;
    if (initialized) {
        Vector3 listenerPosition = listenerTransform.position;
        Vector3 sourceFromListener = sourceTransform.position - listenerPosition;
        RaycastHit[] hits = Physics.RaycastAll(listenerPosition, sourceFromListener, sourceFromListener.magnitude, occlusionMaskValue);
        foreach (RaycastHit hit in hits) {
            if (hit.transform != listenerTransform && hit.transform != sourceTransform) {
                occlusion += 1f;
            }
        }
        return occlusion;
    }
}
```

The `GvrAudio.cs` script does not need to be attached to any game object in order to function. It is meant to be simply left in its containing folder.

- **GvrAudioListener**: Google VR's listener class has the sole purpose of enhancing Unity's standard `AudioListener` component so that the spatial audio features can work properly.

The `GvrAudioListener.cs` script needs to be attached as a component to the game object that represents the listener. This script is not supposed to substitute the standard `AudioListener` component; both need to coexist in the same game object.

As previously mentioned, the `Quality` parameter defined in the `GvrAudio` class can be adjusted from the `Inspector` once the `GvrAudioListener.cs` script has been attached to a game object. Also, a couple of attributes of interest are introduced in this class:

- **globalGainDb** (attribute): This attribute determines the gain that is applied to the final audio the listener experiences. Its value in decibels is set to zero as default, but it can be adjusted within the range between -24 and 24 dB.

```csharp
public float globalGainDb = 0.0f;
```
• **occlusionMask** (attribute): This second attribute establishes which layers are to be considered for the sound occlusion detection.

```csharp
public LayerMask occlusionMask = -1;
```

Both the *globalGainDb* and *occlusionMask* attributes are available to be adjusted from the *Inspector* as well.

![Gvr Audio Listener component's menu in the Inspector.](image)

**GvrAudioSource**: The *GvrAudioSource* is basically an improved version of *Unity's AudioSource*. It introduces the advanced audio spatialization features that the standard audio source component of the game engine lacks.

The *GvrAudioSource.cs* script can be attached as a component to any game object that is meant to emit sound with advanced spatial characteristics. This script substitutes the standard *AudioSource* component.

Its appearance in the *Inspector* is very similar to that of the standard *AudioSource* component, as it also includes most of its simple spatialization features\(^9\), such as the Doppler effect level, spread and volume rolloff curves, as well as basic adjustment options such as gain, volume, pitch, etc. However, it presents a few more settings that correspond to the following attributes:

• **hrtfEnabled** (attribute): This attribute determines whether HRTF binaural rendering for the audio source is activated or not. If the previously mentioned *Quality* adjustment is set to *Stereo*, this setting has no effect at all.

```csharp
private bool hrtfEnabled = true;
```

\(^9\) These basic spatial audio features are described in detail in section 8.1.
• **occlusionEnabled** (attribute): This second attribute simply establishes if the sound occlusion effect is enabled or not.

```csharp
public bool occlusionEnabled = false;
```

The next four attributes of the `GvrAudioSource` class listed below refer to the concept of directivity. Directivity is one the main features introduced by the Google VR spatial audio plugin. It refers to the pattern that defines how sound, conditioned by occlusion, emanates from an audio source. Most sources do not emit sound in equal measure in every direction. For instance, when a person talks, the emitted sound is much louder in front than behind of the talking person. The Google VR plugin allows shaping the source emission directivity in the horizontal plane, as well as the pattern of sound sensitivity of the listener for a specific source.

• **directivityAlpha** (attribute): This particular alpha parameter determines the shape of the source emission. It controls the balance between an omnidirectional and a dipole directivity pattern. A value of zero corresponds to a fully omnidirectional source emission pattern, whereas a value of 1 corresponds to a pure dipole pattern.

```csharp
public float directivityAlpha = 0.0f;
```

• **directivitySharpness** (attribute): This other source emission attribute sets the sharpness of the directivity pattern. Higher values result in a more narrow emission pattern. It can be set to a value between 1 and 10.

```csharp
public float directivitySharpness = 1.0f;
```

• **listenerDirectivityAlpha** (attribute): This attribute determines the shape of the listener’s sound sensitivity pattern for the particular audio source. Just like the directivityAlpha attribute, it controls the balance between an omnidirectional and a dipole directivity pattern. A value of zero corresponds to a fully omnidirectional source emission pattern, whereas a value of 1 corresponds to a pure dipole pattern.

```csharp
public float listenerDirectivityAlpha = 0.0f;
```

• **listenerDirectivitySharpness** (attribute): The listenerDirectivitySharpness attribute establishes how narrow the listener’s sound sensitivity pattern is. As
for the `directivitySharpness` attribute, higher values result in a more narrow emission pattern. It can also be set to a value between 1 and 10.

```csharp
public float listenerDirectivitySharpness = 1.0f;
```

The directivity patterns for both the source emission and the listener's sound sensitivity show up visually as *gizmos* in the *Scene* view whenever the game object containing the *GvrAudioSource* component is selected.

![Fig. 8.7 Source emission directivity pattern as seen in Unity's Scene view.](image)

All of the mentioned new attributes appear at the bottom of the *GvrAudioSource* component menu in the *Inspector*, and can easily be adjusted from there.

![Fig. 8.8 Bottom part of the *GvrAudioSource* component's menu in the *Inspector*.](image)

- **GvrAudioRoom**: The *GvrAudioRoom* class is meant to act as a room simulation model. It simulates the effects the surroundings have on the sound emitted by the audio sources. It does so by adding early reflections and reverberation effects into the scene.
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience

The *GvrAudioRoom.cs* script can be attached to any game object. It is usually attached to an empty game object and adjusted appropriately so that it matches the desired physical room dimensions.

This class introduces several adjustable parameters, ultimately offering a fairly complete set of customization options for the introduced room simulation model:

- **SurfaceMaterial** (type): This public type establishes the acoustic properties of a particular room surface. A wide range of predetermined materials with diverse acoustic properties is available for selection.

  For instance, selecting a surface made of wood or plaster will result in much less noticeable reflections than those from a surface made of glass or marble.

```csharp
public enum SurfaceMaterial {
    Transparent = 0,           ///< Transparent
    AcousticCeilingTiles = 1,  ///< Acoustic ceiling tiles
    BrickBare = 2,             ///< Brick, bare
    BrickPainted = 3,          ///< Brick, painted
    ConcreteBlockCoarse = 4,   ///< Concrete block, coarse
    ConcreteBlockPainted = 5,  ///< Concrete block, painted
    CurtainHeavy = 6,          ///< Curtain, heavy
    FiberglassInsulation = 7,  ///< Fiberglass insulation
    GlassThin = 8,             ///< Glass, thin
    GlassThick = 9,            ///< Glass, thick
    Grass = 10,                ///< Grass
    LinoleumOnConcrete = 11,   ///< Linoleum on concrete
    Marble = 12,               ///< Marble
    Metal = 13,                ///< Galvanized sheet metal
    ParquetOnConcrete = 14,    ///< Parquet on concrete
    PlasterRough = 15,         ///< Plaster, rough
    PlasterSmooth = 16,        ///< Plaster, smooth
    PlywoodPanel = 17,         ///< Plywood panel
    PolishedConcreteOrTile = 18, ///< Polished concrete or tile
    Sheetrock = 19,            ///< Sheetrock
    WaterOrIceSurface = 20,    ///< Water or ice surface
    WoodCeiling = 21,          ///< Wood ceiling
    WoodPanel = 22             ///< Wood panel
}
```

- **leftWall** (attribute): It determines the room surface material used for the left wall of the room (in the negative x direction).

```csharp
public SurfaceMaterial leftWall = SurfaceMaterial.PlasterSmooth;
```
- **rightWall** (attribute): It determines the room surface material used for the right wall of the room (in the positive x direction).

  ```java
  public SurfaceMaterial rightWall = SurfaceMaterial.PlasterSmooth;
  ```

- **floor** (attribute): It determines the room surface material used for the floor of the room (in the negative y direction).

  ```java
  public SurfaceMaterial floor = SurfaceMaterial.PlasterSmooth;
  ```

- **ceiling** (attribute): It determines the room surface material used for the ceiling of the room (in the positive y direction).

  ```java
  public SurfaceMaterial ceiling = SurfaceMaterial.PlasterSmooth;
  ```

- **backWall** (attribute): It determines the room surface material used for the back wall of the room (in the negative z direction).

  ```java
  public SurfaceMaterial backWall = SurfaceMaterial.PlasterSmooth;
  ```

- **frontWall** (attribute): It determines the room surface material used for the front wall of the room (in the positive z direction).

  ```java
  public SurfaceMaterial frontWall = SurfaceMaterial.PlasterSmooth;
  ```

- **reflectivity** (attribute): This attribute establishes what proportion of the direct sound is reflected by the room’s surfaces. It is set to a default value of 1, but it can be adjusted from 0 to 2. The higher the value, the more noticeable the reflections will be.

  ```java
  public float reflectivity = 1.0f;
  ```

- **reverbGainDb** (attribute): The **reverbGainDb** attribute applies an adjustment gain to the reverberation effect. Its default value of zero leaves the reverb unaltered. It can be set to a value between -24 and 24 dB.

  ```java
  public float reverbGainDb = 0.0f;
  ```
- **reverbBrightness** (attribute): The `reverbBrightness` attribute determines the balance between the reverberation’s low and high frequencies. It is set to a neutral default value of zero, and can be adjusted from -1 to 1.

```csharp
public float reverbBrightness = 0.0f;
```

- **reverbTime** (attribute): This attribute adjusts the overall duration of the reverberation by a positive scaling factor that can be set to a value between 0 and 3. Its value is set to 1 as default.

```csharp
public float reverbTime = 1.0f;
```

- **size** (attribute): The `size` attribute, as its name suggests, determines the size of the room model. It works exactly like the standard `Scale` option in the `Transform` section of any game object, but is independent to it.

```csharp
public Vector3 size = Vector3.one;
```

All of these options are available to be easily adjusted from the `GvrAudioRoom` component’s menu in the *Inspector*.

![Fig. 8.9 The GvrAudioRoom component’s menu in the Inspector.](image)
• **GvrAudioSoundfield**: This final class is specifically designed to create sound field sources instead of the previously explained standard Google VR audio sources. This kind of sources play first-order Ambisonic files, creating a sound field in which audio can be heard in different ways for every direction. Sound field sources only depend on head rotation, so they are especially suitable to represent distant sounds.

Just like the regular GvrAudioSource, the GvrAudioSoundfield.cs script component can be attached to any game object that is meant to emit this sort of spatial sound. It also substitutes the standard AudioSource component.

This component also shares some basic similarities with Unity's AudioSource. However, unlike the GvrAudioSource, it lacks the built-in spatialization features, as these would be useless for the sound field format it uses. Apart from the basic features it shares with the regular AudioSource, such as gain regulation, pitch, etc., it does not offer any adjustment options at all.

The GvrAudioSoundfield component has the following appearance in the Inspector:

![GvrAudioSoundfield component's menu in the Inspector.](image)

**Using Google VR's spatial audio support in Unity**

Below is a brief summary of the different steps required to make use of Google VR's spatial audio support:

1. Download the Google VR SDK for Unity from the developers website [11]. The current version of this plugin only works with Unity 5.6 or later versions.
2. Drag the downloaded package file to the project folder in Unity. The downloaded package includes all kinds of extra files such as demos, materials and many other assets that are not required in order for the spatial audio features to work, as the download is for the entire Google VR development kit and not specifically for the audio part. It is also possible to just keep the Plugins folder and the audio scripts to avoid all of these extra files.

3. Go to the AudioManager menu (Edit -> Project Settings -> Audio) and select GVR Audio Spatializer in the Spatializer Plugin field:

![Fig. 8.11 GVR Audio Spatializer selection.]

4. Insert the GvrAudioListener.cs script component\(^\text{10}\) in the game object that is going to act as the listener, keeping the standard AudioListener component. This game object is usually the Main Camera.

5. To add an audio source with spatial support to the current scene, attach the GvrAudioSource.cs script component to a game object. Make sure the Enable Occlusion and Enable HRTF options are activated, or else the software will not take the sound occlusion and spatialization effects into consideration, and the resulting acoustic experience will be rather bland.

(Optional) To add a sound field audio source to the current scene instead, attach the GvrAudioSoundfield.cs script component to a game object.

\(^{10}\) All of the Google VR audio-related scripts can be found in the GoogleVR-> Scripts -> Audio folder in the Assets.
6. Insert a monaural audio file into the AudioClip field of the GvrAudioSource component. If the file is in stereo, it can be easily converted to mono using any kind of audio editing software. Audacity is an effective free software option to do so.

If the audio source is of the sound field type, insert a couple of audio clips instead, with two-channels each (the WY and XZ channel pairs for the first-order Ambisonics B-format, respectively). This sort of audio files can be obtained either by creating them synthetically from a monaural file using a digital audio workstation such as Ambix, or by recording them with a special Ambisonics microphone.

7. If it is desired to include a room simulation model, attach the GvrAudioRoom.cs script component to any game object. Adapt the dimensions of the room model with the Size parameters so that they correspond to those of the physical room.

Use recommendations

Based on the experience gathered from experimenting with Google VR's spatial audio support SDK, it is recommended to bear in mind the following advices:

- Use this plugin for scenes that involve a complex acoustic scenario. It can be scenes with a high number of audio sources, scenes that require including a room simulation model, etc.
- Use sound files that are relatively long or that repeat themselves\(^{11}\). The spatializer is not as effective for very short sounds, just like in natural sound localization.
- Use complex sounds. It is important to avoid excessively quiet sounds, simple tones such as sine waves and sounds that lack in high frequencies.
- Avoid using audio files that include reverberation characteristics.
- Animate the sound sources if possible. Moving sources are much quicker to locate than static ones.
- Set the Quality field of the GvrAudioListener to High, as long as it does not compromise the performance of the scene.
- Change the occlusion value mentioned for the ComputeOcclusion function of the GvrAudio class from the default value of 1 to 0.8, as the default intensity for the occlusion effect is slightly excessive.
- Use a room simulation model only for scenes that happen in an enclosed space.

---

\(^{11}\) The looping of a source can be achieved by enabling the Loop option in the respective GvrAudioSource or GvrAudioSoundfield component.
8.4 Comparison chart

Below is a table summarizing the capabilities of each of the described audio software kits for the Unity game engine, specifically for those features related to audio spatialization:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Unity's built-in audio features</th>
<th>Unity's Audio Spatializer SDK</th>
<th>Google VR's Spatial Audio Support SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard panning</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Volume rolloff</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Spread</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Doppler effect</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>HRTF filtering</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Room sim. model</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Occlusion</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Directivity</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 8.1 Comparison chart between audio software options.

This comparison chart can serve as a good reference for selecting the most appropriate audio software kit for a particular project. It is recommendable to choose the least complex kit that fulfils the desired capabilities, in order to avoid excessive file sizes and/or CPU usage.
9 Spatial audio demos in *Unity*

After spending a long period of time getting used to the *Unity* game engine and experimenting with the several presented audio plugins, three demonstration scenes with different complexity levels have been developed. These demos have been based on the implementation of spatial audio in different kinds of scenarios, allowing the user to experience and ultimately understand the virtues audio spatialization provides in VR-related applications. The three demos are described in greater detail in the next sections, explaining the concept behind each of them, what audio features they include, which plugins and hardware they use and how to operate them.

9.1 Demo 1: Direct HRTF test

9.1.1 Concept

The sole purpose behind this first demo's simple design is to demonstrate how effective a direct HRTF filtering system can be at source localization in comparison to the built-in basic panning system.

It is an interactive experience in which the user is able to control the movement of a sphere that acts as an audio source. The sphere can be moved around a mannequin head 3D model that represents the listener’s own head from a third-person view. This way, the user hears the audio the same way the third-person model would, and is still able to visually tell what the position of the audio source is at all times, even when it is behind him.

9.1.2 Spatial audio features

The only spatial audio features considered in this demo are the mentioned direct HRTF filtering, logarithmic volume rolloff and the Doppler effect. Spatialization can be activated and deactivated at any moment for the comparison purposes mentioned earlier.

9.1.3 Plugins and hardware

Considering the necessity of solely incorporating direct HRTFs, *Unity’s* native audio plugin SDK has been used in this particular demo, as it is the least complex option that includes the required feature.

This demo is meant to be visualized in any standard 2D screen, so no special virtual reality-related hardware is required. Only regular headphones are needed in order to experience the spatial audio as intended.
9.1.4 Operation

The movement of the audio source sphere within the space surrounding the head model is controlled with the following set of keys:

- **Z-axis (forward/backward):** W (forward) and S (backward) keys
- **X-axis (left/right):** A (left) and D (right) keys
- **Y-axis (up/down):** Up and Down arrow keys

The audio spatialization feature is disabled when first opening the scene. It can be enabled or disabled by pressing the *Space* bar of the keyboard accordingly. The current state of the spatializer is displayed at the top left corner of the screen as a green or red underline for the *ON* and *OFF* options.

![Fig. 9.1 Game view of Demo 1.](image)

9.2 Demo 2: Office cinematic scene

9.2.1 Concept

The second demo was basically designed bearing in mind two main purposes. On the one hand, it intends to evaluate how effective HRTF-based source localization can be for moving audio sources in comparison to static ones. On the other hand, it tries out features that allow the acoustic replication of surrounding spaces to be possible, with the addition of reflections and reverberation effects.
This demo is basically a cinematic sort of experience that happens in a virtual office environment. The user sits facing a fixed direction. In front of him/her is a desk with several common office objects and devices. In addition, there is a half-open window that communicates with the outside world right above the desk. All of these elements, as well as other emerging ones, take turns in creating the resulting acoustic experience.

Up to six different game objects act as temporary sound sources in this scene, including a laptop computer, a telephone, a plane and a fly. Some of these sources remain still, while others move around and are therefore easier to locate. There is also a constant non-spatialized background sound file that plays during the entire demo at a low volume, representing the ambient noise of the office.

9.2.2 Spatial audio features

This demo includes HRTF filtering, logarithmic volume rolloff, sound occlusion and a room simulation model. The purpose behind the incorporation of the two latter audio features is to add the perception of being in a room and not in an open space. The room simulation model allows the reflections and the reverb of the surroundings to be taken into account, and sound occlusion can be noticed for audio sources on the outside of the building that pass alongside the open window.

9.2.3 Plugins and hardware

Google VR’s spatial audio support SDK has been used in this demo due to the necessity of including sound occlusion and a room simulation model as part of the acoustic experience, in addition to the HRTF filtering.
The “Office cinematic scene” demo is meant to be visualized in a 3D screen. A JVC BlackSapphire 3D TV has been used for testing it in the virtual reality lab. Audio-wise, just regular headphones are needed.

9.2.4 Operation

This particular demo does not require any input from the user. It is simply a two minutes long cinematic experience in which acoustic events occur automatically at predetermined times, following the order depicted below:

![Diagram of events](image)

1 – The laptop starts up.

2 – The telephone rings.

3 – Someone walks behind the listener from right to left.

4 – A folder is dropped onto the desk.

5 – A plane flies by.

6 – A fly enters the room and flies around the listener.

Fig. 9.3 Succession of events in Demo 2

9.3 Demo 3: Interactive VR acoustic room

9.3.1 Concept

The idea behind the third demo’s design was to create a final interactive VR experience that included as many of the spatial audio features presented throughout the report as possible. Unlike the previous demos, this experience also allows the user to move freely by introducing head tracking, which in turn makes source localization a lot more effective.

At the beginning of the demo, the user finds him/herself in a minimalistic small room. In front of the user is a round omnidirectional loudspeaker hanging from the ceiling at about the same height as his/her head, as well as a simple control panel with several buttons and a lever. By interacting with these controls, the user is able to turn on the loudspeaker and activate its movement around the room following a predetermined square path. The height of
the loudspeaker and the size of the room can both be adjusted as well by pressing the pertinent buttons. Every time the size of the room changes, the loudspeaker adapts its path and its movement speed to the new dimensions. In addition, a couple of wooden panels located in front and behind of the user are also included in order to apply sound occlusion effects in some of the loudspeaker’s movement configurations.

![Image of a control panel with buttons and switches](image)

**Fig. 9.4 Game view of Demo 3.**

### 9.3.2 Spatial audio features

Just like the previous demo, this final demo includes HRTF filtering, logarithmic volume rolloff, sound occlusion and a room simulation model, as well as a subtle Doppler effect. However, the introduction of head tracking, and the consequent real time adaptation of the HRTF filtering process to the relative direction between the source and the user, results in a substantial improvement in the user’s source localization capabilities.

As mentioned in the demo’s concept, the sound occlusion effects happen whenever the direct path of the sound emitted by the loudspeaker is blocked by one of the wooden panels.

The room simulation model plays a particularly important role in this third demo. This model adapts its dimensions to the dynamic size of the room throughout the demo, changing the reflection and reverberation effects accordingly. For instance, when the physical room is at its smallest state, reverberation is much less noticeable than when it adopts larger dimensions.
9.3.3 Plugins and hardware

Google VR's spatial audio support SDK has been used in this demo, as it is the only option out of the presented plugins that allows HRTF filtering, sound occlusion and room simulation models to be all incorporated in a project.

The visual part of this final demo is designed to be experienced with a HTC Vive VR headset, which supports head tracking and includes handheld tracked controls. Only regular headphones are needed in order to experience the spatial audio as intended.

9.3.4 Operation

The operation of this particular demo revolves around the use of the 3D control panel included in the scene. All of the adjustable options described in the demo’s concept section can be controlled by interacting with the components of this panel.

Moving the POWER lever to the ON position activates the loudspeaker audio source. A little green LED lights up on the top part of the loudspeaker and it starts to play a sound file. The loudspeaker can be deactivated at any moment by moving the lever back to the OFF position, which also stops its movement.

The top row of buttons marked with the SPEAKER HEIGHT label allow the user to control the vertical position of the loudspeaker when pressed. The following options are available:

- **LOW**: The loudspeaker moves to the bottom of the room and remains at that height.
- **MID**: The loudspeaker stays at a middle distance between the floor and the ceiling of the room.
- **HIGH**: The loudspeaker moves to the top of the room and remains at that height.
- **\(\wedge\wedge\)**: The loudspeaker moves up and down following a triangular wave motion.
On the other hand, the bottom row of buttons marked with the *MOVEMENT PHASE* label control the dimensions of the room, as well as the path and the speed\(^{12}\) of the loudspeaker. The four different configurations to choose from are the following:

- **1**: This is the configuration that is first active when the demo starts. It is the configuration in which the room has the smallest dimensions and the loudspeaker moves the slowest.

- **2**: The longitudinal dimension of the room in this second configuration is slightly longer than in the first, but the rest stays the same. This elongation is enough to create a space between the front and back walls and each of their closest wooden panels. The new path of the loudspeaker passes behind these panels; so sound occlusion effects are present at times.

- **3**: In this configuration, the room suffers a significant lateral elongation. The front and back walls stay the same as in the first configuration. The loudspeaker moves faster, as it has a longer distance to travel.

- **4**: The fourth configuration drastically changes the dimensions of the room, making it a lot bigger overall. In addition, the floor is lowered and the ceiling rises. However, the platform on which the user stands remains at the same height as in the other configurations. The loudspeaker moves the fastest in this configuration.

\(^{12}\) The speed of the loudspeaker is proportional to the distance it has to travel to complete a full lap of its determined path. The bigger the dimensions of the room, the higher the velocity.
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience

**Fig. 9.8 Configuration 3**

**Fig. 9.9 Configuration 4**
10 Project budget

This section details all of the costs associated with the development of the project. The project’s budget has been estimated considering the following aspects:

- Time dedication
- Software licenses
- Equipment
- Tuition fees

10.1 Time dedication

A total time amount of approximately 480 hours of labour has been dedicated to the development of the project since its start at the beginning of last February. For the most part, this time dedication comprises the learning and research processes, the experimentation and the creation of demos in the *Unity* game engine, and the making of this report.

The chronological distribution of these activities over the course of the project has been roughly the following:

![Fig. 10.1 Distribution of activities over time.](image)

Considering an hourly wage of 12€ for a junior engineer, the cost derived from manual labour amounts to a total of about 5760€.

10.2 Software licenses

For the practical part of this project, several programs have been used. These programs include the *Unity* game engine, the spatial audio plugin SDKs, a couple of digital audio editors and a word processor.
Most of the utilized software has been free; except for the *Ableton live* digital audio workstation and the *Microsoft Office* license. Below is a table listing the used software and its associated costs:

<table>
<thead>
<tr>
<th>Software type</th>
<th>Name</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td><em>Unity Personal 5.6.0</em></td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td><em>Audacity 2.1.3</em></td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td><em>Ableton Live 9 Standard</em></td>
<td>349€</td>
</tr>
<tr>
<td></td>
<td><em>Microsoft Office</em></td>
<td>149€</td>
</tr>
<tr>
<td>Plugin</td>
<td><em>Unity native audio plugin SDK</em></td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td><em>Google VR SDK</em></td>
<td>Free</td>
</tr>
</tbody>
</table>

**TOTAL** 498€

Table 10.1 Software costs.

### 10.3 Equipment

The practical part of the project has also required the utilization of several devices. Some of the equipment has been lent by the university’s *Centre of Virtual Reality* [13]. The used hardware devices and their associated costs are detailed below:

<table>
<thead>
<tr>
<th>Hardware device</th>
<th>Market price</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ASUS F556U PC</em></td>
<td>799€</td>
</tr>
<tr>
<td><em>Apple MacBook Pro 15” Retina Display i7 2.2GHz</em></td>
<td>2000€</td>
</tr>
<tr>
<td><em>Apple EarPods</em></td>
<td>35€</td>
</tr>
<tr>
<td><em>JVC BlackSapphire 55” 3D TV</em></td>
<td>1080€</td>
</tr>
<tr>
<td><em>HTC Vive</em></td>
<td>699€</td>
</tr>
</tbody>
</table>

**TOTAL** 4613€

Table 10.2 Equipment costs.

### 10.4 Tuition fees

The execution of this project also involves a series of tuition fees, which cover the institution enrolment and the tutoring expenses associated with the development of the project. These costs amount to a total of 3275€.
10.5 Total cost of the project

The total cost for the entire project, obtained by adding together all of the previous mentioned expenses, is summarized in the table below:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time dedication</td>
<td>5760€</td>
</tr>
<tr>
<td>Software licenses</td>
<td>498€</td>
</tr>
<tr>
<td>Equipment</td>
<td>4613€</td>
</tr>
<tr>
<td>Tuition fees</td>
<td>3275€</td>
</tr>
<tr>
<td><strong>GRAND TOTAL</strong></td>
<td><strong>14146€</strong></td>
</tr>
</tbody>
</table>

Table 10.3 Total cost of the project.
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience
Conclusions

Over the course of this project, it has been possible to learn and provide explanations for most of the aspects concerning the world of spatial audio. A lot of time has also been put into experimenting with the implementation of three-dimensional audio in virtual reality applications, which has served as an ideal source of practical knowledge.

Below are a series of conclusions that can be taken from this report:

- The two-channel based human auditory system is capable of locating sound sources using psychoacoustics.
- Lateral localization is possible thanks to Interaural Time Differences (ITD) and Interaural Level Differences (ILD). These two cues complement each other at different frequencies.
- Head motion is essential in eliminating emerging lateral localization ambiguities.
- Vertical localization is based exclusively on the geometry of a person’s pinnae, which causes the frequency response for an incoming sound to be dependent on the direction to the source.
- Range is estimated with loudness variations, motion parallax and the ratio of direct to reverberant sound.
- Reflections do not interfere with the human ability to locate sources, as only the early signals that reach the ears are taken into account.
- It is possible to record sound and keep its spatial properties using either binaural dummy head recording (for headphones) or Ambisonic (for loudspeakers) techniques. However, these two techniques by themselves have certain limitations that make them inappropriate for most VR applications.
- The Head Related Transfer Function (HRTF) determines what the frequency response at the listener’s eardrums is for a certain audio source. It encompasses all of the physical cues that condition sound localization.
- HRTFs can be implemented in audio systems as complex mathematical models or using measured values. They allow audio to be spatialized synthetically.
- Headphones are usually a better choice than loudspeakers for audio systems that are based on HRTFs, although they require head tracking when used in VR applications that involve user motion.
- The Convolvotron and the Ambisonics + Binaural rendering systems both work using measured HRTFs. The first is suitable for simple spatial audio scenarios with few audio sources, whereas the second is capable of handling a large number of sources with a lower CPU usage.
Incorporation of 3D audio into virtual reality scenes for an improved immersive experience

- **Unity**'s built-in spatial audio capabilities are very poor. They are practically limited to standard panning for a rough lateral localization and volume rolloff curves for range estimation.

- **Unity**'s audio spatializer SDK introduces direct HRTF implementation, which greatly improves the ability to estimate the direction to audio sources. However, it still does not consider the effects of the surroundings, which is a huge limitation.

- **Google VR**'s spatial audio support SDK introduces HRTF filtering as well. In addition, it provides features such as sound occlusion, directivity patterns and advanced room simulation models.

- It is recommendable to use the least complex of these software kits that includes the necessary features, in order to avoid excessive CPU usage and/or file sizes.

It has been proved that there is a considerable amount of effective ways to record, create and reproduce spatial audio. Despite this, it is still quite uncommon to come across advanced three-dimensional audio in the world of multimedia in general, especially when compared to the popularity of 3D visuals.

Even though most of the theory behind sound localization has been known for decades, the incorporation of spatial audio in VR applications has not begun to be effective until recently. With the high level of complexity game engines and similar kinds of software have reached during the last years, it is now starting to become viable to add advanced audio features into already complex scenes without compromising functionality.

Spatial audio can definitely be expected to become an essential part of virtual reality systems during the upcoming years, and the introduction of other senses considered to be dispensable until now will eventually follow. After all, the ultimate purpose of virtual reality is to recreate our perception of the world, and such a recreation will never even come close to being complete until all five senses are taken into consideration and perfected.
Acknowledgements

First of all, I would like to express my gratitude to my supervisor Toni Susín for presenting me with the opportunity to develop a project based on a subject I am truly passionate about, providing me with general guidance and placing his trust in me from the very first day.

I would also like to thank Jordi Moyés and the Centre of Virtual Reality for supplying me with the room and the equipment necessary to test and perfect the practical part of the project, as well as helping out whenever needed.

Finally, I would like to acknowledge with appreciation the role of the people involved in my previous project on binaural audio, especially Björn Henriksson and Mårten Eriksson. At the time, they helped in the construction of a homemade binaural recording dummy head, which has turned out to be fairly convenient for this new project.
Bibliography

Bibliographical references


Complementary bibliography


