

PERCEPTUAL AUDIO SOURCE CULLING FOR VIRTUAL  
ENVIRONMENTS

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF INFORMATICS  
OF  
MIDDLE EAST TECHNICAL UNIVERSITY

ALI CAN METAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF MASTER OF SCIENCE  
IN  
MODELLING AND SIMULATION

MARCH 2016



Approval of the thesis:

**PERCEPTUAL AUDIO SOURCE CULLING FOR VIRTUAL ENVIRONMENTS**

submitted by **ALI CAN METAN** in partial fulfillment of the requirements for the degree of **Master of Science in Modelling and Simulation Department, Middle East Technical University** by,

Prof. Dr. Nazife Baykal \_\_\_\_\_  
Dean, Graduate School of **Informatics**

Assoc. Prof. Dr. Hüseyin Hacıhabiboğlu \_\_\_\_\_  
Head of Department, **Modelling and Simulation**

Assoc. Prof. Dr. Hüseyin Hacıhabiboğlu \_\_\_\_\_  
Supervisor, **Modelling and Simulation, METU**

**Examining Committee Members:**

Prof. Dr. Yasemin Yardımcı Çetin \_\_\_\_\_  
Infomational Systems, METU

Assoc. Prof. Dr. Hüseyin Hacıhabiboğlu \_\_\_\_\_  
Modelling and Simulation Department, METU

Assoc. Prof. Dr. Ahmet Oğuz Akyüz \_\_\_\_\_  
Computer Engineering Department, METU

Assist. Prof. Dr. Tolga İnan \_\_\_\_\_  
Electrical and Electronics Engineering Department, TEDU

Assoc. Prof. Dr. Alptekin Temizel \_\_\_\_\_  
Modelling and Simulation Department, METU

**Date:** \_\_\_\_\_



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ALI CAN METAN

Signature :

# ABSTRACT

## PERCEPTUAL AUDIO SOURCE CULLING FOR VIRTUAL ENVIRONMENTS

, Ali Can Metan

M.S., Department of Modelling and Simulation

Supervisor : Assoc. Prof. Dr. Hüseyin Hacıhabiboğlu

March 2016, 59 pages

Game engines and virtual environment software that are in use today, use various techniques to synthesize spatial audio. One such technique, is through the use of head related transfer functions, in conjunction with artificial reverberators. For any virtual environment, synthesizing large amounts of spatial audio through these methodologies, will impose a performance penalty for the underlying hardware. The aim of this study is to develop a methodology that improves overall performance by culling inaudible and perceptually less prominent sound sources to be rendered in order to avoid performance implications. Through the use of distance attenuation and auditory masking, minimal decrease in perceived sound quality was intended to be achieved. The algorithm proposed in this paper is benchmarked and compared with the existing approaches to this problem. Subjective evaluation of the audio quality is also provided with the MUSHRA tests.

Keywords: audio source culling, perceptual audio optimization

# ÖZ

## SANAL ORTAMLAR İÇİN ALGISAL SES KAYNAĞI KESİMİ

, Ali Can Metan

Yüksek Lisans, Modelleme ve Simülasyon Bölümü

Tez Yöneticisi : Doç. Dr. Hüseyin Hacıhabiboğlu

Mart 2016 , 59 sayfa

Günümüzde kullanılan oyun motorları ve sanal ortam yazılımlarında uzamsal ses sentezi için çeşitli teknikler kullanılmaktadır. Bu tekniklerden biri, "baş ilişkili transfer fonksiyonu" ve yapay yankılandırıcıların bir arada kullanımı ile sağlanır. Herhangi bir sanal ortam için, çok sayıda uzamsal ses kaynağını bu yöntemlerle sentezlemek, altta bulunan donanım için bir performans gerektiren bir işlemdir. Çalışmamızın amacı, programın genel performansı geliştiren bir metodoloji geliştirerek, duyulamayan ve algısal olarak değeri az olan ses kaynaklarını keserek, performans artışı sağlamaktır. Uzaklığa bağlı ses azaltması ve işitsel maskeleme kullanılarak, algılanan ses kalitesinde minimum düşüş yaşanması hedeflenmiştir. Bu tezde önerilen işlemsel süreç, var olan diğer yaklaşımlara göre karşılaştırılmalı olarak değerlendirilmiştir. İşlemsel sürecin öznel olarak değerlendirilmesi de MUSHRA testleri ile sağlanmıştır.

Anahtar Kelimeler: ses kaynağı kesimi, algısal ses optimizasyonu

to those who never stop learning



## ACKNOWLEDGMENTS

I would like to thank my supervisor Assoc. Prof. Dr. Hüseyin Hacıhabiboğlu for suggesting and guiding what has turned out to be a really interesting research topic. He has shared his knowledge generously and his encouragement is much appreciated. Including my supervisor, I would like to thank members of my thesis committee for their insightful recommendations.

Many thanks to my family for their continuous support and mentorship throughout my study.

Finally, I would like to offer my special thanks to Ezgi Tolu. I could not have finished this thesis without her encouragement and friendship.

# TABLE OF CONTENTS

ABSTRACT . . . . .	vi
ÖZ . . . . .	vii
ACKNOWLEDGMENTS . . . . .	ix
TABLE OF CONTENTS . . . . .	x
LIST OF TABLES . . . . .	xiv
LIST OF FIGURES . . . . .	xv
LIST OF ALGORITHMS . . . . .	xvii
LIST OF ABBREVIATIONS . . . . .	xviii
CHAPTERS	
1 INTRODUCTION . . . . .	1
1.1 Motivation . . . . .	1
1.2 Organization of the Thesis . . . . .	1
2 BACKGROUND INFORMATION . . . . .	3
2.1 Human Auditory System . . . . .	3
2.1.1 Physiology of Human Auditory System . . . . .	4
2.2 Sound Metrics . . . . .	5

2.2.1	Monaural Hearing . . . . .	5
2.2.2	Binaural Hearing . . . . .	7
2.3	Audio Rendering in Virtual Environments . . . . .	8
2.3.1	Head Related Transfer Functions (HRTFs) . . . . .	8
2.3.1.1	Computational Cost of Binaural Audio in Virtual Environments . . . . .	9
2.3.2	Distance Attenuation . . . . .	9
2.4	Sound Source Culling . . . . .	9
2.4.1	Position Based Culling Methodologies . . . . .	9
2.4.2	Perception Based Culling Methodologies . . . . .	11
3	PROPOSED APPROACH . . . . .	13
3.1	Overall Algorithm . . . . .	13
3.2	Offline Analysis . . . . .	14
3.2.1	Systematic Limitations . . . . .	14
3.2.2	MPEG-1 Psychoacoustic Model, Layer I . . . . .	15
3.2.2.1	FFT Analysis . . . . .	16
3.2.2.2	Determination of SPL . . . . .	16
3.2.2.3	Finding of Tonal and Non-Tonal Components . . . . .	16
3.2.2.4	Decimation of Tonal and Non-Tonal Components . . . . .	17
3.2.2.5	Calculation of Individual Masking Thresholds . . . . .	17

	3.2.2.6	Calculation of the global masking threshold LTg . . . . .	18
	3.2.2.7	Determination of the Minimum Masking Threshold . . . . .	19
	3.2.3	Generation of Masking Information . . . . .	19
	3.2.4	Infeasibility of Precalculated Decision Making . . . . .	19
3.3		Real-Time Analysis . . . . .	19
	3.3.1	Concept of Auditory Events and Event Manager . . . . .	20
	3.3.2	Decaying Event Priority Queue . . . . .	22
	3.3.2.1	Hash Functions For Variable Length String Comparison . . . . .	24
	3.3.3	Timed Circular Buffer . . . . .	24
4		PERFORMANCE AND ANALYSIS . . . . .	31
	4.1	Theoretical Information . . . . .	31
	4.1.1	Algorithmic Complexity of Real-Time Analysis . . . . .	31
	4.1.2	Factors That Affect The Cost of Real-Time Analysis . . . . .	31
	4.2	System Performance . . . . .	32
	4.2.1	Individual Cost of Real Time Analysis . . . . .	32
	4.2.2	Performance Gain From a Single Audio . . . . .	33
	4.2.3	Worst Case Scenario . . . . .	34
5		PSYCHOACOUSTICAL EVALUATION . . . . .	35
	5.1	Preliminaries . . . . .	35

5.2	Chosen Test Methodology . . . . .	35
5.3	Test Procedure . . . . .	35
5.3.1	Presentation of Stimuli . . . . .	35
5.3.2	Grading . . . . .	36
5.4	Experiment Details . . . . .	36
5.4.1	Selection of Sound Source Locations . . . . .	37
5.4.2	Generation of Test Cases . . . . .	38
5.5	Statistical Analysis . . . . .	39
6	CONCLUSION . . . . .	45
6.1	Contributions and Discussion . . . . .	45
6.2	Future Work . . . . .	46
	Bibliography . . . . .	47
APPENDICES		
A	REAL TIME ANALYSIS PSEUDOCODE . . . . .	51
B	EXPERIMENT ORDERS . . . . .	57
C	APPROVAL OF ETHICS COMMITTEE . . . . .	59

## LIST OF TABLES

### TABLES

Table 3.1	Overlapping FFT Frames . . . . .	16
Table 4.1	Average Times Required to Complete Real Time Analysis . . .	33
Table 4.2	Costs Of Execution For an Integrated Audio Rendering Pipeline	33
Table 5.1	Contents of Test Scenes . . . . .	37
Table 5.2	Test Setup, Positional Information . . . . .	39
Table 5.3	Mean Values . . . . .	41
Table 5.4	Results of Independent-Samples t-test . . . . .	43
Table B.1	Psychoacoustical Test Order . . . . .	57

# LIST OF FIGURES

## FIGURES

Figure 2.1	Cross Section of Human Ear . . . . .	3
Figure 2.2	Signals and Basilar Membrane . . . . .	4
Figure 2.3	Absolute Threshold of Hearing . . . . .	6
Figure 2.4	Sample Masking Thresholds . . . . .	7
Figure 2.5	Interaural Time Difference . . . . .	7
Figure 2.6	Example Audio Rendering Pipeline . . . . .	8
Figure 2.7	Head Related Transfer Function . . . . .	9
Figure 2.8	Position Based Culling Illustration . . . . .	10
Figure 3.1	Overall Culling Pipeline . . . . .	13
Figure 3.2	Offline Analysis Overview . . . . .	14
Figure 3.3	Real Time Analysis, Auditory Event Activity Diagram . . . . .	21
Figure 3.4	Event Manager, Auditory Event Handling . . . . .	22
Figure 3.5	Event Manager, End-Of-Audio Event Handling . . . . .	23
Figure 3.6	Decaying Event Priority Queue, Example Execution . . . . .	25
Figure 3.7	Timed Circular Buffer, Example Execution . . . . .	29
Figure 5.1	Mushram User Interface . . . . .	36

Figure 5.2 Source Positions For Tested Scenes . . . . .	38
Figure 5.3 Subjective Scores From Listening Tests . . . . .	40
Figure C.1 Approval of Ethics Committee . . . . .	59



# LIST OF ALGORITHMS

## ALGORITHMS

Algorithm 1	Event Manager, Auditory Event Handling . . . . .	52
Algorithm 2	Event Priority Queue, Enqueuing . . . . .	53
Algorithm 3	Timed Circular Buffer, Handling Masking Values, Part 1	54
Algorithm 3	Timed Circular Buffer, Handling Masking Values, Part 2	55

## LIST OF ABBREVIATIONS

ANOVA	Analysis Of Variance
BM	Basilar Membrane
CGMT	Circular Global Masking Threshold
DEPQ	Decaying Event Priority Queue
EM	Event Manager
GPL	General Public License
(G)UI	(Graphical) User Interface
HDD	Hard Disk Drive
HRTF	Head Related Transfer Functions
HRIR	Head Related Impulse Response
ITD	Interaural Time Difference
ILD	Interaural Level Difference
IID	Interaural Intensity Difference
LPCM	Linear Pulse-Code Modulation
WAVE or WAV	Waveform Audio File Format
PCM	Pulse-Code Modulation
SIL	Sound Intensity Level
SNR	Signal-to-Noise Ratio
SSD	Solid State Drive
SPL	Sound Pressure Level
TCB	Timed Circular Buffer

# CHAPTER 1

## INTRODUCTION

### 1.1 Motivation

Virtual environments create a perception of being physically present in a non-physical world. Two main components for achieving this, are presentation of auditory and visual stimuli. With today's technology, creation of vivid environments require significant computational power. As such, optimization of any existing processes will make better use of the limited resources.

Dealing with the complexity of rendering spatialized audio in large quantities will impose a performance penalty for the underlying hardware. Synthesizing spatial audio through the use of head related transfer functions is a common methodology used in today's systems. To deal with the performance implications, most game engines and virtual environment software that are in use today, employ distance based culling methodologies. While these methodologies effectively reduce the amount of sound sources, perceived richness of the resulting scene will be compromised.

In order to cope with the performance requirements, while preserving the perceived sound quality, audio source culling through the use of auditory masking is proposed in this thesis. With this methodology, perceptually inaudible sound sources will not be rendered and perceived richness will be maintained.

Most games and virtual environments include loud and broadband sound sources. Explosions, gun shots, engine sounds and other loud sources mask one another. In those cases, proposed algorithm will be able to exploit this psychoacoustical feature. While performing psychoacoustical analysis for every sound requires more system resources, they provide a good balance between subjective audio quality and performance gain. In the remaining parts of this thesis, we will investigate this approach.

### 1.2 Organization of the Thesis

Thesis is divided into 8 chapters. Following this introductory part is the chapter 2, which presents background information about spatial audio synthesis. Chapter 3 includes explanations of psychoacoustical analysis and the proposed algorithm. Chapter 4 details performance implications of the proposed algorithm. Chapter 5 provide information on psychoacoustical evaluation of the

audio produced with the methodology proposed. Finally, chapter 6 summarizes the thesis and gives a discussion of our work. After these sections, pseudocode of the algorithm and the details on the subjective experiments can be found in the appendix section.

## CHAPTER 2

### BACKGROUND INFORMATION

This chapter explains certain concepts of sound, how human auditory system works, how binaural audio is rendered and literature related to this research.

#### 2.1 Human Auditory System

Human auditory system converts atmospheric pressure caused by sound waves into neural excitations. This section briefly describes how human auditory system works.

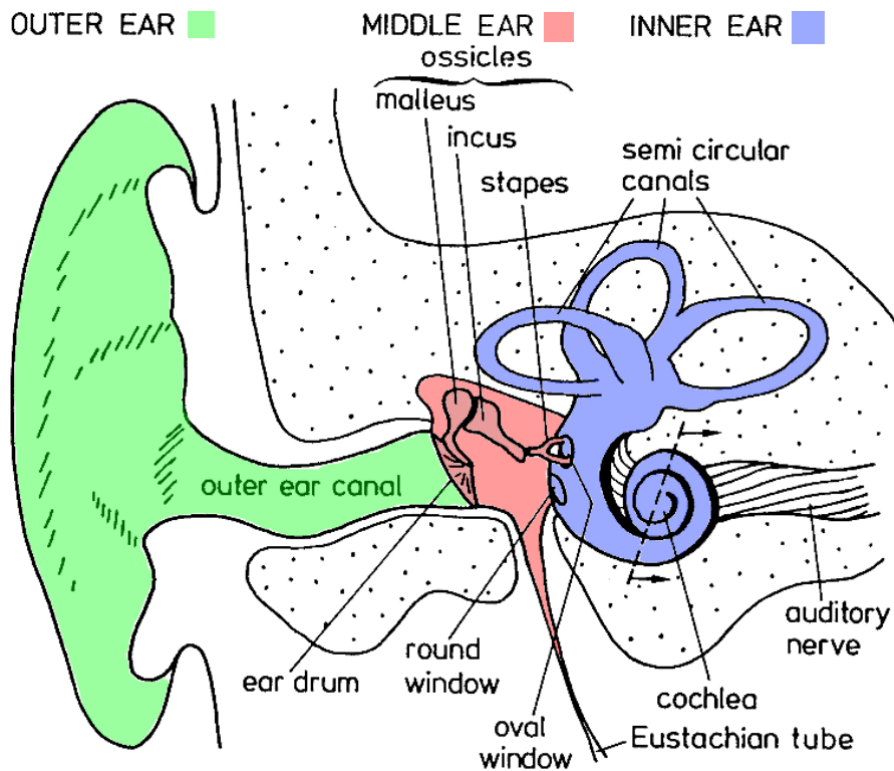


Figure 2.1: Cross Section of Human Ear [42]. Modified with permission from Springer.

### 2.1.1 Physiology of Human Auditory System

**Extremities:** Contrary to common belief, hearing functionality of the human auditory system is affected by the body components close to the ear. In an acoustic free field, the sound pressure level reaching to an ear drum is influenced by extremities such as shoulders, shape of the head, as well as the clothing people wear [31][41]. These parts of the human body influence our hearing by causing reverberations and shadowing.

**The Outer Ear:** Its main function is to direct sound waves through the outer ear canal, onto the ear drum [5][42]. Sound waves enter through the pinna, which modifies the sound waves depending on the direction of the sound. Then the waves travel through the ear canal to reach the ear drum. Outer ear also protects the middle ear from being damaged by external elements.

**Middle Ear:** Middle ear starts with a thin membrane called the ear drum. Unlike outer ear, middle ear starting from this ear drum contains fluids. There are three bones attached to this membrane; the malleus, incus and stapes. Pressure changes in the ear drum are transferred by these bones to the oval window, which is the entrance to the inner ear. Sound pressure reaching to the oval window (through stapes) is amplified by a factor of about 20 due to the size difference between ear drum and oval window [24].

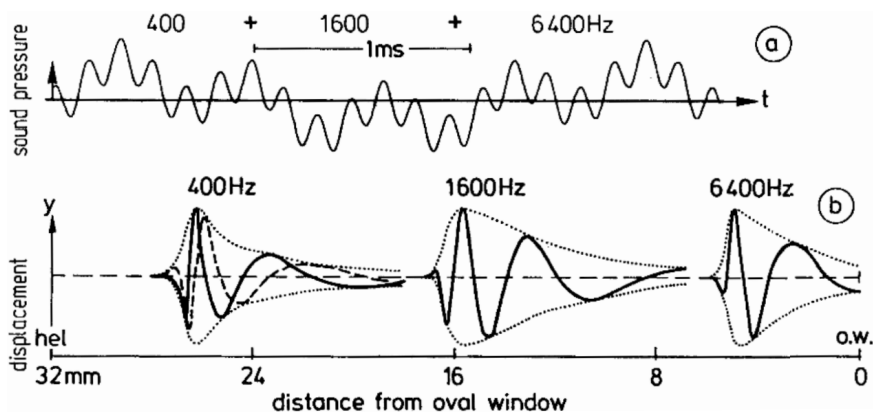


Figure 2.2: A signal that has three tonal components traveling on basilar membrane. Various frequencies create peaks at corresponding locations along the BM, causing different neural excitations. [42] Reprinted with permission from Springer.

**Inner Ear (cochlea):** Because of its coiled up shape, cochlea looks like a snail. It is embedded deep into the temporal bone and it contains two types of fluids and three scalae [42][19]. The stapes is in direct contact with the fluid in scala vestibuli. Pressure differences coming from the stapes are transferred to the basilar membrane (BM) and they cause it to vibrate. Segments of the basilar membrane act like a bandpass filter, causing partial neural excitations along its narrowing structure (see figure 2.1). Located on the basilar membrane is the organ of Corti. It is responsible of converting mechanical oscillations into signals

that can be processed by the nervous system.

## 2.2 Sound Metrics

Before we get into details of how hearing works, there are concepts of sound that needs to be known. This section explains some measurements of sound.

**Sound Pressure Level and Sound Intensity:** Before we get into details of how hearing works, we should know what sound pressure level (SPL) and sound intensity are. SPL is a logarithmic unit, used to express the pressure of a sound relative to a reference value [42]. Sound intensity on the other hand, is defined as the sound power per unit area. Sound intensity, denoted by  $I$  and measured in  $Wm^2$ . These terms are related and are defined by the following formula;

$$L = 20 \log(p/p_0) \text{ dB} = 10 \log(I/I_0) \text{ dB}$$

Where  $p_0$  is the reference value of sound pressure, which is standardized to  $p_0 = 20\mu Pa$ . The reference value  $I_0$  is defined as  $10^{-12}W/m^2$  [42].

**Critical Bandwidth and the Bark Scale:** Critical bandwidths represent the frequency resolution of the auditory system [36]. Signals that are in the same critical bandwidth have the potential to mask one another (auditory masking is explained in 2.2.1).

Bark scale is a frequency scale on which a given frequency denote the width of the critical band [36].

$$z = \frac{26.81}{1 + (1960/f)} - 0.53 \text{ Bark}$$

Details on how to calculate the bandwidth and its application is explained in the auditory masking section 2.2.1 and offline analysis section 3.2.2.5.

### 2.2.1 Monaural Hearing

**Range Of Human Hearing** Human range of hearing is commonly stated to be in between 20 Hz and 20 kHz [28][29]. Although some research support that it could range between 12 Hz [22] and 28 kHz for the young [2]. It should be noted that the sound pressure which we can hear these frequencies are not the same;

#### Absolute Threshold of Hearing (ATH)

Also known as threshold in quiet, ATH denote the required SPL at which pure tones are barely audible [42]. Figure 2.3 shows the threshold in quiet. For example a pure tone at 0.1 kHz with 10 dB SPL will not be audible humans. Natural sounds we hear are composed of multiple spectral components. Hence, some frequencies of quieter sounds may not be heard, depending on their loudness. Fig. 2.3 shows spectral regions of speech and music signals, as well as thresholds of pain and damage risk.

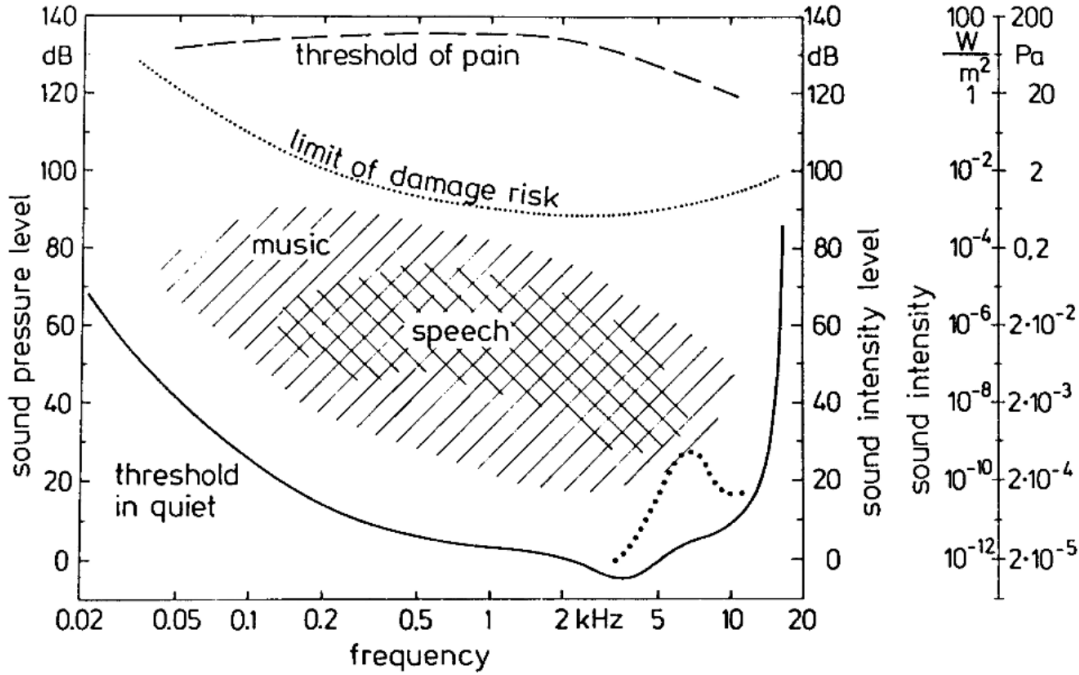


Figure 2.3: Absolute Threshold of Hearing, dotted lines on the bottom denote hearing loss due to exposure to loud sound [42]. Reprinted with permission from Springer.

### Auditory Masking

Auditory masking is a common phenomenon that happens in everyday life. For example when two are people having a conversation, if somebody turns on a really loud music, those two people will not be able to hear each other anymore. They can either raise their voices to increase sound pressure or turn the music down, so that the other person will be able to hear them. In this case, music acts as a masker signal for the two speech signals. This phenomenon is called auditory masking.

Auditory masking happens on three stages, backward masking (before the masker is presented), simultaneous masking (in presence of the masker) and forward masking (after the masker is removed) [42]. For this paper, we will focus on simultaneous masking.

Concept of simultaneous masking is similar to that of threshold in quiet. Presence of a sufficiently loud pure tone or a signal with a certain bandwidth will change this threshold of hearing [42]. The change of masked threshold depends on type of the masker and the SPL of the masker. Examples of such cases are demonstrated in the figure 2.4.

Graphs demonstrated in figures are one such example of threshold calculation. Other mathematical models for calculating masked thresholds exist [20]. As a matter of fact, masking models are actively used in audio compression that we



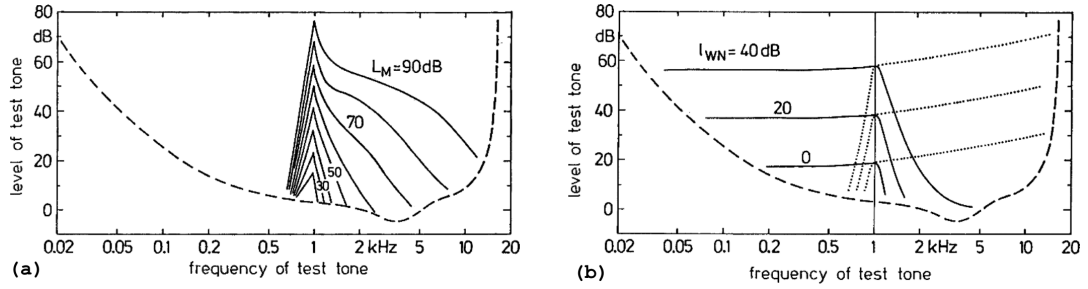


Figure 2.4: **a)** Masking thresholds of a pure tone masker with varying SPLs. **b)** Masking thresholds for low pass-noise (solid curves) and high pass noise (dotted curves) [42]. Reprinted with permission from Springer.

use today. By representing the original audio with as low bit rate as possible while rendering quantization noise inaudible, audio compression can be achieved [32]. A well known compressed audio format mp3, is widely used today.

The psychoacoustical model employed by MPEG layer 1 is utilized in this thesis [16]. Further information regarding the masking threshold calculation can be found in 3.2.2 of this thesis.

### 2.2.2 Binaural Hearing

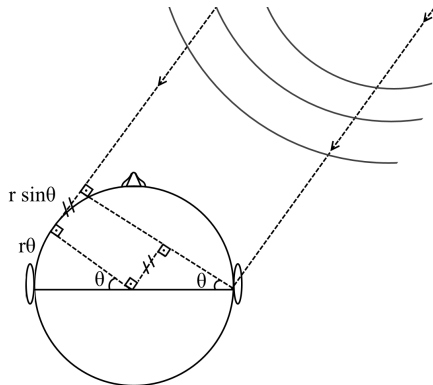


Figure 2.5: Interaural Time Difference

When it comes to binaural hearing, we are interested in sound source localization. Duplex theory of sound localization proposes that there are two factors that enable this[27]. Time differences (ITDs) and intensity differences (ILDs) of sounds reaching to each ear.

**Interaural Time Difference:** Assume a situation like in the figure 2.5. Width of average human skull can be taken as 18 cm wide [12]. So a sound wave coming from an angle  $\theta$  will arrive at each ear on different times. Time difference can be calculated with

$$ITD = \frac{r(\theta + \sin(\theta))}{c}$$

Where  $c$  is the speed of sound. Though there are limits as to where ITD is effective. For a symmetrical human head, any sound sources on the median sagittal plane will not cause this difference, hence different queues will be required [30]. Secondly there is an upper frequency limit for the sound signal to resolve the time difference. Frequency limit which this phenomena occurs is dependent on head width.

**Interaural Level Difference:**

Also known as the interaural intensity difference (IID), ILD is best utilized for higher frequency range of hearing [30]. SPL of the sound signals reaching to each ear will be different due to the shadowing of the head. As with the figure 2.5, the bigger the angle  $\theta$  is, the bigger ITD will get. Listeners will perceive this difference as differences in loudness[30].

### Contextual Difference:

The context of the sound itself might make a perceptual disparity for the angles perceived. For example people would expect bird sounds coming from above, or walking/step sounds to be coming from below. Emotional state of the subject can also affect this perception as people traveling in the woods will tend to look behind them, in the case of cone of confusion.

## 2.3 Audio Rendering in Virtual Environments

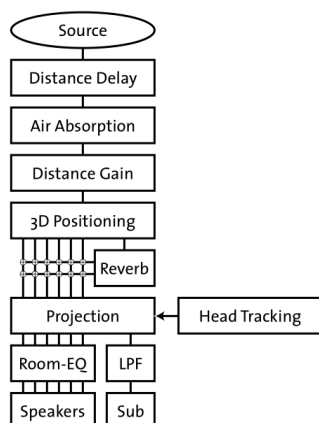


Figure 2.6: Example Audio Rendering Pipeline [21].

Although there are procedurally generated audio in virtual environments, using prerecorded, canned audio such as Foley effects and voice overs are predominantly used. Live inputs that come from a network are rarely utilized in virtual environments. Typically game engines will read prerecorded audio data into volatile memory and process them before finally outputting them to sound buffer. An example of a audio rendering pipeline can be seen in figure 2.6 [21]. In what follows, some of the processing stages which result in most of the computational complexity will be discussed.

### 2.3.1 Head Related Transfer Functions (HRTFs)

There are multitude of ways to synthesize binaural spatial audio. Earlier methodologies included approaches such as amplitude panning to simulate 3-D audio[6]. While they provided a tangible solution, realistic results were not achieved. Later on, head related transfer functions that included ITD and ILD cues, are considered to provide a more accurate and realistic solution[13][1]. While there are realistic approaches that still include panning with delays [15] [26], usage of HRTFs are still prevalent.

HRTFs map how a sound arrives from a specific point, in a specific medium, to the outer end of the ear canal [3]. Different spectral properties of the system is captured with the impulse response, forming the Head Related Impulse Response (HRIR) [3]. Fourier transform of HRIR forms the HRTF. HRTFs include effects such as shadowing of the head, sound attenuation from pinnae and sometimes reverberations coming from the shoulder. Since these factors are unique to each individual, a set of HRTF only maps the transfer function specific to the subject measured [3]. However they still provide a good approximation as to how the listener would hear the same sound, with the given parameters.

### 2.3.1.1 Computational Cost of Binaural Audio in Virtual Environments

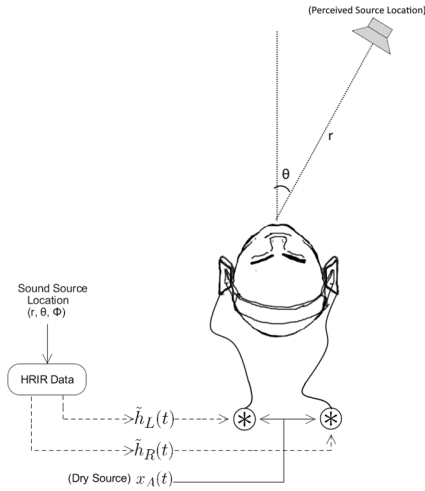


Figure 2.7: Head Related Transfer Function

Synthesizing spatial binaural audio with HRTFs can be accomplished in two ways. Either a convolution of the original sound source with the HRIR, or multiplication of the frequency response HRTF. For an output size  $N$ , convolution in time domain has a complexity of  $O(N^2)$ . In order to reduce computational complexity, taking fast fourier transform (FFT) of each sequence, multiplying them point-wise and computing the inverse FFT is performed. This operation still takes  $O(N \log(N))$  (REF).

Above method calculates the resulting audio of a single binaural audio in a 3-D environment. There is also the added cost of distance attenuation along with the cost of calculating reverberation of the synthesized audio.

### 2.3.2 Distance Attenuation

Given a reference intensity and distance, an omnidirectional sound source's intensity will fall almost exactly 6 dB for each subsequent doubling of distance from a source [3]. Distance attenuation requires same number of multiplications of sampling frequency of the original audio. In our case, that would amount to 44100 multiplication per seconds for each audio.

## 2.4 Sound Source Culling

For applying source culling to the prerecorded sound sources in a scene, various auditory culling methodologies are used today. They can be classified into two categories, position based approaches and perception based approaches.

### 2.4.1 Position Based Culling Methodologies

Position based culling methodologies only take the position of the listener and preset properties of the auditory source into account. Algorithm itself is not concerned with the context of the audio itself. As all the properties of the audio, including the culling methodology, is handled by the designer of the scene.

Commercial game engines in use today exclusively use this methodology [35][11][8]. The reason being that position based culling provide the best performance when it comes to the cost of culling the audio itself. However numerous sound sources being in acceptable positions would nullify this performance gain. It will be discussed later in this section.

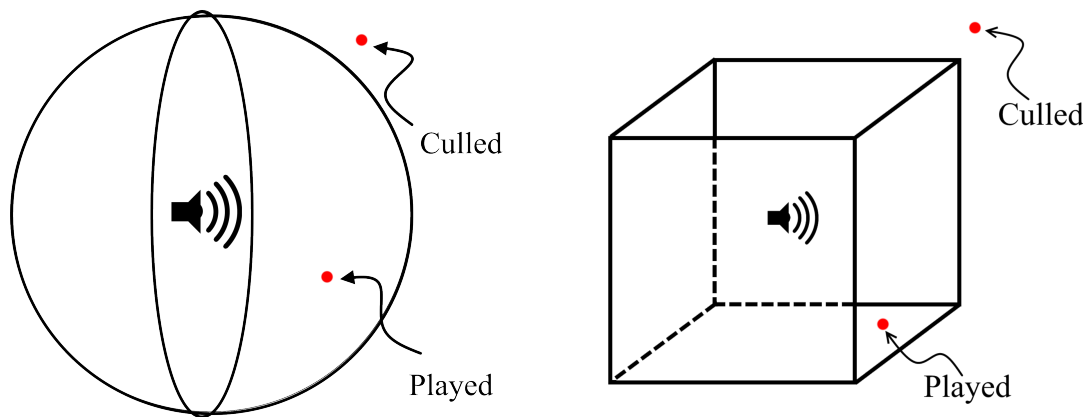


Figure 2.8: Illustration of Position Based Culling. Red dots denote positions of the listener while geometric shapes denote culling volumes assigned to sound sources.

**How position based culling methodology works:** Designer of each scene, assigns each audio source a geometric culling shape like a cone, a cube or a sphere. They can also determine distance attenuation according this shape or assign natural sound attenuation stated just like above. When the scene is launched and when an auditory event comes, engine checks whether the listener is inside this volumetric shape or not. If the listener is not inside this shape, audio is culled and will not be rendered. A spherical culling volume, for example, is simply an euclidean distance culling methodology.

Advantage of this methodology comes from the performance cost of making a culling decision. It is fast and effectively reduce active audio sources.

There are several possible disadvantages for this methodology.

1. *Culling of distant and loud sound sources:* As the spectral properties of sound sources are not incorporated, loud and distant sound sources normally would've been heard, will be culled. It is up to the developer to adjust culling distance and attenuation according to the contents of each sound. So it forces you to manually test and adjust attenuation for each sound.
2. *Crowded scenes:* If the listener is placed in a crowded sound field, reduction of active sound sources is not met by this methodology. A separate algorithm would be required to render crowded scenes with high fidelity and efficiency.
3. *Erroneous Auditory Environments:* Since both culling volumes and sound attenuation of audio sources are up to game developers' to decide, it can lead to erroneous auditory fields. Some sound sources can be unnecessarily silent (which decreases audio fidelity) or unrealistically loud. Maladjustment of culling distances will mean either unnecessary rendering of sounds or not playing the sound when needed.

### 2.4.2 Perception Based Culling Methodologies

Perceptual approaches utilize various aspects of human sensory processing.

There are those who utilize visual perception to cull sound sources. More commonly referred as crossmodal effects, these algorithms cull sound sources that are outside the view frustrum of the listener [14][18].

Also there exists one methodology that will cull sound sources according to their estimated audibility. There haven't been much research on this subject but a paper that is somewhat similar to our methodology is present [37]. For each auditory frame, a binaural loudness estimation for each source in the scene is made. Sources are then sorted according to their projected loudness. After the first source that is deemed inaudible is reached, rest of the sources for that frame are not rendered. After the remaining operations are completed for the selected audio, results are mixed to be played to the listener [37].

Methodology proposed in this paper is an auditory perception based culling methodology. Details of the proposed algorithm can be found in the remaining of this thesis.



## CHAPTER 3

### PROPOSED APPROACH

Dealing with the complexity of rendering spatialized audio in large quantities in real time becomes taxing for any virtual environment. In order to cope with the performance requirements, while maintaining the perceived sound quality, audio source culling through the use of auditory masking is proposed in this thesis.

The algorithm proposed in this chapter, consists of two parts: offline analysis and real time analysis. A diagram of the proposed algorithm is shown in figure 3.1

#### 3.1 Overall Algorithm

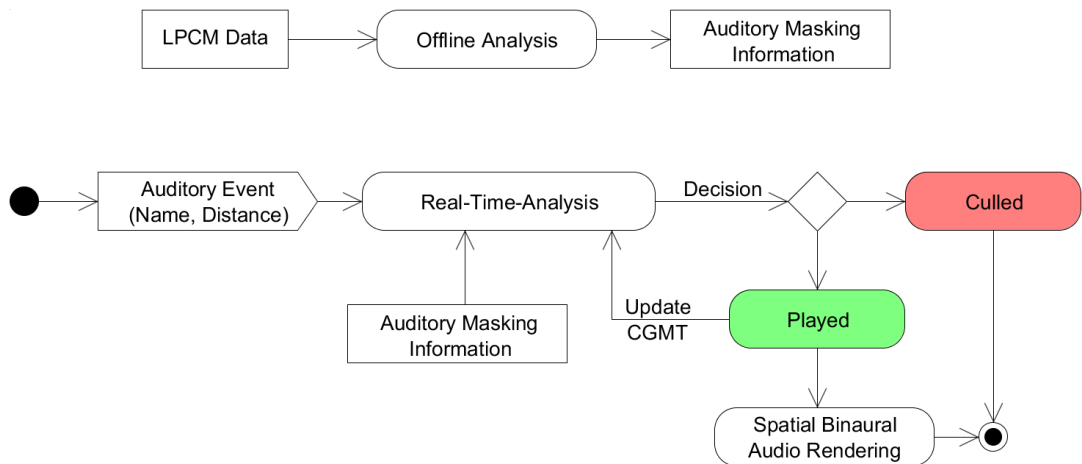


Figure 3.1: Overall pipeline for the culling algorithm

Offline analysis is where the psychoacoustical information is extracted from each sound source in the scene. This process happens prior to the execution of the real program and the masking thresholds of each audio is calculated and stored separately.

Data extracted from the offline analysis are then used in the decision making process of the real time analysis stage. This is also the stage where the calculation of the global masking threshold and auditory source culling takes place.

After the decision to render the audio is made by the algorithm, circular global masking threshold (CGMT) is updated and an external audio engine can then handle the rest of the rendering process.

Details of each process mentioned here, are elaborated in detail in the remaining sections of this chapter.

## 3.2 Offline Analysis

The psychoacoustical data extraction of the audio sources contained within a scene, starts with the offline analysis. As the name suggests, we perform this analysis before execution of the actual program takes place. On this step, masking thresholds are calculated from the given sound source signal, which is to be used in the real time analysis stage to decide whether a given audio shall be rendered or not. A visual representation is given in figure 3.2.

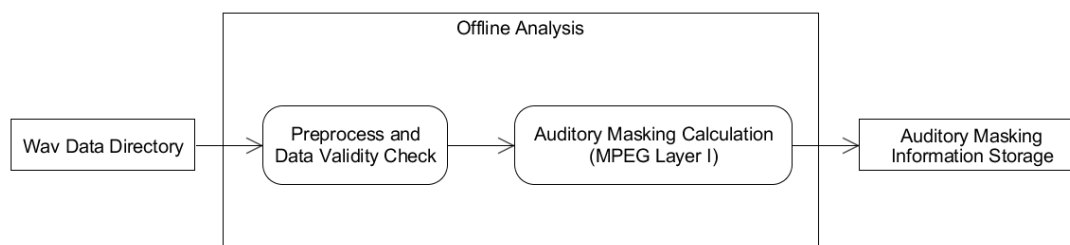


Figure 3.2: Brief overview of the offline analysis

Before the directory where the audio files are stored is prepared for the output files, validity of the input those files are controlled according to system limitations. Details of the system limitations are stated below, in 3.2.1 of this thesis.

After the validity of the input data files are confirmed, we proceed to the generation of the masking thresholds for each sound file according to the MPEG-1 Psychoacoustic Model I. This is where the tonal and non-tonal components of the input signal is estimated. After relevant maskers have been selected among those components, **local** masking thresholds are calculated and stored in memory. Details of how the calculation of auditory masking thresholds performed is stated in 3.2.2 of this thesis. The explanation of how and at what stage of the analysis the data is stored is also explained in the section 3.2.3.

### 3.2.1 Systematic Limitations

There are four constraints on the input signals that are to be analyzed.

*Firstly, the audio file must sampled and quantized for LPCM.* For the purposes for our algorithm WAVE file format (.wav) was used. Perhaps other file formats whose bitstream encoding is LPCM could also have been used within the MPEG's psychoacoustic model, thus in our algorithm. However WAVE file for-



mat have sufficed for our purposes and including other audio file formats are out of the scope of this research.

*Secondly, the sampling frequency for every signal must be 44100 Hz.* This was done due to the limited number of sampling frequencies provided by MPEG's psychoacoustic model, albeit highest frequency for of human ear can hear is roughly 20,000 Hz [42]. So according to the Nyquist–Shannon sampling theorem [25], sampling frequency should be at least:

$$F_s > 20,000 * 2 = 40,000Hz$$

*The third requirement, is the condition that the bit depth of the input signal must be 16.* Higher bits per sample would yield to higher signal-to-quantization noise ratio (SQNR) values, hence a more accurate representation of the masking threshold could have been achieved. As the output of the offline analysis includes 64 bits per sub-band masking values. But for our purposes, ease of implementation and faster analysis we chose bit depth to be 16.

*Fourth requirement, is the condition of the input audio signal to be monophonic.* While it is totally possible to process two channeled stereo recordings for a modified version of the proposed algorithm, it is also unnecessary. Left and right channels of the audio can be processed separately, producing twice the amount of offline data. These two files can then be assessed psychoacoustically with the other two global masking buffers of the real time algorithm, which would also require twice as much operations in real time analysis. So we deem this approach unnecessary for our purposes and prohibit multi-channel audio as its input.

*Fifth and the final requirement, is that the analyzed audio file cannot be longer than 5 seconds.* This is a requirement comes from the real time analysis stage, which can be changed according to the hardware. Details and reasons behind this will be explained in the global masking threshold section, 3.3.3.

### **3.2.2 MPEG-1 Psychoacoustic Model, Layer I**

MPEG's audio compression algorithm is intended to be used as the first international standard for the digital compression of high-fidelity audio[16][33]. A constituent of the algorithm, is the psychoacoustic model which is used to calculate signal-to-mask-ratio (SMR) for all subbands. SMR is then compared with the signal-to-quantisation-noise-ratio (SQNR) to determine the audibility of the coding noise for the compression to take place[16].

For our purposes, the psychoacoustic model is used to determine whether an incoming audio is audible inside the existing scene or not. For convenience, we have used an existing implementation of the psychoacoustic model for our purposes [23].

Following sections describe how to retrieve simultaneous masking information from a fixed sample rated digital audio signal, using MPEG-1's Psychoacoustic Model, Layer 1.

### 3.2.2.1 FFT Analysis

For the MPEG Layer I, audio samples are is divided into 512 sample frames and each frame goes under psychoacoustic evaluation separately.

Layer 1 model accounts for both the delay of the audio data through the filter bank and an offset from the Hann window to coincide with the samples of the analysis frame. There are 32 bands coming from the polyphase filter bank and blocks of 12 samples were formed in each subband ( $32 * 12 = 384$ ). A new bit allocation is calculated for each 384 sample. The delay coming through the filter bank is 256 samples, as well as the  $(512 - 384)/2 = 64$  offset delay is required to center, making a total of 320 point initial offset. Before performing a Fourier transform of length 512 on the data, a standard Hann weighting is applied.

Table3.1: 512 sample FFT Frames. 64 samples from the previous frame and 64 samples from the next frame are overlapped

...	...	384 samples	...	...
n-64	n		n+383	n+447

Frequency resolution (for our system) is

$$f_s/512 = 44100/512 = 86.1328125 Hz$$

Hann window,  $h(i)$ :

$$h(i) = 0.5 * [1 - \cos(2\pi(i)/511)] \quad \text{where} \quad 0 \leq i \leq 511$$

Also the power density spectrum  $X(k)$  is calculated as follows:

$$X(k) = 20 * \log|1/Nh(l)s(l)e^{-jkl\frac{2\pi}{N}}| \text{dB} \quad k = 0..N/2$$

### 3.2.2.2 Determination of SPL

The sound pressure level (SPL) in subband n is computed with the following formula:

$$Lsb(n) = MAX[X(k), 20 * \log(scfmax(n) * 32768) - 10]dB$$

Where k is the frequency index and  $X(k)$  in subband n. "scfmax(n)" is in Layer I the scalefactor. The sound pressure level  $Lsb(n)$  is computed for every subband n.

### 3.2.2.3 Finding of Tonal and Non-Tonal Components

MPEG Layer 1 separates the tonal and noise-like components of the audio signal. The reason relies under the different spreading characteristics of simultaneous masking by these two types of maskers.

This model identifies tonal components  $X(k)$ , based on the local maxima which is determined by:

$$X(k) > X(k - 1) \quad \text{and} \quad X(k) \geq X(k + 1)$$

After that, for each critical band, the remaining noise-like components are summed into a single non-tonal component. A local peak is added into a list of tonal components if:

$$X(k) - X(k + j) \geq 7dB,$$

where  $j$  is chosen differently for different frequency bands such that:

$$\begin{aligned} j = -2, +2 & \quad \text{for} \quad 2 < k < 63 \\ j = -3, -2, +2, +3 & \quad \text{for} \quad 63 \leq k < 127 \\ j = -6, \dots, -2, +2, \dots, +6 & \quad \text{for} \quad 127 \leq k \leq 250 \end{aligned}$$

Remaining spectral lines are calculated to form non-tonal components. For the data that has 44.1kHz sampling rate, there are 24 critical bands which is determined from a fixed table stated in the standard [16]. For each critical band, SPL of the new non-tonal component is calculated by summing the power of the spectral lines corresponding to that critical band.

### 3.2.2.4 Decimation of Tonal and Non-Tonal Components

Unrelated maskers are determined and eliminated in this step. There are two conditions to determine whether a masker is eliminated or not.

Firstly, any tonal or non-tonal component which generates masking thresholds that are below the absolute threshold of hearing are eliminated. As these components will not be audible due to the threshold [16].

Secondly, all less powerful tonal components within the distance of less than 0.5 Bark. Components with the highest power are kept in the list of tonal components and smaller ones are eliminated. Remaining components will be used for masking threshold calculation.

### 3.2.2.5 Calculation of Individual Masking Thresholds

For the masking threshold calculation, only a subset of samples are utilized. Here is how subsampling is used for an audio that has 44.1 kHz sampling rate:

- First 6 subbands: no subsampling
- Second 6 subbands: every second spectral line is used.
- Remaining subbands, every fourth spectral line is used (up to  $f = 20$  kHz).

So the number of samples,  $i$ , in the subsampled frequency domain is 106 for 44.1kHz sampling rate for the Layer 1. MPEG Layer 1 uses 12 samples per subband so there are:

$$384/12 = 32 \text{ subbands}$$

The individual masking thresholds of the selected components are calculated with the following expression:

$$LT_{tm}[z(j), z(i)] = X_{tm}[z(j)] + av_{tm}[z(j)] + vf[z(j), z(i)]dB$$

$$LT_{nm}[z(j), z(i)] = X_{nm}[z(j)] + av_{nm}[z(j)] + vf[z(j), z(i)]dB$$

Where "tm" stands for tonal masker and "nm" stands for non-tonal masker.  $LT_{nm}$  and  $LT_{tm}$  denote the individual masking thresholds at critical band rates in Bark, given in dB. The term  $X_{tm}[z(j)]$  is the SPL of the masking component with the frequency index  $j$  at the corresponding critical band rate  $z(j)$ .

Terms  $av$  and  $vf$  denote the masking index and is the masking function of the masking component  $X_{nm}[z(j)]$ , respectively.  $av_{tm}$  (index for tonal maskers) and  $av_{nm}$  (index for non-tonal maskers) are given by;

$$av_{tm} = -1.525 - 0.275 * z(j) - 4.5dB,$$

$$av_{nm} = -1.525 - 0.175 * z(j) - 0.5dB.$$

The masking function  $vf$  for both tonal and non-tonal maskers are given by;

$$\begin{array}{ll} vf = 17 * (dz + 1) - (0.4 * X[z(j)] + 6) \text{ dB} & \text{for } -3 \leq dz < -1 \text{ Bark} \\ vf = (0.4 * X[z(j)] + 6) * dz \text{ dB} & \text{for } -1 \leq dz < 0 \text{ Bark} \\ vf = -17 * dz \text{ dB} & \text{for } 0 \leq dz < 1 \text{ Bark} \\ vf = -(dz - 1) * (17 - 0.15 * X[z(j)]) - 17 \text{ dB} & \text{for } 1 \leq dz < 8 \text{ Bark} \end{array}$$

where  $dz$  is the distance in Bark:  $dz = z(i) - z(j)$ .

### 3.2.2.6 Calculation of the global masking threshold LTg

The global masking threshold for each frequency sample index  $i$  is denoted by  $LTg(i)$ . They are derived from the upper and lower slopes of the individual masking thresholds of each of both tonal and non-tonal maskers. Additionally from the precalculated threshold in quiet  $LTq(i)$  is also incorporated.

Formula for calculating the global masking threshold is just to sum the powers of individual masking thresholds and threshold in quiet.

$$LTg(i) = 10\log(10LTq(i)/10 + LT_{tm}[z(j), z(i)] + LT_{nm}[z(j), z(i)])$$

### 3.2.2.7 Determination of the Minimum Masking Threshold

For each subband  $n$ , the minimum masking threshold is calculated according to each frequency in the subband.

$$LTmin(n) = MIN[LTg(i)]dB$$

### 3.2.3 Generation of Masking Information

After every frame have been processed, masking thresholds of the 32 subbands (double format) are stored in the respective binary file. For a three second audio that has a sampling rate of 44.1 kHz, output will have  $\lfloor (44100 * 3) / 384 \rfloor = 344$  frames because of windowing. So the file size will be 64 bits \* 32 subbands \* 344 frames = 704512 bits which is equal to 86 kilobytes.

Offline analysis is concluded by storing the masking thresholds for each frame. Global masking threshold of the entire scene will be calculated in the real time analysis 3.3.

### 3.2.4 Infeasibility of Precalculated Decision Making

One could argue whether calculating the global masking threshold in real time and not simply pre-calculating and storing them is a feasible option or not. In a given scene, there can be as little as zero or one concurrent sound sources at a time. However masking can only occur after at least two concurrent sound sources are played at the same time, which is what we are interested in. There is no theoretical upper limit of the number of sound sources that can be played at a given time. Even though the number of sound files are fixed, since the same sound file can be played infinitely many times, making infinitely many possible combinations for a scene.

Let there be a scene with 30 sound files which are 4 seconds each. There are  $\lfloor (44100/384)*4 \rfloor = 459$  frames per sound file. If there are only two, unique sound files played at a time, there would be  $\binom{30}{2} * (459 + 459 - 1) = 398\,895$  possible combinations just for two concurrent sound sources. If there are only three unique sound files played at a time, there would be  $\binom{30}{3} * (459) * (459 + 459 - 1) = 1\,708\,866\,180$  combinations. We excluded multiples of the same sound sources to be played at a time which would even yield to a bigger sum.

So the possible combinations are increasing exponentially and the memory required to store all number of concurrent sound sources becomes infeasible with the contemporary hardware.

## 3.3 Real-Time Analysis

After the offline analysis was performed on the audio contained within the scene, and the required data was stored in memory, real time analysis becomes available to be performed during the program execution. The main purpose of the real time analysis, is to calculate the global masking threshold on the fly and identify

whether an incoming sound source is audible or not. Determining this will enable us to cull inaudible sound sources, hence reduce the computational overhead of rendering spatial audio.

Visual representation of the real time analysis, for handling an auditory event is shown in the figure 3.3.

Calculation of the global masking threshold, through the use of offline analysis data, involves several stages of execution. Firstly, when an auditory event is triggered, event manager (EM) does the preliminary operations required to handle this event (see 3.3.1 for details). EM then places or re-prioritizes this event in the decaying event priority queue in order to reduce hard drive access delay (3.3.2 for details). After the data was successfully stored in heap, timed circular buffer (TCB) adds the masking values to the global masking threshold. Then audibility percentage of the audio is calculated and if the threshold is bigger than the calculated percentage, audio is culled (3.3.3 for details).

A pseudocode of the proposed algorithm can be found in appendix A.

### 3.3.1 Concept of Auditory Events and Event Manager

For the proposed algorithm, game engine's audio component is encapsulated. For every spatial audio render request, we generate auditory events, later to be handled by the auditory event manager (EM). Each audio event includes:

- Distance from the listener; with 64 bits.
- Audio name; which denotes both the actual audio file path and the binary file which contains the masking information.

Event manager that was stated above has two responsibilities; handling auditory event requests and handling end-of-audio events coming from the engine. Event manager runs on its own thread with a single mutex that locks each time an event is received. Receiving end-of-audio events require the same mutex lock acquisition as receiving an auditory event. Hence they are synchronized, even if multiple threads call them simultaneously. One key difference between them, is that receiving end-of-audio event is a blocking operation whereas auditory event handling is not. Details of each operation are explained below.

**Auditory Event Handling:** Whenever an auditory event arrives, event manager tries to acquire the thread lock (via the only mutex it has). If it fails to acquire the lock, it simply returns without doing any operations and the audio will be rendered. If it acquires the lock, meaning if this is the only running real time analysis, it controls whether the masking information for the given audio exists or not. If there are no masking information present, audio will be automatically rendered, without doing any further operations.

When everything goes well, lock is acquired and the required data is present, event manager proceeds with the remaining operations. After all the operations and calculations are done and the decision is made, event manager releases the

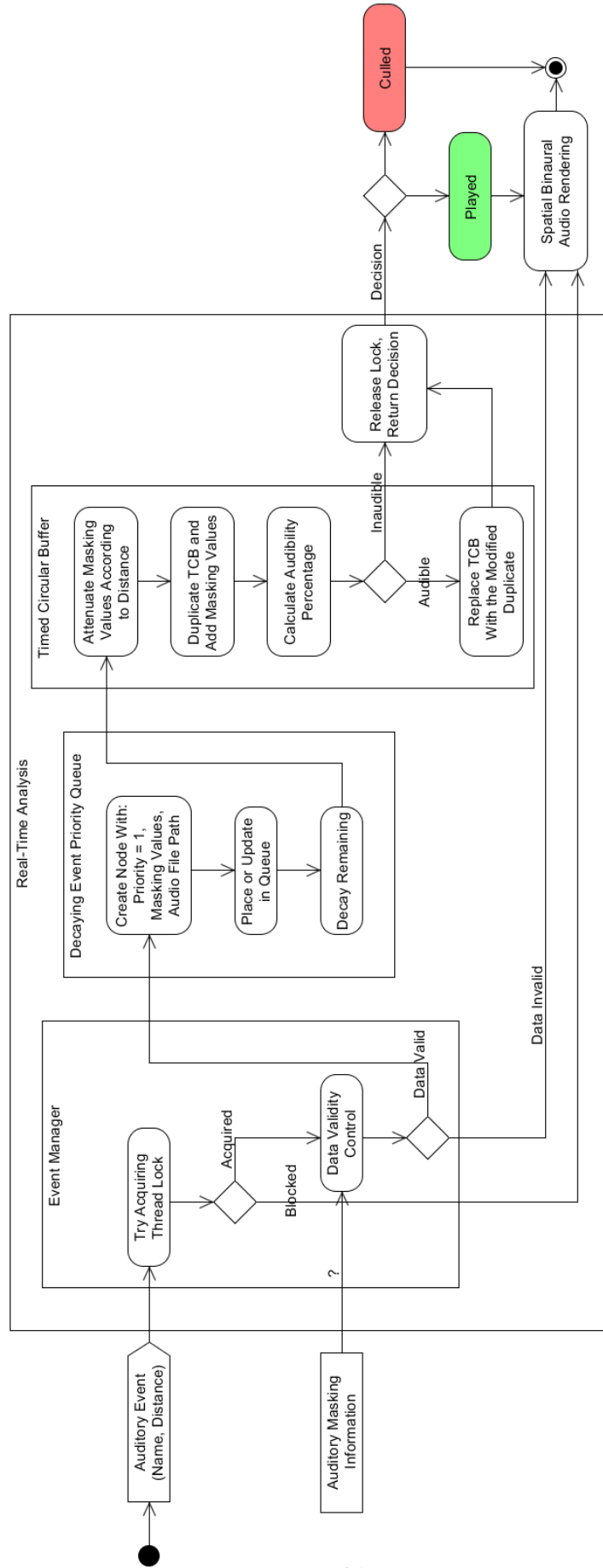


Figure 3.3: An activity diagram for handling auditory events in the real time analysis.

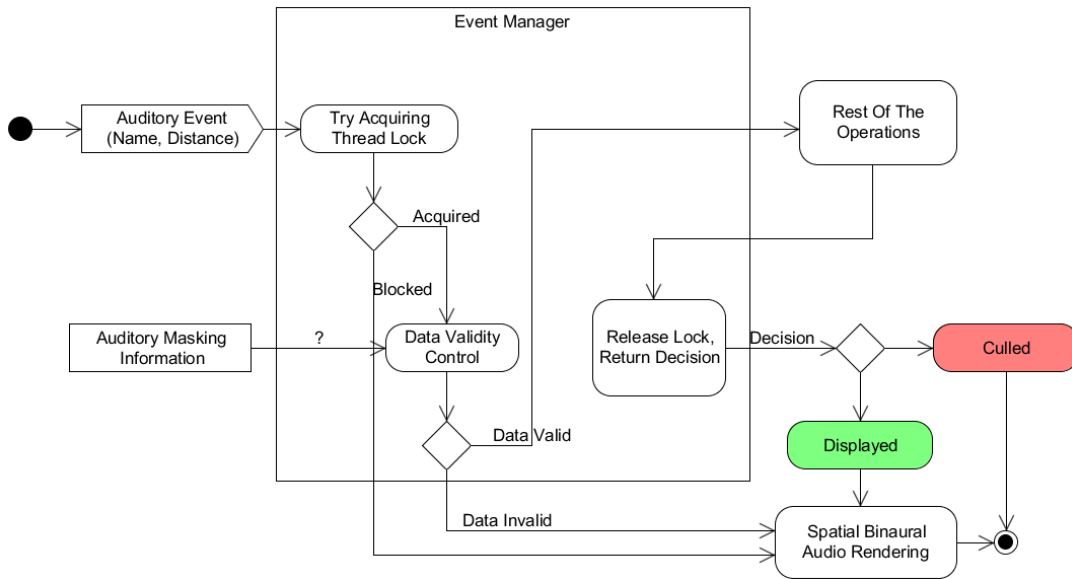


Figure 3.4: Event manager workflow for handling auditory events

thread lock and returns the result to the calling thread. After that if the audio is marked as 'inaudible', it will be culled, thus no further operations will be necessary. If the audio is marked as 'audible', engine proceeds with spatial binaural audio rendering. A visual representation of the workflow is displayed in figure 3.4.

**End-Of-Audio Event Handling:** These events contain no information. They are just signals that indicate that a single audio file has finished playing, and the information of which audio file is finished is not conveyed. These events are required in the execution of the circular masking threshold which will be explained in section 3.3.3.

When an *audio-is-finished* event comes, lock must be acquired. Caller thread will be blocked until this lock is attained. After the program counter is in the critical section, event manager decrements the active audio count. If there are no active sound in the scene, timed circular buffer is notified and the lock is released. A visual representation of the workflow is displayed in figure 3.5.

### 3.3.2 Decaying Event Priority Queue

As it was mentioned in the section 3.2, auditory masking information for each sound is stored in persistent data storage such as HDDs or SSDs. Compared to the data that was allocated dynamically from heap or stack, information retrieval is significantly slower from these resources. To counteract this effect, the retrieved auditory masking information is temporarily stored in a "decaying priority queue".

Decaying event priority queue (DEPQ) is a dynamically allocated priority queue



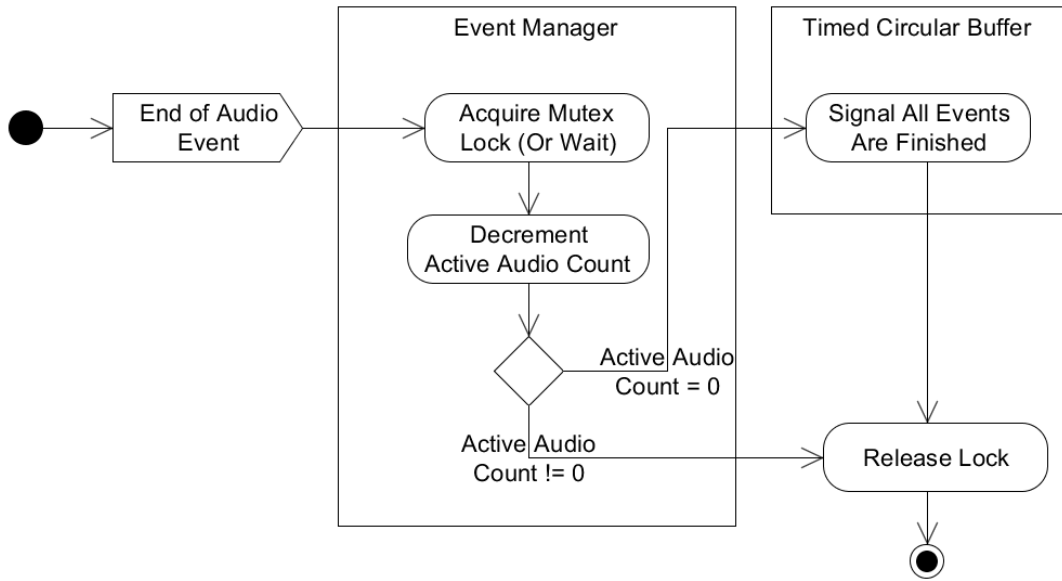


Figure 3.5: Event manager workflow for handling end-of-audio events

that stores a fixed amount of past auditory event information. Priorities for each event are set according to their arrival order. As the name suggests, priorities for each event are decremented every time a new event comes. When the queue is full and a unique event comes to be handled, the event with the lowest priority is replaced with the newer event. An example execution is given in figure 3.6, which is explained in the remaining paragraphs of this section.

When the program first starts, the DEPQ is initially empty. Then the auditory event that requests "A.wav" to be played back, comes. After failing the search through the DEPQ for "A.bin", data that contains the masking information is retrieved from the disk and the first lowest priority item in the list is replaced.

Same operation is repeated until handling of "C" is completed, while reducing the priorities of each event by one, each time an event comes. By the time the auditory event that requests "D.wav" to be displayed comes, the DEPQ is already full. Hence the event that has the lowest priority (the "oldest" event) is replaced with the content that "D.bin" contains.

When the auditory event "B" comes in the second time, the DEPQ already contains the required information about "B.bin" at index 2. At this point, there is no reason to retrieve data from the hard drive, hence we simply update the priority of "B.bin" to one and decrement the remaining data by one. Same process occurs with the second event of "D". On the last event "E", the event with the minimum priority "C", because of the decay, is replaced.

### 3.3.2.1 Hash Functions For Variable Length String Comparison

Both the string comparison and comparison of hash keys, are operations of time complexity  $O(n)$ . Though the hash keys will most likely be shorter than their string counterparts, hence would result in faster average searches through the DEPQ. Though this approach is still not applicable in our case because string that is to be hashed is of variable length. Problem comes from the perfect hash functions.

A perfect hash function maps all possible combination of elements into a set of integers, with no collisions [9]. Since the length of the path of audio file is variable, number of unique combinations are not fixed, hence a perfect hash function does not exist.

For non-perfect hash functions, it is guaranteed that two identical strings would give equal hash values. The reverse however, is not guaranteed for a non-perfect hash function. A given hash value will have the potential to be produced from several possible string values. Hence, even if we have calculate a hash key, the data could be coming from different strings (file paths). This would result in erroneous retrieval of the requested data so the hash functions cannot be used in conjunction with DEPQ.

However if the length of the audio file path is fixed, the number of possible unique combinations becomes fixed and a perfect hash function becomes available. We did not impose such a requirement hence we are not utilizing hash functions.

### 3.3.3 Timed Circular Buffer

Timed circular buffer (TCB) contains the global masking threshold and incorporates new masking values into it. All of the masking values come from binary files that was stored in the initial offline analysis stage. TCB also decides whether a given sound is audible or not by calculating audibility level as a percentage of the number of audible frames to the number of inaudible frames of the coming audio. TCB then returns the assessment to the event manager. Before we get into how global masking threshold is calculated, let us have a look at the proposed circular buffer implementation.

A circular buffer is a fixed size data structure where the end of the buffer is 'connected' to the beginning of the buffer. Meaning, when the end of the buffer is reached, data will be written over the initial indices of the array. Our timed circular buffer is a variation of this with 5 variables besides the size of the buffer and the buffer itself;

1. Start Index, inclusive; Points to the first valid data of the buffer. Updated according to time, stored as unsigned integer.
2. End Index, exclusive; It points to the first empty index of the buffer. Stored as unsigned integer.
3. Valid Index, exclusive; This points to the end of previous sound data and

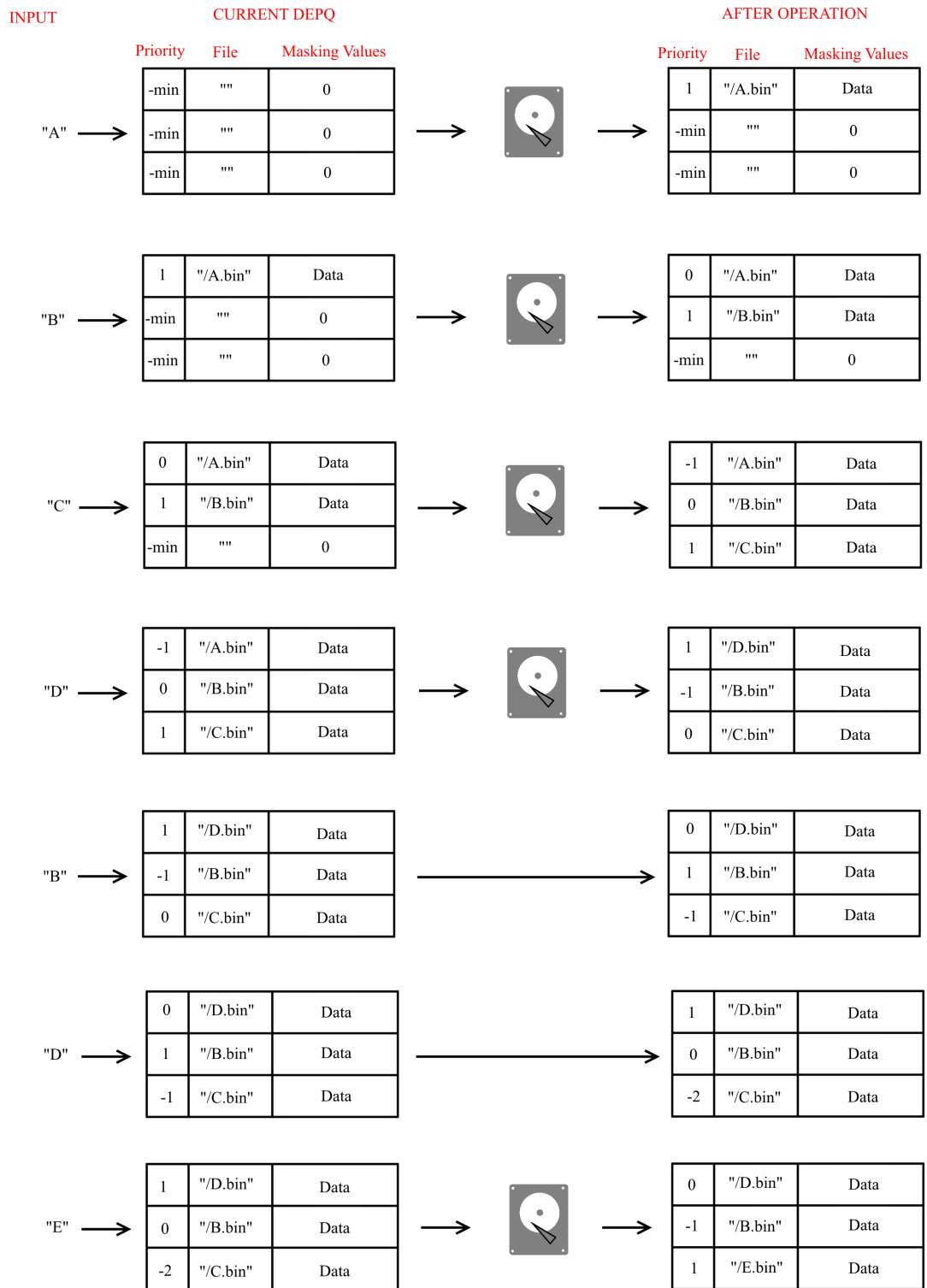


Figure 3.6: An example execution for the decaying event priority queue

index from which the array needs to be cleared from. One could also say it is this first index after the last valid data. Stored as unsigned integer.

4. Is Alive; Denotes whether the contents of the entire array is valid or not. It is a boolean variable.
5. Audibility Percentage; For a given audio and an existing scene, determines how much of the given auditory event is audible in the foreseeable future.

Details of how each index works in conjunction with the algorithm will be explained in the example execution found below. In the mean time, calculation of the frame time can be found below:

```
Frame Time = (FFT Size * 1000 ms) / Sampling Rate
            = (384 * 1000 ms) / 44100
            = 8.7074829931972789115646258503401 ms;
```

We take 512 point FFT but the actual sample sizes are shifted 384 samples every frame. Since MPEG layer 1 uses 12 samples per subband, there are 32 subbands (see 3.2.2.5 above for details). Hence the actual time that each frame denotes is about 8.7 milliseconds. Size of the buffer, in the case that the buffer will store 5 seconds of masking information, is calculated as follows:

```
bufferSize = Size of buffer for 5 seconds + 1 extra empty frame
            = floor(5000 / Frame Time) * 32 + (32)
            = floor(574.21875) * 32 + 32
            = 18400
```

Of course the size of buffer can be shorter or longer depending on user requirements.

As mentioned before, the start index is only modified by the actual time difference. It is updated each time an auditory event comes and is utilized for determining audibility of the incoming auditory event. Calculation of start index, when it was requested, is implemented as follows:

```
startIndex += (static_cast<unsigned int>
              (floor(timer.getElapsedTime() / frameDuration)) << 5);
startIndex = startIndex % circBfrSize;
```

Before masking values are added to the current global masking values, they are attenuated according to distance. Formula for the change of sound pressure level (*not* sound pressure) or sound intensity level according to distance is given below [4].

$$\Delta L_p = |20 \log(\frac{r_1}{r_2})| \text{ dB} = |10 \log(\frac{r_1}{r_2})^2| \text{ dB}$$

When determining the audibility of the audio for a given event, a threshold value called "audibility percentage" is utilized. This is defined as:

$$\text{Audibility percentage} = \frac{\text{number of audible frames}}{\text{total number of frames}} * 100$$

To determine whether a frame is audible or not, masking values are first compared with the current global masking values. If the given masking value is bigger for any of the 32 subbands, that frame of the audio will be deemed audible.

Below in the figure 3.7 you can find an example execution of the global masking threshold with its explanation listed below.

In this example, a frame of 32 doubles are depicted as a single block of data. So the buffer in the example can store  $5 * 32 = 160$  doubles which corresponds to 1280 bytes in our system. Initially the buffer is empty (stage 0) and let us assume that the masking values inside the buffers are already attenuated. So for example the audio indexed as 1 in the below example (the one that has three blocks of 10) has 10s in all of its each 32 subbands for all three of its frames. In the example below, the audibility percentage was set to %51. Each step of the example given below are explained with their corresponding index;

1. Our first auditory event which is indexed as 1, has a duration of 3 frames. Current time stamp is at index 0 (start index) and since the buffer is empty, valid index and end index are also 0. The attenuated masking values are first compared to the corresponding buffer values from 0 through 2 (shaded area in operation details). Since  $10 \geq 0$  for all three frames, audibility percentage is  $(3/3) * 100 = 100$  which is bigger than 51 percent. The buffer from indexes 0 through 2 needs to be cleaned and the masking values will be added to the circular buffer. The end index becomes  $0 + 3 = 3 \pmod{6}$  after the addition.
2. Only 1 frame time has passed since the initial auditory event and the audio indexed 1 is requested to be played again. Since 1 frame has passed since the previous event, start index is incremented by 1 which becomes 1 (triangle). For the frames 1 and 2,  $10 \geq 10$  still holds true and for the frame 3  $10 \geq 0$  holds true as well. So the audibility percentage for this event is  $(3/3) * 100 = 100$  as well. The audio will be played so the end index and valid index needs to be updated as well. Previous end index (2) becomes the valid index. Since the new audio will last 1 frame longer; end index becomes 4. Since the contents of the 3rd frame could be corrupt, so it will be cleaned (all elements between the new valid index until the end index). Then the masking values of audio 1 is added to the buffer.
3. Two more time frame have passed and an event containing audio number 2 comes. This audio has 4 frames of data, and the remaining buffer only has 3 elements. So the fourth element will be written on the index 0. Start index becomes  $1(\text{initial value}) + 2 = 3$ . Valid index becomes end index (4)

and the end index becomes  $4 + 3 = 1 \pmod{6}$ . All elements between 4 forward until 1, is cleansed. Frame number 3 is inaudible ( $10 \not\geq 20$ ) but all other 3 frames are audible ( $10 \geq 0$  for for 4,5 and 0). Hence audibility percentage is  $(3/4) * 100 = 75 > 51$ . Since the audio is decided to be played, masking values are added to the buffer.

4. The same procedure applies to here as well. For indexes 5 and 0,  $5 \not\geq 10$  they are not audible but for indexes 1 and 2,  $5 \geq 0$  so they are audible. However this event's audibility percentage is  $(2/4) * 100 = 50 \not\geq 51$  so this sound is not rendered.
5. On this event, all active audio that had been playing has already stopped. When the last "end of audio" event is received, buffer is reset and the new masking values are added over the empty buffer. Just like how it happened in the first event.
6. Since the audibility percentage is only %20, audio is not rendered in this case and the index 4 is assigned to zero. Unlike the picture depicted in the figure, end index reverts back to its original value once culling decision is approved.
7. This event comes at the last frame of the event number 2. Since  $75 \geq 51$  the audio is rendered. Indexes between 4 through 0 is assigned to zero.
8. Last event comes at the same time as the previous event. However since the previous event was deemed audible, this event is essentially masked by the previous event. If the ordering of the events were the reverse, both of the events would have to be played.

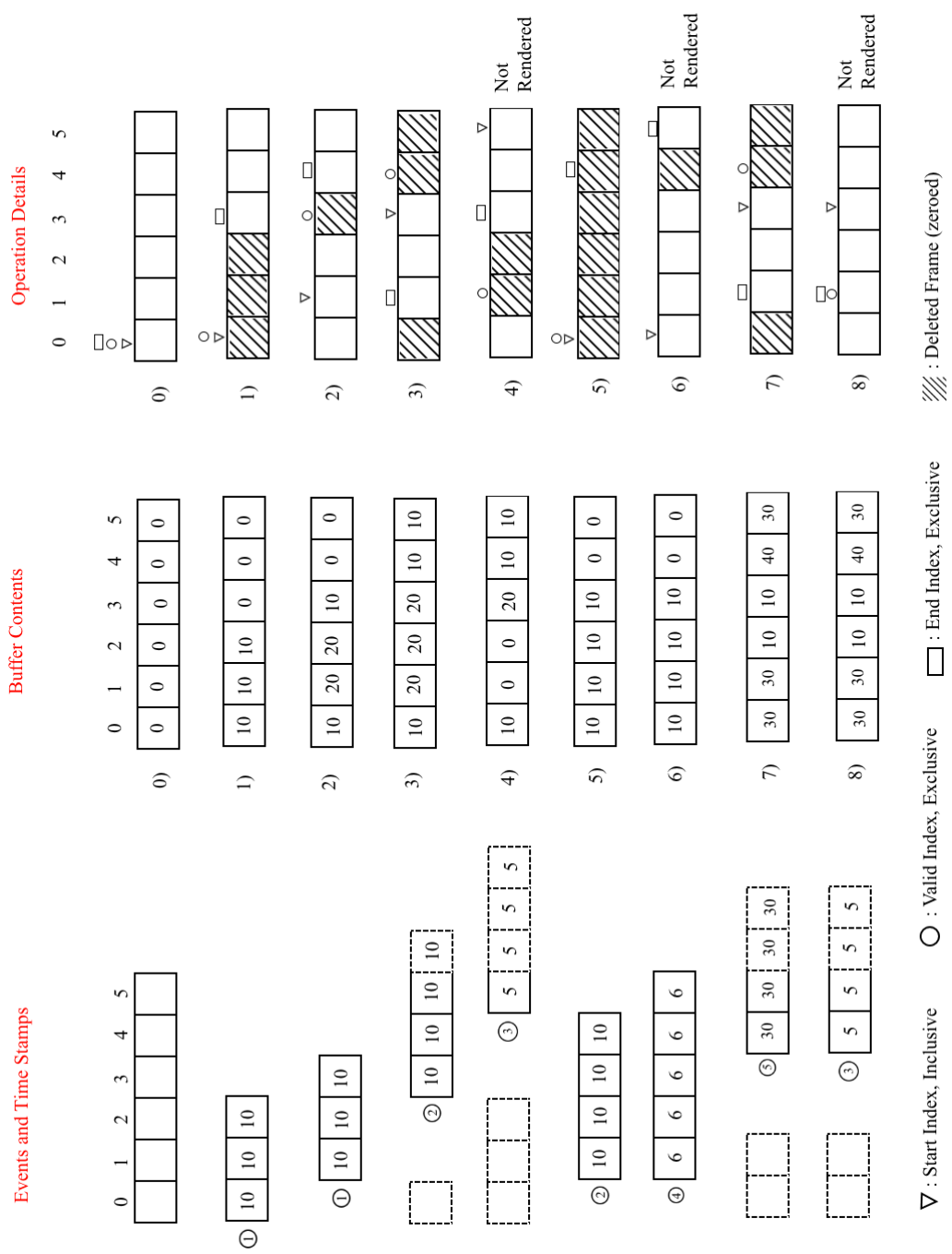


Figure 3.7: An example execution for the timer based circular buffer with an audibility threshold of %51





## CHAPTER 4

### PERFORMANCE AND ANALYSIS

The only purpose of this algorithm is to provide a performance advantage over an existing synthesis pipeline. Advantages and disadvantages of the proposed solution as well as the factors effecting the algorithm will be discussed in this chapter.

#### 4.1 Theoretical Information

##### 4.1.1 Algorithmic Complexity of Real-Time Analysis

Pseudocode of major parts of the algorithm can be found in the appendix A. Real time analysis is essentially composed of three components; event manager, decaying event priority queue and timed circular buffer. All operations of event manager has a complexity of  $O(1)$ , so we will investigate two other components here. In this section, size of the priority queue assumed to be  $m$  and size of the masking values are assumed to be  $n$ .

Decaying event priority queue (DEPQ) has two main functionalities; handling auditory events and handling end-of-audio events. End of audio events are of complexity  $O(1)$ . Handling auditory events on the other hand, not only requires interaction with the queue itself but also the audio masking values in the file system. Firstly, if size of the queue is  $m$ , searching through the queue takes  $O(m)$ . Secondly, fetching the masking values from the file system and placing them into the queue takes  $O(n)$  time.

For timed circular buffer, size of the input values are traversed twice. Once for attenuating the masking values according to distance and second for adding the masking values to the global masking threshold. So the complexity of this operation is  $O(n)$ .

##### 4.1.2 Factors That Affect The Cost of Real-Time Analysis

This section discusses the sole cost of real time analysis without rendering the actual scene.

For our system, sound can be as small as 1 frame of data as well as the full 5 seconds which amounts to  $\lfloor 44100 * 5/384 \rfloor = 574$  frames for a sound that has

44100 sampling rate. There are only three factors that determine the cost of real time analysis;

1. Length of the sound signal. Length of the sound affects many factors in regards to performance:
  - (a) In the case that the file isn't already stored in DEPQ, read time (excluding the seek time) is proportionate to the length of the offline analysis file.
  - (b) Each frame coming from the analysis file are compared to the global masking threshold in the TCB to determine whether a given audio is audible or not. Hence number of frames affects the duration of the analysis.
2. Type of persistent data storage. While retrieving the masking values from the hard drive, whether it is from an HDD or SSD matters in terms of performance.
3. Whether the audio is stored within DEPQ or not. If it isn't stored in program stack, time costs of seek time, data transfer rate and rotational latency (for HDDs) are added to the time cost of finalizing the analysis [34].

The resulting file from the offline analysis (file that contains the masking information) is approximately one thirds of the original audio file. Maximum file size permitted from our system is about 86 kilobytes (for a 5 second audio).

One major concern of the system is the type of persistent data storage used to retrieve offline analysis data. HDDs incur spin-up delays, seek time delays and slow data transfer rates compared to SSDs [10][39][40]. In a system that highly depends on time, such compromises render the system useless due to hardware delays. So if the end user were to use HDDs, increasing DEPQ size and storing the whole masking information into the queue is the only option.

## 4.2 System Performance

Here you can find various performance metrics for the proposed system. Time values listed here are hardware dependent. Below is a list of average times for each of the task, performed 10 times.

### 4.2.1 Individual Cost of Real Time Analysis

This section describes the isolated cost of the culling algorithm, without the cost associated with the rendering operations. Table 4.1 shows the average times to reach a decision, based on the conditions listed. Different hardware architectures would have different results.

Table4.1: Average Times Required to Complete Real Time Analysis

Input Size	Storage in DEPQ	Memory Type	Delta Time
1 Frame	Stored	RAM	0 ms
574 Frames	Stored	RAM	0.51 ms
1 Frame	Absent	HDD	15.626 ms
574 Frames	Absent	HDD	15.918 ms
1 Frame	Absent	SSD	0 ms
574 Frames	Absent	SSD	0.53 ms

#### 4.2.2 Performance Gain From a Single Audio

For a sound that was stored in DEPQ, average time it takes to render a 5 second audio is 0.51 milliseconds. During this time, a single CPU core is utilized at %100 percent while the overall utilization was %25. We can regard the total cost of a program execution as follows;

$$CostOfExecution = Average\ CPU\ Utilization * Time\ In\ Milliseconds$$

We can think of cost of execution as total number of instruction cycles. Since audio rendering incurs a performance penalty in the beginning of each frame, we can see whether this methodology is profitable or not. Since we're not considering the memory delay for retrieving the actual audio data (wav file) from memory, we are not taking consideration for the masking values either. For an audio that is already inside the DEPQ, and for an audio file that was already stored in RAM, the performance metrics are listed in the table as follows;

Table4.2: Costs Of Execution For an Integrated Audio Rendering Pipeline

Input Size	Culling Assessment	Cost Of Execution
1 Frame	No Culling Algorithm	$9ms * \%0.44 = 3.96$
1 Frame	With Culling Algorithm	$9ms * \%0.46 = 4.14$
574 Frames	No Culling Algorithm	$5000ms * \%0.45 = 2250$
574 Frames	With Culling Algorithm	$5000ms * \%0.49 = 2450$

As we can see from this table, culling algorithm incurs about %8 more performance over the existing audio synthesis pipeline. However in the case that the audio will be culled, we will save %92 percent of the clock cycles. One disadvantage of the culling algorithm, is that it will block a CPU core completely until its analysis is done.

Average CPU loads listed above, were calculated from "GetSystemTimes" function of "windows.h" header file.

### 4.2.3 Worst Case Scenario

Worst case scenario for our algorithm would be the arrival of unique, concurrent and large number of auditory events. This case would also be the worst case for any audio rendering pipeline with or without any culling methodology applied. But in the case of our algorithm, this would cause repeated replacement of items in the DEPQ and repeated access to memory. Even if the algorithm might save performance in the long run, this would cause a performance spike in the initial arrival.

A simple precaution of limiting maximum number of auditory events would alleviate this situation. With that many sound sources, listener would be highly unlikely to hear any difference anyway.

## CHAPTER 5

### PSYCHOACOUSTICAL EVALUATION

#### 5.1 Preliminaries

Audio produced with sound source culling, regardless of the methodology, is dependent on the user determined variables. Resulting auditory richness produced by the aforementioned methodologies, provide a crude approximation to the original scene. In order to find out which parameters provide sufficient perceived richness, a psychoacoustical evaluation was performed.

#### 5.2 Chosen Test Methodology

At this time of writing, a subjective testing standard for assessing auditory richness do not exist. However a method for subjective assessment of intermediate quality level of audio systems, does. The ITU-R recommendation BS.1534 proposes a method called “MUlti Stimulus test with Hidden Reference and Anchor (MUSHRA)” [17] which is appropriate for assessing intermediate audio quality. It is our premise that MUSHRA is also applicable for evaluating intermediate auditory richness.

The resulting systems that undergo the tests are expected to introduce significant auditory impairments. Our assumption here is that culling large number of sound sources introduce this effect. Production of mediocre sound richness, through selection of different algorithmic variables, make this testing methodology appropriate for our purposes.

#### 5.3 Test Procedure

Details of the chosen methodology is explained below.

##### 5.3.1 Presentation of Stimuli

In the MUSHRA test method, a high quality reference signal, a low quality anchor signal and other signals that fall in between them in terms of quality are evaluated [17]. At any given test, there is a single reference and a single anchor that the user is expected to find. Both experiments themselves and the stimulus contained within the experiments are randomly presented. So not only

each listener listens experiments in a different order but each listener faces a randomized presentation order of the stimuli.

During each experiment, all of the signals are presented on the same GUI. Each prerecorded sound can be played as many times as the listener desires. This allows listeners to compare sounds and alter their scores for any given stimulus.

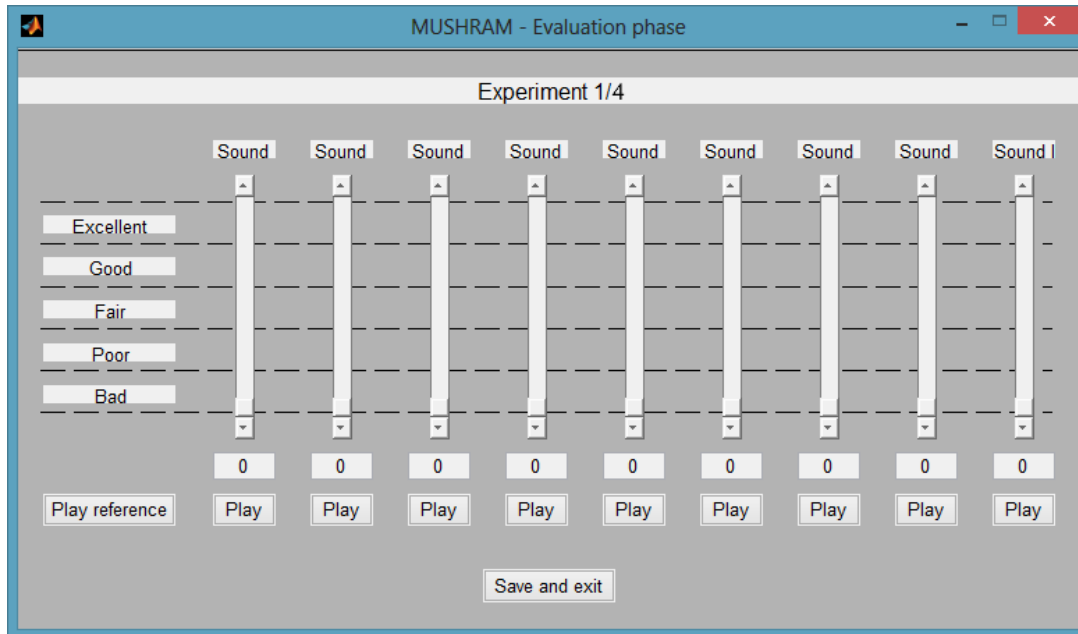


Figure 5.1: Mushram User Interface for the First Experiment

For the experiments, a Matlab interface called MUSHRAM was utilized [38]. Figure 5.1 shows a screenshot of an example used for assessment in this thesis. As mentioned above, the subjects that took the test were given different order of experiments. Ordering of the experiments is given in appendix B.

### 5.3.2 Grading

The whole test procedure is comprised of giving ratings to test signals which are displayed in a random sequence. Listeners were instructed to score the presented samples in comparison with the explicitly provided reference signal. The scores that are given, can range between 0 and 100. At any given test, there is a single reference and a single anchor. Listeners were asked to find and rate them 100 and 0, respectively. For any of the remaining stimuli, listeners were asked to rate them according to their richness. They were also encouraged to listen to the available reference signal, prior to giving scores on the stimuli.

## 5.4 Experiment Details

Scope of the applied experiments, positioning of the sound sources and other required information about the auditory scene are stated under this section.

The MUSHRA test applied for this paper is comprised of four different experiments. Each experiment involves a combination of different contextual sound sources. Each sound source is categorized under four contextual categories; impulsive sounds, music, static sound effects and speech signals. From these sound sources, we arranged four different psychoacoustical experiments, involving sounds under different categories. In each of these scenes, there was a total of 13 sound sources in the case of reference signals. The applied culling methodology served to reduce the number of sound sources at each rendered scene. Composition of each scene is given in the table below.

Table5.1: Contents of Test Scenes

Experiment	Contents (ss = sound sources)
Impulsive Scene	13 impulsive ss.
Impulsive + Speech + Music	6 impulsive ss., 6 speech ss., 1 Music sound
Impulsive + Sound Effects + Music	6 sound effect ss., 6 impulsive ss., 1 Music sound
Sound Effects + Speech	7 speech ss., 6 sound effect

Each test case (stimulus) produced from these scenes needed to be short enough, so that the listener would be able to recall the whole sample. This was important for the experiment because the whole duration of the audio were to be assessed. Our test scenes were adjusted so that the maximum length of a stimulus does not exceed 2 seconds. <sup>1</sup>

For each of the experiments above, stimuli with different parameters were produced to be tested in MUSHRA. Both perceptual culling (our method) and distance based culling (spherical culling) methodologies were tested during the same experiment.

#### 5.4.1 Selection of Sound Source Locations

Since both systems needs to be tested on the same conditions, a scene that is appropriate to both culling methodologies was required. Sound sources also needed to be distributed along the horizontal plane in order to achieve maximum perceptibility.

In order to achieve this, sound sources were distributed along the horizontal plane of the listener. Assuming one source will be directly in front of the listener, the axis needed to be divided into  $180/(13-1) = 15$  degrees. Listener was placed at the coordinate location (700, 700), facing towards the negative X axis.

In order to achieve a controllable spherical culling methodology, each of the sound sources were placed in a fixed radius away from the listener. They were

---

<sup>1</sup> According to Guillermo Campoy, auditory short-term memory decay happens at around 3 seconds. [7]

grouped into pairs and were placed at an equal distance and symmetrical angles from the listener. Though their auditory events were not actuated on the same time frame. Coordinates of sound sources were calculated according to the formula:

$$X = r \cos(\theta) + 700$$

$$Y = r \sin(\theta) + 700$$

You can find the related information on the Table 5.2 and a top down view that demonstrates distances sound sources of the scene in Figure 5.2.

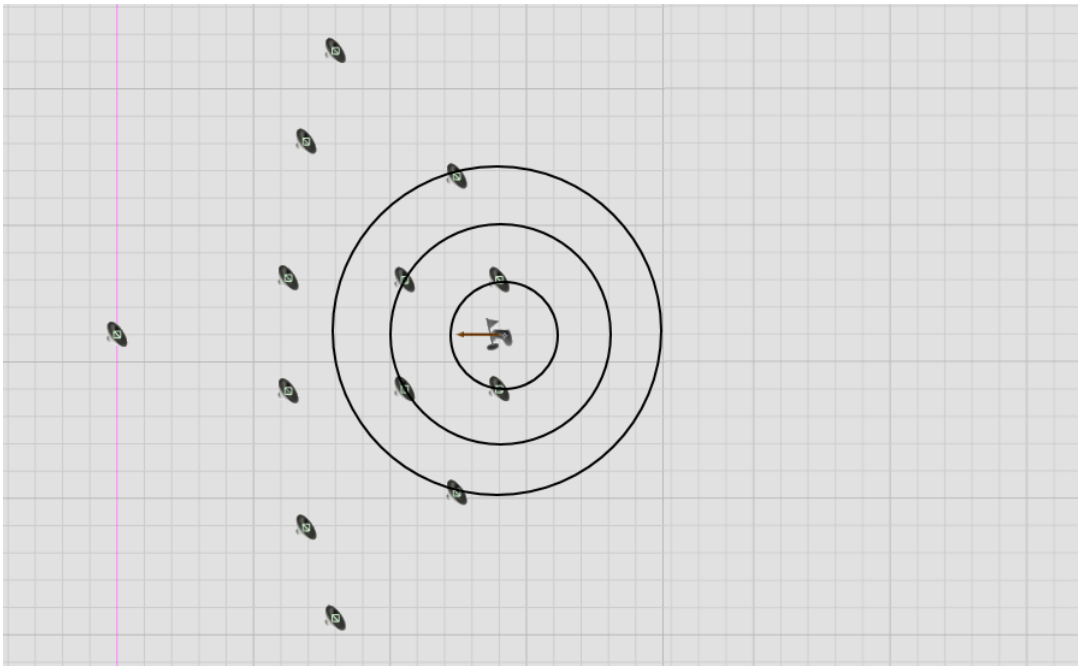


Figure 5.2: Source Positions For Tested Scenes

This way culling distance values can be adjusted with ease. For example a culling distance of 200 will not render a sound source positioned 300 units away from the listener.

#### 5.4.2 Generation of Test Cases

There are nine sound files in each experiment. The reference signal includes all 13 sounds. The anchor signal involves only a single sound source to achieve the lowest richness in a scene. Remaining signals are results of scenes with culling methodologies applied. For the perceptual culling methodology, various audibility percentages were tested. For spherical culling methodology, various culling distances were tested. In total, there were 4 scenes with 9 sound files. Making a total of 36 test cases.



Table5.2: Test Setup, Positional Information

Point	X	Y	Degrees	Distance From Listener
Listener	700	700	-	-
1	700	600	270	100
2	622,354	410,222	255	300
3	400	180,385	240	600
4	346,447	346,447	225	500
5	526,795	600	210	200
6	313,630	596,472	195	400
7	0	700	180	700
8	313,630	596,472	165	400
9	526,795	600	150	200
10	346,447	346,447	135	500
11	400	180,385	120	600
12	622,354	410,222	105	300
13	700	600	90	100

Among the 7 test cases, 4 of them were produced with perceptual culling and 3 of them were produced with spherical culling. For perceptual culling, %0, %10, %20 and %30 audibility percentages were used. For spherical culling, 650, 550 and 450 cm culling distances were used.

## 5.5 Statistical Analysis

There were 11 participants that performed the test. Figure 5.3 shows the results of subjective scores from the test. Green and khaki colors display subjective results of audibility and volumetric culling respectively. As it can be seen from the figure 5.3, scenes that have 8, 10, 12 and 13 sources have results for both of the culling methodologies. However, for scenes that have 9 and 11 sound sources, results for volumetric culling are not available. This is due to the configuration of the scene (see 5.4.1 for details).

In order to see whether our methodology has any significance over volumetric culling methodology for a given scene, we form the following null hypothesis:

$H_0$ : For a stimulus with same amount of sound sources, audibility based culling methodology will have no significant effect on the subjective test scores.

In order to make a fair comparison of the two culling methodologies, we need to assess results that have the same number of sound sources. Hence we form the following null hypotheses:

$H_0$ : For a stimulus with 8 sound sources, audibility based culling methodology

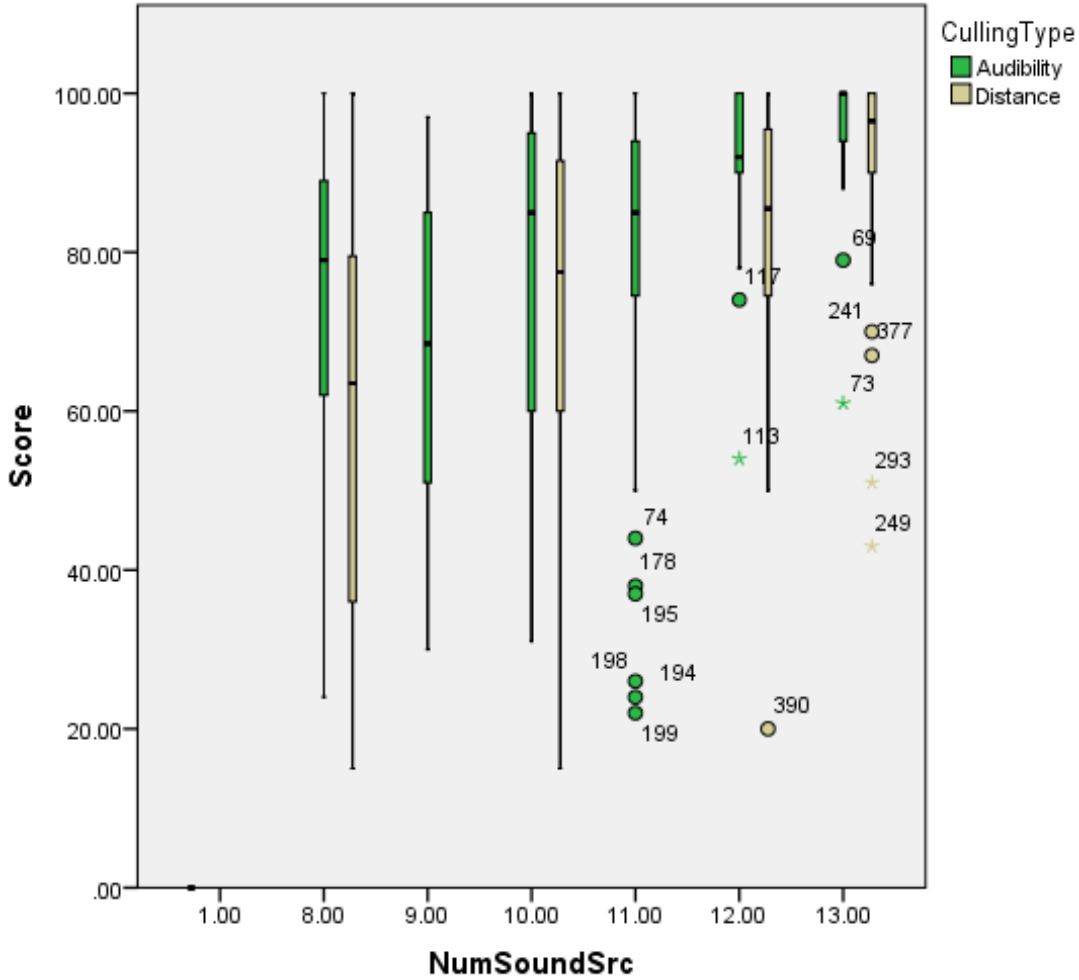


Figure 5.3: Subjective Scores From Listening Tests

will have no significant effect on the subjective test scores.

$H_0$ : For a stimulus with 10 sound sources, audibility based culling methodology will have no significant effect on the subjective test scores.

$H_0$ : For a stimulus with 12 sound sources, audibility based culling methodology will have no significant effect on the subjective test scores.

$H_0$ : For a stimulus with 13 sound sources, audibility based culling methodology will have no significant effect on the subjective test scores.

In order to test these hypotheses, we perform independent-samples t-test. Table 5.4 shows the results of t-tests for stimuli with 8, 10, 12, 13 sources in rows labeled as sc8, sc10, sc12, sc13 respectively.

For stimuli produced with 8 sound sources, we first look at results of Levene's test. We see that it is 0.031 which is smaller than 0.05. This shows that the variability within each of culling methodology are significantly different compared to each other. Thus, we can obtain the p value from the bottom row. It

is 0.003 which is smaller than 0.05. Hence we can conclude that there is a statistically significant difference between the means of audibility and volumetric culling methodologies for 8 sound sources. We can reject the null hypothesis.

For stimuli produced with 10 sound sources, we first look at results of Levene’s test. We see that it is 0.045 which is smaller than 0.05. This shows that the variability within each of culling methodology are significantly different compared to each other. Thus, we can obtain the p value from the bottom row. It is 0.005 which is smaller than 0.05. Hence we can conclude that there is a statistically significant difference between the means of audibility and volumetric based culling methodologies for 10 sound sources. We can reject the null hypothesis.

For stimuli produced with 12 sound sources, we first look at results of Levene’s test. We see that it is 0.056 which is bigger than 0.05. This shows that the variability within each of culling methodology are not scientifically different compared to each other, which is desired. We will obtain the p value from the top row. It is 0.052 which is bigger than 0.05. So there is no statistically significant difference between two culling methodologies for 12 sound sources. We fail to reject the null hypothesis.

For stimuli produced with 13 sound sources, we first look at results of Levene’s test. We see that it is 0.825 which is bigger than 0.05. This shows that the variability within each of culling methodology are not scientifically different compared to each other, which is desired. We will obtain the p value from the top row. It is 0.707 which is bigger than 0.05. So there is no statistically significant difference between two culling methodologies for 13 sound sources. We fail to reject the null hypothesis.

In light of these results, we can form two alternative hypotheses:

$H_A$ :For a stimulus with 8 sound sources, audibility based culling methodology has a significant effect on the subjective test scores.

$H_A$ :For a stimulus with 10 sound sources, audibility based culling methodology has a significant effect on the subjective test scores.

Table5.3: Mean Values

CullingType	Audibility			Distance		
	Mean	N	Std. Deviation	Mean	N	Std. Deviation
sc8	73.6364	33	20.36974	57.6591	44	25.42197
sc10	78.3939	33	20.80406	73.2500	44	23.07231
sc12	90.0000	22	11.19098	82.0227	44	17.11994
sc13	93.4545	11	12.78565	91.8182	44	12.86111

From the results of t tests, we have seen that culling methodology for 8 and 10 sound sources does have a significant effect on the result. Whereas for 12 and 13 sources, it didn’t have any significant effect on the results. Looking at the means of stimuli with 8 and 10 sources ( see table 5.3 ), it can be seen that

audibility based culling has a significant advantage. For the remaining cases, it also yielded with a higher mean without statistical significance.

Table5.4: Results of Independent-Samples t-test

		Independent Samples Test				
		Levene's Test for Equality of Variances		t-test for Equality of Means		
		F	Sig.	t	df	Sig. (2-tailed)
sc8	Equal variances assumed	4.850	.031	2.965	75	.004
	Equal variances not assumed			3.060	74.636	.003
sc10	Equal variances assumed	4.072	.045	2.767	152	.006
	Equal variances not assumed			2.850	151.046	.005
sc12	Equal variances assumed	3.789	.056	1.980	64	.052
	Equal variances not assumed			2.270	59.268	.027
sc13	Equal variances assumed	.049	.825	.378	53	.707
	Equal variances not assumed			.379	15.469	.710



## CHAPTER 6

### CONCLUSION

#### 6.1 Contributions and Discussion

Within the scope of this thesis, a sound source culling methodology was developed. An advantage over its predecessors, is its consideration of psychoacoustics when culling the sound at hand. While this approach is more expensive than the traditional distance based culling, it has been demonstrated to provide better auditory richness. For the same number of sound sources, our subjective tests dictate superior subjective results.

While algorithm itself will provide better utilization of CPU in the long run, it has its drawbacks. First of all, every auditory event will create a short performance spike. Longer audio files will stall a processor longer than a short audio file. While these are insignificant in small numbers, large number of auditory events triggered in the same audio frame, will cause an issue. Secondly, if the size of the DEPQ is or cannot be sufficiently large to encompass all audio files, persistent data storage will be of importance. Despite recent progress of solid state disks, hard disk drives are still prevalent today due to their price and already existing hardware. HDD metrics like spin-up delays, seek time delays and slow data transfer rates render this algorithm disadvantageous. This is because a 15 millisecond delay amounts to about 2 audio frames of a delay for 44.1kHz sampling rate. While this can be acceptable for distant sound sources (due to the speed of sound), it is not acceptable for close proximity sound sources.

Where this algorithm shines, compared to the distance based approach, is on the high end systems with either large amounts of RAM or hardware equipped with SSDs. In the case when an audio is culled, it will save about 92% of the machine cycles that would have occurred if audio was not culled. Adjusting the permeable audibility percentage will even cause more sounds to be culled, at the cost of perceived richness. However at 0% percent audibility threshold, perceived richness will likely to be no different than a scene without culling. Another area where this algorithm performs well is arrival of sparse, yet large number of auditory events. In that case, perceived richness of the audio will be superior than a distance based approach.

An additional contribution made with this thesis, is the result of subjective tests. Even though the tests were performed with our methodology, they have made a case for perceptual sound source culling. As demonstrated in Chapter 5, per-

ceptual sound source culling produces more realistic soundscapes. Whether this information is useful for other perceptual approaches, such as culling methodologies that incorporate view frustrum, is up to the reader to decide.

In a nutshell, we provided a unique, performance enhancing spatial sound source culling algorithm. It has proven to be advantageous for providing high fidelity audio for high end hardware. For sparse and numerous auditory events, it provides performance advantage without compromise. However the algorithm is proven to be disadvantageous for systems with low end hardware and dense auditory events. In those cases, either an additional change to the algorithm or a distance based approach can be employed.

## 6.2 Future Work

As we have discussed above, there are weaknesses that need to be addressed. An alteration of the algorithm that will handle dense auditory events would be vital as an industrial product. Since distance based culling has an undeniable performance advantage, an algorithm that would incorporate both methodologies would perform the best, while providing the best possible auditory richness.

Our algorithm involved mutex locks for critical sections. While this enabled multithreading, incorporation of lock free programming, namely incorporation of atomic variables, would increase its performance further.

In the real time analysis, we are merely comparing global masking thresholds for the scene to the masking thresholds of the incoming audio. Yet an audio can have a lower masking threshold than the scene and still be audible. This is because signal's power can still exceed global masking threshold without having its own threshold higher. So at the cost of performing additional calculations or storage, a more precise perceptual model can be developed.

Within the scope of our algorithm, we have not incorporated moving sound sources. An algorithm that would incorporate such cases can be devised.



## Bibliography

- [1] V Ralph Algazi, Richard O Duda, Dennis M Thompson, and Carlos Avendano. The cipc hrtf database. In *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*, pages 99–102. IEEE, 2001.
- [2] Kaoru Ashihara, Kenji Kurakata, Tazu Mizunami, and Kazuma Matsushita. Hearing threshold for pure tones above 20 khz. *Acoustical science and technology*, 27(1):12–19, 2006.
- [3] Durand R Begault and Leonard J Trejo. 3-d sound for virtual reality and multimedia. 2000.
- [4] UdK Berlin. Decrease in level of sound pressure and sound intensity with distance.
- [5] Jens Blauert. *Spatial hearing: the psychophysics of human sound localization*. MIT press, 1997.
- [6] Alan Dower Blumlein. Improvements in and relating to sound-transmission. *Sound-recording and Sound-reproducing Systems, UK Patent*, 394:325, 1931.
- [7] Guillermo Campoy. Evidence for decay in verbal short-term memory: A commentary on berman, jonides, and lewis (2009). 2012.
- [8] Crytek.
- [9] Zbigniew J Czech, George Havas, and Bohdan S Majewski. An optimal algorithm for generating minimal perfect hash functions. *Information Processing Letters*, 43(5):257–264, 1992.
- [10] Cagdas Dirik and Bruce Jacob. The performance of pc solid-state disks (ssds) as a function of bandwidth, concurrency, device architecture, and system organization. In *ACM SIGARCH Computer Architecture News*, volume 37, pages 279–289. ACM, 2009.
- [11] Epic.
- [12] FJ Fry and JE Barger. Acoustical properties of the human skull. *The Journal of the Acoustical Society of America*, 63(5):1576–1590, 1978.
- [13] William G Gardner and Keith D Martin. Hrtf measurements of a kemar. *The Journal of the Acoustical Society of America*, 97(6):3907–3908, 1995.
- [14] David Grelaud, Nicolas Bonneel, Michael Wimmer, Manuel Asselot, and George Drettakis. Efficient and practical audio-visual rendering for games using crossmodal perception. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 177–182. ACM, 2009.

- [15] David Griesinger. Stereo and surround panning in practice. In *Audio Engineering Society Convention 112*. Audio Engineering Society, 2002.
- [16] ISO / IEC. *Coding Of Moving Pictures And Associated Audio For Digital Storage Media At Up To About 1.5 Mbit/S Part 3 Audio*, June 1993.
- [17] ITU-R. *Method for the Subjective Assessment of Intermediate Quality Level of Audio Systems*, bs.1534-3 edition, 10 2015.
- [18] Christian Lauterbach, Anish Chandak, and Dinesh Manocha. Interactive sound rendering in complex and dynamic scenes using frustum tracing. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1672–1679, 2007.
- [19] Ruth Litovsky. Development of the auditory system. *Handbook of clinical neurology*, 129:55, 2015.
- [20] Brian CJ Moore. Psychophysical tuning curves measured in simultaneous and forward masking. *The Journal of the Acoustical Society of America*, 63(2):524–532, 1978.
- [21] Martin Naef, Oliver Staadt, and Markus Gross. Spatialized audio rendering for immersive virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 65–72. ACM, 2002.
- [22] Harry Ferdinand Olson. *Music, physics and engineering*, volume 1769. Courier Corporation, 1967.
- [23] Fabien A. P. Petitcolas. MPEG Psychoacoustic Model I for MATLAB. [www.cl.cam.ac.uk/fapp2/software/mpeg/](http://www.cl.cam.ac.uk/fapp2/software/mpeg/), 2003.
- [24] Christopher J Plack. *The sense of hearing*. Psychology Press, 2013.
- [25] John G Proakis and Dimitris G Manolakis. *Introduction to digital signal processing*. Prentice Hall Professional Technical Reference, 1988.
- [26] Ville Pulkki. Virtual sound source positioning using vector base amplitude panning. *Journal of the Audio Engineering Society*, 45(6):456–466, 1997.
- [27] John William Strutt Baron Rayleigh. *The theory of sound*, volume 2. Macmillan, 1896.
- [28] Stuart Rosen and Peter Howell. *Signals and systems for speech and hearing*, volume 29. Brill, 2011.
- [29] Thomas D Rossing. *Handbook of acoustic*, 2007.
- [30] Douglas Self. *Audio engineering explained*. Taylor & Francis, 2009.
- [31] Roland Sottek and Klaus Genuit. Physical modeling of individual head-related transfer functions. *J. Acoust. Soc. Am*, 105(2):1162, 1999.
- [32] Pramila Srinivasan and Leah H Jamieson. High-quality audio compression using an adaptive wavelet packet decomposition and psychoacoustic modeling. *Signal Processing, IEEE Transactions on*, 46(4):1085–1093, 1998.

- [33] Gerhard Stoll and Karlheinz Brandenburg. The iso/mpeg-audio codec: A generic standard for coding of high quality digital audio. In *Audio Engineering Society Convention 92*, Mar 1992.
- [34] Hamid D Taghirad and Ehsan Jamei. Robust performance verification of adaptive robust controller for hard disk drives. *Industrial Electronics, IEEE Transactions on*, 55(1):448–456, 2008.
- [35] Unity Technologies.
- [36] Hartmut Traunmüller. Analytical expressions for the tonotopic sensory scale. *The Journal of the Acoustical Society of America*, 88(1):97–100, 1990.
- [37] Nicolas Tsingos, Emmanuel Gallo, and George Drettakis. Perceptual audio rendering of complex virtual environments. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 249–258. ACM, 2004.
- [38] E Vincent. Mushram: A matlab interface for mushra listening tests. *Online/ <http://www.elec.qmul.ac.uk/people/emmanuelv/mushram>*, 2005.
- [39] OCZ Official Website.
- [40] Seagate Official Website.
- [41] Xiao-li Zhong and Bo-sun Xie. Overall influence of clothing and pinnae on shoulder reflection and hrtf. *SHENGXUE JISHU*, 25(2):113, 2006.
- [42] Eberhard Zwicker and Hugo Fastl. *Psychoacoustics: Facts and models*, volume 22. Springer Science & Business Media, 2007.



## APPENDIX A

### REAL TIME ANALYSIS PSEUDOCODE

This section describes some of the algorithms used in this methodology. Real time analysis starts with the event handling from the event manager. Rest of the process is handled by the Decaying Event Priority Queue and Timed Circular Buffer, as portrayed in the figure 3.3. After a sound event is over, end-of-audio event will be handled by the event manager as well. Only the necessary algorithms are described below.

```

input : Sound file, distance, cullingIsOn
output: Rendering decision

if cullingIsOn then
  initialize;
  try acquiring mutex;
  if mutex is locked then
    call inquire for masking values inside DEPQ;
    if masking values are inside DEPQ then
      retrieve values from the queue and reprioritize;
      hold values in volatile memory;
    else
      read and parse data from persistent memory;
      hold values in volatile memory;
      place values in DEPQ and reprioritize;
    end
    call Pass masking values to TCB;
    if Audio is deemed audible then
      increment active audio count;
      release mutex;
      //Audio will be played
      return true;
    else
      //Inaudible audio according to threshold
      release mutex;
      return false;
    end
  end
  else
    //Another thread is executing analysis, just play the sound file
    return true;
  end
else
  //Culling is off
  return true;
end

```

**Algorithm 1:** Event Manager, Auditory Event Handling

```

input : Array of values, audio file path
output: none

//Assume first item in queue has the lowest priority
minPriority ← priority of the first item in queue;
minIndex ← 0;

for  $i \leftarrow 0$  to array size do
    if priority of  $i$ 'th element > minPriority then
        minPriority ← priority of  $i$ 'th element;
        minIndex ←  $i$ ;
    end
end

//Now we can write over the event with the lowest priority
queue[minIndex] ← input values, data and file path

//Priority queue will be reprioritized
priority of queue[minIndex] ← 1.
tempIndex ← 0.

while tempIndex < minIndex do
    decrement queue[minIndex]'s priority;
    increment tempIndex;
end
increment tempIndex;
while tempIndex < arraySize do
    decrement queue[minIndex]'s priority;
    increment tempIndex;
end

```

**Algorithm 2:** Event Priority Queue, Enqueuing

```

input : inputMaskingValues (array of data), distance
output: rendering decision

tempAttenuatedInput ← input masking values;
currentFrameInaudible ← true;
audibleFrameCount ← 0;
renderDecision ← false;
call update starting frame;

//Calculate the finishing index of the input.
//(this number can be numerically more than array size)
tempResult ← inputSize + startIndex;

// With the addition of new data, end of all events can be longer
if tempResult > endIndex then
|   oldEndIndex ← endIndex;
|   endIndex ← tempResult (mod circularBufferSize);
end

//If we are going to adding masking values,
//Invalid data needs to be removed
call clear depreciated data;

for unsigned index ← 0 to data size do
|   tempAttenuatedInput[index] ←
|   inputMaskingValues[index] - |20 * log10  $\frac{1}{Distance}$ |
|   if 0 == index (mod circularBufferSize) then
|   |   currentFrameInaudible ← true;
|   |   minIndex ← i;
|   end
|   // If the new masking value is bigger than the contents of TCB
|   if gmtData[startIndex + index] (mod circularBufferSize)
|   ≤ tempAttenuatedInput[index] and currentFrameInaudible then
|   |   currentFrameInaudible ← false;
|   |   increment audibleFrameCount;
|   end
end

```

**Algorithm 3:** Timed Circular Buffer, Handling Masking Values, Part 1



```

// Finding the percentage of audible frames
// Dividing input size by 32 (shifting right), to see the frame count
tempPercentage ← (sizeofinputMaskingValues) >> 5;
tempPercentage ← audibleFrameCount/tempPercentage;
if tempPercentage ≥ acceptableAudibilityPercentage then
| renderDecision ← true;
end
// Audio decided to be rendered. Add to the masking threshold.
if renderDecision == true then
| for index ← 0 to input size do
| | gmtData[startIndex + index]
| | (mod circularBufferSize)+ = tempAttenuatedInput[index]
| end
else
| // Audio wasn't deemed audible. Reverting to initial values.
| endIndex ← oldEndIndex;
end

```

**Algorithm 3:** Timed Circular Buffer, Handling Masking Values, Part 2



## APPENDIX B

### EXPERIMENT ORDERS

Here you can find the order of sequence for the performed experiments.

Experimenter	Exp. 1	Exp. 2	Exp. 3	Exp. 4
1	1	2	3	4
2	4	3	1	2
3	1	4	3	2
4	4	1	3	2
5	3	4	1	2
6	2	1	3	4
7	2	3	1	4
8	4	3	2	1
9	2	3	4	1
10	3	2	1	4

TableB.1: Psychoacoustical Test Order

Experiment numbers listed above denote the following scenes. For further information, refer to section 5.3.1 of this paper.

1. Impulsive
2. Impulsive + Music + Sound Effects
3. Impulsive + Speech + Music
4. Sound Effects + Speech



## APPENDIX C

### APPROVAL OF ETHICS COMMITTEE

UYGULAMALI ETİK ARAŞTIRMA MERKEZİ  
APPLIED ETHICS RESEARCH CENTER

ORTA DOĞU TEKNİK ÜNİVERSİTESİ  
MIDDLE EAST TECHNICAL UNIVERSITY

DÜMLÜPİNAR BULVARI 06800  
ÇANKAYA ANKARA/TURKEY Sayı: 28620816/ 316 - 37  
T: +90 312 210 22 91  
F: +90 312 210 79 59  
ueam@metu.edu.tr  
www.ueam.metu.edu.tr

17 Ağustos 2015

Gönderilen : Yrd.Doç.Dr. Hüseyin Hacıhabiboğlu  
Modelleme ve Simülasyon Bölümü

Gönderen : Prof. Dr. Canan Sümer  
IAK Başkan Vekili

İlgi : Etik Onayı

Danışmanlığını yapmış olduğunuz Oyun Teknolojileri Bölümü öğrencisi Ali Can Metan'ın "**Sanal Ortamlarda Uzamsal Ses Kaynağı Kesimi**" isimli araştırması "İnsan Araştırmaları Komitesi" tarafından uygun görülerek gerekli onay verilmiştir.

Bilgilerinize saygılarımla sunarım.

Etik Komite Onayı  
Uygundur  
17/08/2015


  
Prof. Dr. Canan Sümer  
IAK Başkan Vekili  
ODTÜ 06800 ANKARA

Figure C.1: Approval of Ethics Committee