Procedural 3D Audio for AR Applications

Angeliki Skandalou
John Jeremy Ireland

Supervisor:
Michael Rose

Kongens Lyngby 2017

Abstract

This is the first paragraph

THis is the second

THird paragraph of abstract

Four paragraphs is enough I guess

Acknowledgements

We would like to express our gratitude and appreciation to our supervisor for his support and guidance throughout this thesis work. Several discussion sessions and advice helped us take the most out of this project and make this study possible.

We would like to express special thanks also to the rest of our classmates who did their thesis at the same time under the same supervisor and offered us their advice. Furthermore, we would like to thank **To do** name of the recording guy (1) for providing us with the acoustics room to perform recordings. And last but not least to our family and friends whose support throughout this thesis was invaluable.

List of Figures

List of Tables

Contents

Abbreviations

ASW Apparent Source Width.

AVIL Audio Visual Immersion Lab.

BRIR Binaural Room Impulse Response.

CS Compressive Sensing.

DOA Direction of Arrival.

ERB Equivalent Rectangular Band.

HATS Head And Torso Simulator.

HOA Higher Order Ambisonics.

HRTF Head-Related Transfer Function.

IACC Inter-Aural Cross Coherence.

ILD Inter-Aural Level Difference.

ITD Inter-Aural Time Difference.

STFT Short-Time Fourier Transform.

WFS Wave Field Synthesis.

Nomenclature

$\mathbf{\Omega}_{LS}$ Vector containing directions of Loudspeakers in reproduction.

$\mathbf{\Omega}_{L}$ Grid of directions used for the CS algorithm.

$\mathbf{\Omega}_{s}$ Subvector of $\mathbf{\Omega}_{L}$ containing only the prominent directions after CS processing.

$\breve{\mathbf{H}}$ Combined transfer matrix for mixed-norm problem.

$\breve{\mathbf{p}}$ Combined measurement pressure vector for mixed-norm problem.

$\mathbf{x}$ Combined amplitude.

ℓ_p Norm-p.

$\mathbf{H}$ Transfer Matrix for plane waves impinging on rigid sphere.

$\mathbf{p}$ Measurement vector for the pressure on the spherical array.

$\mathbf{x}$ Amplitude vector for plane waves impinging on the sphere.

$\widetilde{\mathbf{p}}$ Pressure vector reconstructed from prominent plane waves.

B_n^m Ambisonics coefficients.

L Number of plane waves in a discrete grid of directions.

LS Number of Loudspeakers in reproduction.

N Truncation order for the spherical Harmonic Functions.

P_n^m The associated Legendre polynomials of the first kind.

Q Number of sampling points on the spherical microphone array.

R_0 Radius of reproduction area.

Y_n^m Spherical harmonic Functions.

Ω Angular Dependency on both azimuth and inclination angle.

λ Regularization factor for natural field HOA processing.

$\mathbf{B}_N$ Ambisonics coefficients vector truncated at order N.

$\mathbf{S}$ Loudspeaker signals resulting from HOA decoding.

$\mathbf{W}$ Vector containing radial functions W_n.

$\mathbf{Y}_N(\mathbf{\Omega}_L)$ Spherical harmonics vector truncated at order N for all measurement angles in vector $\mathbf{\Omega}_L$.

$\mathbf{p}'$ Residual pressure.

ε Noise parameter for Compressive Sensing Algorithm.

a Radius of microphone array.

"w/ Residual" Exploiting the residual pressure (full implementation of signal path in Figure **??**).

"w/o Residual" Residual pressure is neglected (only upper path in Figure **??**).

Chapter 1

Introduction

Immersion and all these stuff that makes our thing good. Why we are doing it and what do we want to give to the community?

Audio in interactive projects like video games and VR/AR applications, plays a significant role for user immersion and realism. Visual and acoustic experiences are interconnected and lacking one of them spoils the whole experience.

The most difficult task is to produce realistic virtual sounds inside the application, difficult to distinguish them from the real ones. This can be achieved not only by playing back a realistic sound, but also by taking care of the environment effects and the context. For example, striking a nail on a board when it still vibrates from the previous struct, produces a different sound that gets added to the previous one [Cook, 2002].

Stuff about 3d sound as well.

Stuff about sound in general and description of the thesis

Sounds are strongly related to our everyday life and the ways we understand things. Through our experience, we can visualize an event by only hearing the sound it produces (e.g. a car approaching). The vibration caused by the collision of two objects produces sound that depends on the collision force, the duration of the interaction and the changes over time of it, but also on the size, shape, material and texture of the two objects. All these attricutes form a unique sound and the sound waves produced from the interaction give the information to the listener [Gaver, 1993b].

In this thesis we present an audio design tool made for Unity®software platform, for physics-based sound synthesis in virtual environments.

To do describe how much better it is to have procedural audio than pre-recorded sounds and on top of that physics-based sounds!! WOW!

Why is our method better that others? (eg wavetable)? And why we think this is the future of the audio in video games?

Chapter 2

Theoretical Background

Short overview of the theory parts
This is a way to link to explanations Direction of Arrival (DOA)
THis is a todo: **To do** todo test (3)
THis is smth done:

2.1 State-Of-The-Art

Wildes and Richards (1988) defined the angle of internal friction (ϕ), a shape-invariant acoustical parameter that heuristically categorizes sounds into material categories, as

$$\tan(\phi) = \alpha/\pi f, \tag{2.1}$$

where $\alpha = \tau_e$ is the damping coefficient, with τ_e being the time for the vibration amplitude to decay to $1/e$ of its original value after the object is struck and f is the vibration frequency[Giordano and McAdams, 2006].

Modal parameter extraction

- Van den Doel (Scanning physical interaction behavior of 3D objects): robotic device to measure impulse response of an object being struck in different points
- O'Brien et al (Synthesizig sounds from rigid-body simulation): FEM
- Raghuvanshi and Lin (Interactive sound synthesis for large-scale environments.): spring-mass system approximation
- Ren,Yeh,Lin (Example-Guided Physically Based Modal Sound Synthesis): one simple recording

To do turn the bullets into text (5)

This thesis adopted the parameter extraction from recordings. We used everyday objects, making it easy for us to record sounds and model them on the computer. We also chose this method to make it easy for users of our product to extend the repository of physically-based impact sounds of objects.

2.2 How physical attributes affect sound

To do Note smwhere that we're using only vibrating solid objects (not liquids).Gaver[Gaver, 1993b] pg10 has info (6)

Impact sounds consist of an excitation that dampens over time. Hence, the amplitude of the oscillation depends only on the damping. On the other hand, scraping sounds consist of continuous supply of energy that adds to the amplitude value on top of the damping of the oscillation throughout the object interaction. In addition, the force of the interaction plays the most significant role for the amplitude of the oscillation. The stronger the force, the biggest it imposes the amplitude value to be - and the louder the sound [Gaver, 1993b].

Furthermore, the material of the interacting objects affects their vibrating oscillation. Damping factor is a material-specific attribute and the bigger it is, the faster the objects lose energy and thus the oscillation lasts shorter. For example, wood has way bigger damping factor than metal and this is why they produce a "thud" and a "ringy" sound respectively.

Moreover, the configuration of the object also affect its vibration. The size of it determines how high or low pitched sound it will produce. More specifically, the smaller the object the more high pitched will be the sound.

Table 2.1 shows the acoustic information affected by each physical attribute.

Source	**Effects on the Sound Wave**
Interaction	
Type	Amplitude, spectrum
Force	Amplitude, bandwidth
Material	
Restoring force	Frequency
Density	Frequency
Damping	Amplitude, frequency
Homogeneity	Amplitude, frequency
Configuration	
Shape	Frequency, spectral pattern
Size	Frequency, bandwidth
Resonating cavities	Spectral pattern
Support	Amplitude, frequency, spectrum

Table 2.1: Acoustic effects of source attributes [Gaver, 1993b].

2.3 Modal Analysis

In this thesis we are using solid objects that are struck in different ways to produce sound. These ways could be falling on the floor or colliding with another object. The sounds produced can be impact, rolling or scratching sounds. When an object is struck, the forces applied cause deformations to it, emitting sound waves through the vibration of its outer surfaces [Van Den Doel et al., 2001].

Modal analysis studies the response of models under excitation. It uses the 3D model of an object to calculate its modal modes (vibration modes). There are multiple ways to do this, with the most accurate being FEM (Finite Element Method). The objective of FEM is to calculate the natural frequencies of a structure when it vibrates freely.

Another method for modal analysis is the "Exampled-guided", where data get extracted using example recordings of the objects being struck. Using a suitable algorithm it is easy

to extract features from the recordings such as the fundamental frequency and its harmonics and the frequency peaks of the signal.

2.3.1 Data Extraction

Modal analysis is performed before modal synthesis, to extract the necessary data. Modal synthesis is the sum of damped oscillators each corresponding to a modal frequency, as it will be discussed further below. The data needed for synthesis and their origin are shown in the table 2.2.

Symbol	Description	Derivation
A_n	Initial amplitude	Modal analysis
d_n	Damping	Material properties
f_n	Modal frequency	Modal analysis

Table 2.2: Derivation of data used in modal synthesis.

Since every different point being struck produces different deformations on the object, we need matrices of size N (N being the number of struck points of the object). More specifically, we need a vector **f** of size **N** corresponding to the modal frequencies of every point, a vector **d** of size **N** corresponding to the damping ratios and a matrix **A** of size **NxK**, where K is the number of modal frequencies calculated in one point, which corresponds to the amplitudes of each mode in every point of the object. All the above gives the modal model which can be symbolized as $\boldsymbol{M = \{f,\ d,\ A\}}$ [Van Den Doel et al., 2001].

2.4 Modal Synthesis

In the modal synthesis part, using the data extracted above, we synthesize the struck sound corresponding to the object. There are different ways to synthesize impact sounds, two of them being "Sinusoidal Additive Synthesis" and "Filter-based Modal Synthesis". The former uses exponential damping and the latter band-pass filters where the damping is the Q-factor of the filter.

2.4.1 Sinusoidal Additive Synthesis

At a struck point k when vibrating in mode n, the impulse response of the model is:

$$y_k = \sum_{n=1}^{N} A_{nk}\ e^{-d_n t}\ \cos(2\pi f_n t) \tag{2.2}$$

if $t >= 0$ and $y_k = 0$ if $t < 0$ [Van Den Doel et al., 2001].

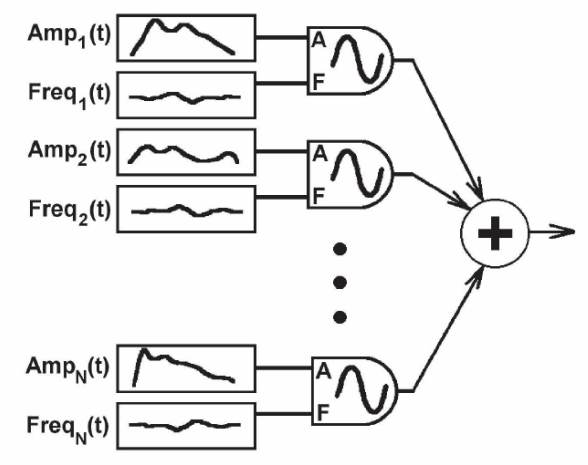


Figure 2.1: Sinusoidal Additive Synthesis Algorithm [Cook, 2002].

2.4.2 Filter-based Modal Synthesis

Band-pass Filters

At this point we will give some basic description of the band-pass filter since it is widely used in this thesis. Band-pass filters (BPFs) take a signal as input and give only a range of it as output, attenuating the rest of the frequencies. This range depends on the central frequency f_c. A filter of this kind is a result of a cascading of a low-pass and a high-pass filter circuit.

The passing range or "band" of frequencies is called **Bandwidth (BW)**. Defining as 0db the resonant peak, we can find the two cut-off frequencies ($f_{c_{\text{LOWER}}}$ and $f_{c_{\text{HIGHER}}}$) at -3dB. The range between them is the bandwidth (equation 2.3). In figure 2.2 we can see the frequency response of a BPF. [AspenCore, Inc.,].

$$BW = f_{c_{\text{HIGHER}}} - f_{c_{\text{LOWER}}} \tag{2.3}$$

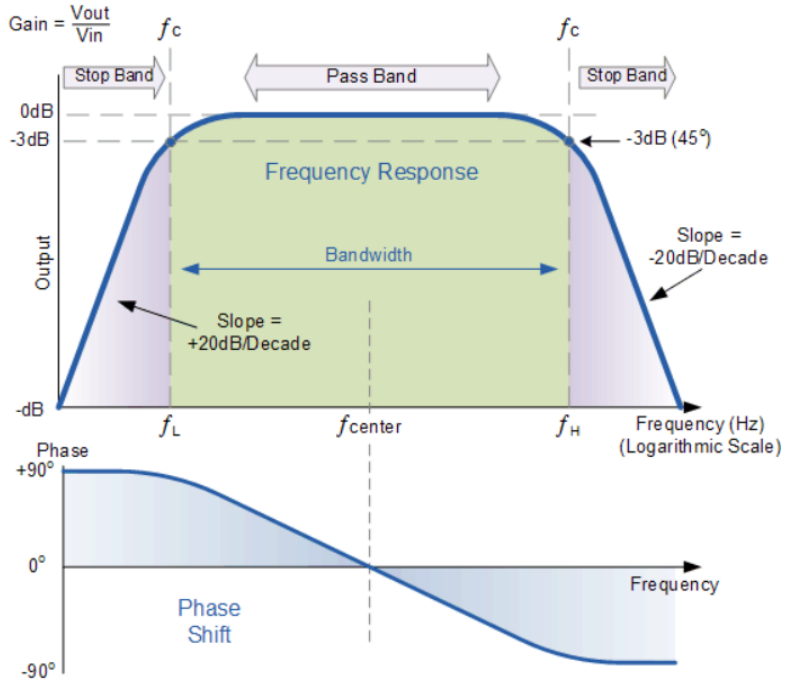


Figure 2.2: Frequency Response of a Band-pass Filter [AspenCore, Inc.,].

Synthesis

This method is also additive, since we are adding the outputs of a number of band-pass filters. To synthesize a sound using this method, we use as many filters as the modal frequencies. The filter takes as input an impulse, the center frequency which is the modal frequency and a **Quality factor (Q-factor)** which specifies the bandwidth of the filter. The Q-factor is calculated heuristically, depending on the material of the sound and is inversely proportional to the bandwidth ($Q = f_c/BW$), so the lower the Q-factor, the wider the bandwidth and vice-versa. Hence, more and less frequencies respectively will be included in the audible range. We call the above structure a *resonator*, which also includes a multiplication with the corresponding amplitude, taken from the A matrix.

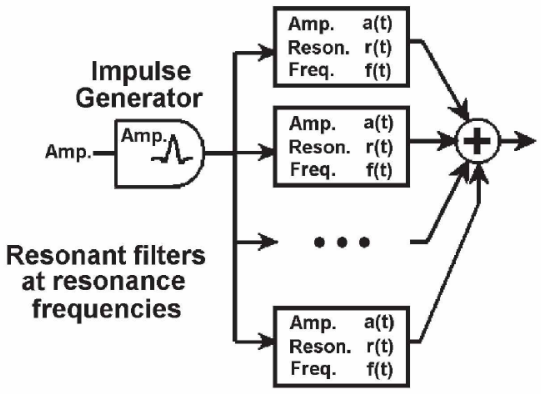


Figure 2.3: Filter-based Modal Synthesis Algorithm [Cook, 2002].

2.5 3D Audio

In VR/AR applications, the location of the incoming sound plays as significant role as the sound itself.

Chapter 3

Method

To do maybe name this chapter smth else?? (7)

A combination of the methods described in Chapter 2 is proposed in the present study.

3.1 Overview of our tool

The target of this thesis is to provide sound designers with an easy-to-use tool which enriches video games with procedural and physics-based audio.

To do replace figure with a better one (8)

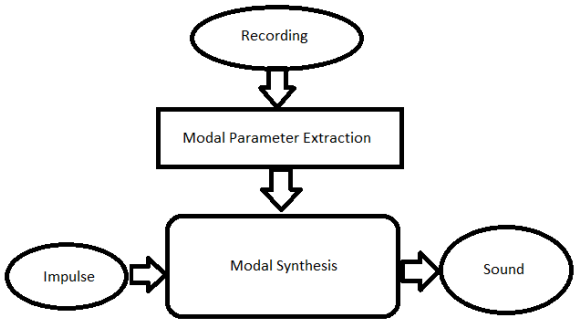


Figure 3.1: Sound Synthesis Procedure.

To obtain this tool, we followed the procedure described in figure 3.1. More specifically, the first step we carried out was to find several everyday objects, separate them into different areas that sound similar when struck and record impact sounds for each area. Afterwards, we used those recordings to extract data needed for sound synthesis. At the same time, we made a 3d model of every object we used. Next, we generated two different PureData patches, introducing two different synthesis methods.

Then, we combined the 3d models with their corresponding data and the synthesis patches inside Unity®software. This is where depending on the point where an object is struck, the corresponding frequencies and amplitudes are assigned to the patch and a sound is sent to the audio DSP chain.

3.2 Recordings

To obtain the necessary data we performed recordings of the sound produced of the object when being hit.

Since we only used simple shape everyday objects, we could easily assume that nearby points produce almost the same sound and thus separate the object into "modal areas" instead of calculating different modal matrices for each point. Tests in users proved that this accuracy-computational complexity trade-off was acceptable. A sample object and its division in areas is shown in figure 3.2.

Figure 3.2: Division of an object into areas with similar sound.

3.3 3D models

To create the 3D models we used Maya Autodesk software [Autodesk, Inc., b] and exported them as FBX® files [Autodesk, Inc., a] which is a format recognizable by Unity® software [Unity Technologies,].

3.4 Chuck language

Chuck language is a music programming language, made for "real-time sound synthesis and music creation" as mentioned in their website [Ge Wang,]. It's biggest advantage is the way it manipulates time. More specifically, the user specifies how long a sound will last, independent of other sounds that may play at the same time.

Modal features extraction code

We used the ChucK language at the starting point of our thesis to identify and extract the peaks of the recorded "wav" files. The algorithm used in this part of the thesis is made by Perry Cook for the course **"Physics-Based Sound Synthesis for Games and Interactive Systems"** held by *Perry Cook* and *Julius O. Smith* at **Kadenze Academy** [P. Cook, J. Smith,].

From a FEM analysis one can find out that each object vibrates in a very high number of modes. Although, most of them are inaudible and do not contribute to the sound model. Thus, we can easily define *ten* to be the total number of peaks identified. After some trials, we found out that *five* peaks are already enough. However, the authors recommend to use twice the sufficient number, hence *ten* for our case. Afterwards, the algorithm having taken a recording as input, computes its histogram and identifies its peaks. The frequencies where

peaks occur are the modal frequencies candidates. Depending on the numbers of peaks we chose, the algorithm outputs a normalization between high and low peaks. Finally, the algorithm finds the maximum value of the signal on each peak, calculating the corresponding amplitude of each mode.

3.5 PureData

Pure Data (pd) is another music programming language. It is open source and the main difference from ChucK language is that pd is a visual or "patcher" programming language, using objects instead of code, linked together to form a sequence [Miller Puckette,].

Re-synthesis patch

For reasons that will be explained below, we had to attach only one pd patch to every 3D object in the demo. Therefore, all synthesized sounds (impact, rolling and scratching) are being synthesized under one main pd patch.

The main part of the re-synthesis patch is the resonator. Using band-pass filters it gives the impulse response of an object struck at a specified point. It also clips the signal to give more brightness and harmonics. **To do** develop more (9)

3.6 Heavy Compiler

Heavy is a compiler that generates audio plugins from pd patches in interactive soudn and music applications [Enzien Audio, Ltd.,]. In this thesis we used it to compile pd patches into Unity audio plugins. Heavy's interface is their website where users upload patches and then are able to download the corresponding plugins and put them into their applications. The plugins we used consist of DLL files and a C# (Unity code) script that allows communication of the plugins with the rest of the script and also enables the sound card to play sounds.

3.7 Unity®

Unity® is a game engine software. This is where all previous work is combined together and gives the final product. For the purpose of this thesis, a sample scene is made inside unity where objects are struck in several points and produce different sounds.

The first part of the Unity implementation is the assignment of modal data to every different area of the object. This is done in $O(n)$ time for n modes.

To do describe onaudiofilterread() (10)

3.8 Microsoft Hololens Emulator

Chapter 4

Measurements

4.1 Recordings

The measurements were conducted in an anechoic chamber at DTU's Department of Electrical Engineering using a microphone placed one meter away from the struck object. To record the sounds we used Bruel & Kjær's half inch pressure field microphone type 4192 and microphone preamplifier power supply type 5935L, as well as the 744T digital audio recorder from Sound Devices at 44100 Hz sampling frequency. The setup can be seen in figure 4.1.

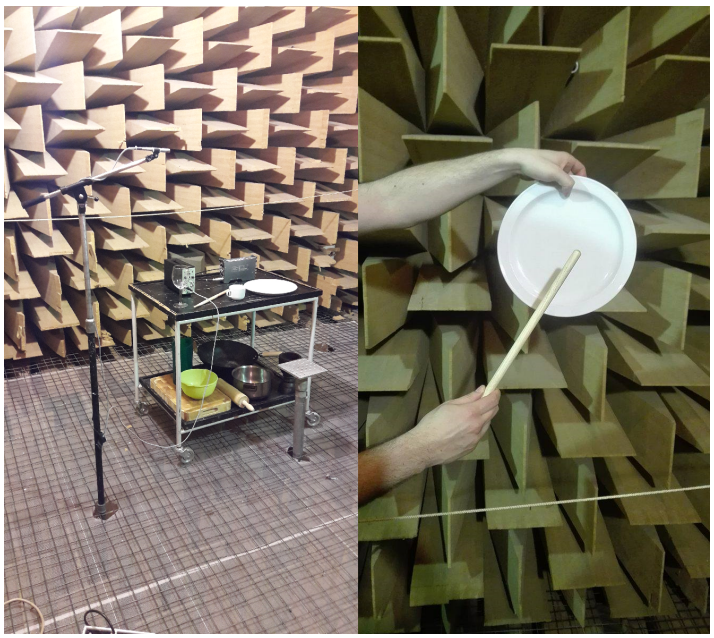


Figure 4.1: Picture of the setup for the measurements (left) and of a struck object (right).

To control the impact we hit the objects by hand with a wooden drumstick (figure 4.1) while trying to use the same impulsive force. Prior to the recordings, every object was divided into different surface areas depending on its shape and the sound produced by these areas. Therefore, several impact locations were chosen and recorded for every objects.

Eleven objects of everyday life made of five different materials (plastic, wood, ceramic, glass and metal) were used for the experiment. The idea of choosing these objects came firstly from the need of owning them (to perform the recording) and our ability to model them for

the demo (as they should be simple enough). Secondly, since we wished to test the immersion of the synthesized sounds on users, we wanted to be sure that the sounds used are familiar to them. In figure 4.2 both real and modeled objects are shown.

(a) Real objects.

(b) 3D models of the objects.

Figure 4.2: The eleven objects used in the thesis.

4.2 User tests

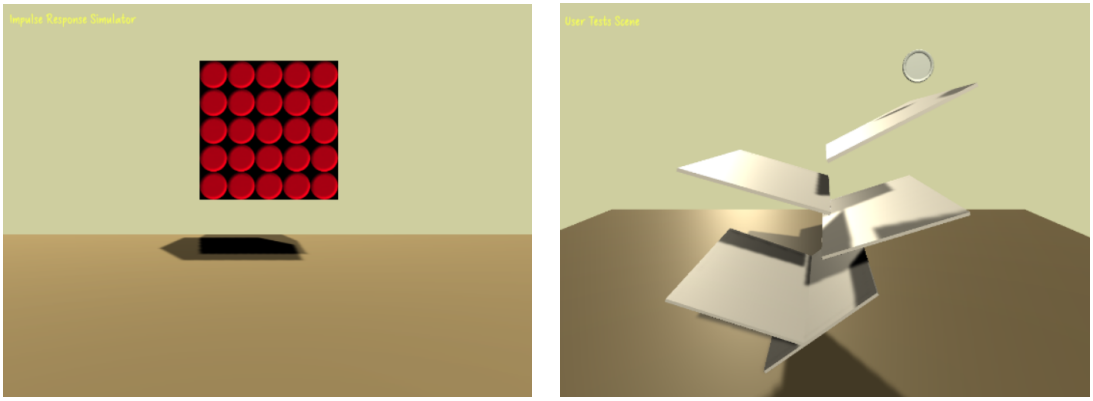
When working on a field where human perception plays a significant role, it is important to test whether your work makes sense for target users and if it solves successfully the problem that was designed for.

To examine the immersion of real-time produced and physics-based sounds on game players, we performed *MUSHRA*[Series, 2014] tests to people. Our aim was to answer the following questions: 1) Which of the two synthesis methods (sinusoidal and filter-based additive synthesis) is closer to reality? 2) Which is the range (in Q-factor values) of every material's sound? 3) Does physics-based synthesis make a sound more realistic and less boring? .

4.2.1 Preparation

For the tests we used several .wav files that we recorder inside the Unity®platform. We designed two special scenes for this purpose, because we wanted to test the sounds both in

real-life conditions -when an object is falling and colliding with other objects- and the impulse response of the synthesis model (figure 4.3).



(a) Impulse Response Simulator Scene. **(b)** Real-Life Conditions Scene.

Figure 4.3: The Unity Scenes designed to record the .wav files used for the user tests.

In the first scene (figure 4.3(a)), we recorded the audio files for the first experiment. During the recording session, we "tagged" the cube as different objects (a process that assigns to the cube different modal data) and let it touch the ground without any bounce. We also repeated the process with an Audio Source and the recordings files, to achieve consistency of the stimuli.

In the second scene (figure 4.3(b)), we recorded the audio files for the second and third experiment. In this scene, objects fall, roll and scratch freely on rotated platforms, simulating a real room with obstacles.

4.2.2 Stimuli

We performed three different tests. On the first test, the stimuli was 44 pairs of impulse responses (one for each synthesis method) corresponding to different areas of the eleven materials and their corresponding reference sounds which were the recordings of the actual sounds produced by the physical objects.

On the second test, the stimuli consisted of 33 pairs of sounds that were produced by falling objects. From the pair, one sound is produced when objects are split into "sound areas" and each area produces a different sound and the other sound is produced when every point of the objects makes the same sound. This single sound was the one produced from the area of each object where the recording taken (section 4.1) was closer to the real sound of this object in our opinion.

The stimuli of the third test was 50 sounds coming from five different objects, one of each material under testing (plastic jug, wooden mortar, ceramic plate, glass bottle and metallic cooking pot). For each object, we used 10 different sounds that corresponded to a small variation on the Q-factor which changes the material of the object. More specifically, starting from a value of 600, we increased the Q-factor by 200 up to 4600, removed some sounds that were too similar with others and provided participants with the rest.

4.2.3 Procedure

Stimuli was presented to the participants through **To do** describe headphones (11), in a room with reduced external noise.

In the first test, participants were provided with a reference sound (the actual recording) and two testing sounds (the sinusoidal and filter-based synthesized sounds), and they were asked to choose the one sounding closer to the reference. In the second test, participants were given pairs of sounds where the first one was made of multiple sounds per area of the object and the second one was made by the same sound assigned to every area. They were asked to answer which of the two they prefered. Finally, in the third test, they were provided with sounds from the same object but with different materials assigned to it and they were asked to point out when a change in material happens.

Every sound file starts 1 second after participant presses play and ends half a second after no sound can be heard.

4.2.4 Participants

To do number of participants, age, gender, normal hearing, job (12)

4.2.5 Test Results

Chapter 5

Implementation

All synthetic sounds have been created in Pd patches and are interpreted by Heavy which generates audio plugins and a C# interface for Unity. This C# script is attached to the GameObject in the scene so that the sound is processed within the game world.

We have created our own script that assigns to every one of the objects the modal parameters we extracted in the analysis part with the ChucK code. This is done independently of synthesis methods used here below.

5.1 Impact Sounds

5.1.1 Sinusoidal Additive Synthesis

In this section we describe in depth how the Pd patch corresponding to the sinusoidal additive synthesis of impact sounds works. The patch attempts to translate equation 2.2 into the programming language of Pd. Some of its terms are referenced in the following explanation.

First of all the frequencies and amplitudes matching the ten modes of the object are initialized. We can therefore feed these frequencies, which we identified as f_n in the equation 2.2, into the different oscillators. In Pd, oscillators output a cosine wave which is equivalent to $cos(2\pi f_n t)$ from the equation which suits our purpose perfectly.

We also translate into Pd the expression $e^{-d_n t}$ which corresponds to the damping of every mode n. Gaver [Gaver, 1993a] states that for each partial the decay rates d_n are controllable through a parameter D which corresponds to a material and that a useful heuristic, that we use in our patch, is to have $d_n = 2\pi f_n D$. By experimenting we established that values of D range from approximately 0.0002 for metal to about 0.05 for plastic sounds, with glass, ceramic and wood sounds in between. The higher the damping the higher the values. Then we multiply the damping by the partial's initial amplitude A_n to obtain an amplitude envelope that varies over time and which we multiply by the oscillator's signal. The output is what we call a partial which is illustrated in figure 5.1.

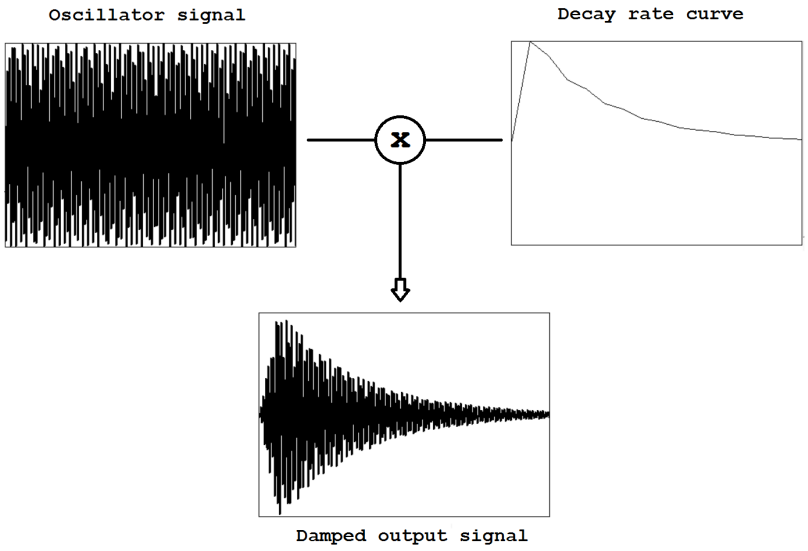


Figure 5.1: Diagram showing how the output partial is created from a 5000 Hz cosine wave and a decay rate curve with D = 0.005.

The final sound is produced by adding together the ten partials. The resulting signal is multiplied by the magnitude of the impact. For this we calculate the kinetic energy with Unity's physics compo As described in [Farnell, 2010], before sending the signal to the DAC we pass it through a clipper that gives richer harmonics and produces brighter sounds the stronger the impact is.

The patch produces an impact sound whenever the OnCollisionEnter() method from Unity is called. This is done when the collider that has the script attached to it touches another collider. When this happens we set the magnitude of the collision and then send an event to excite the patch. This is done by setting the value of t in 2.2 to zero which increases over time making the sound decay.

5.1.2 Filter-based Modal Synthesis

This synthesis method is based on the utilization of a bank of ten bandpass filters. Pd's bandpass filters have three control inputs as seen in figure 5.2. The left inlet is the incoming audio signal, the middle one sets the center frequency and the right input sets the Q factor value. The characteristics of these filters define the virtual object.

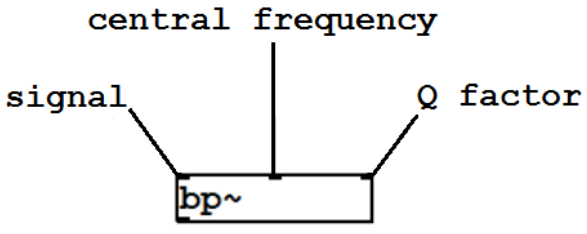


Figure 5.2: Pd's bandpass filter with its three inlets

The same way we did in the previous method, we initialize all ten frequencies and amplitudes of the object. Every frequency f_n is sent into a bandpass filter as the center frequency.

Every filter is characterized by its Q factor which is directly related to the damping. The higher the value of Q, the narrower the bandwidth and the less the resonator becomes damped.

Thus, Q determines the material of the impacted object [Gaver, 1993a]. By manipulating Q we can obtain different material sounds. Through experimentation we have found values of Q that range from about 20 for plastic to 5000 for metal.

To cause the object to sound we use a short impulse signal that excites the filter. The amplitude of the signal goes from 1 to 0 in 2 milliseconds as represented in figure 5.3. This impulse is multiplied by the value of the kinetic energy when the object impacts with another.

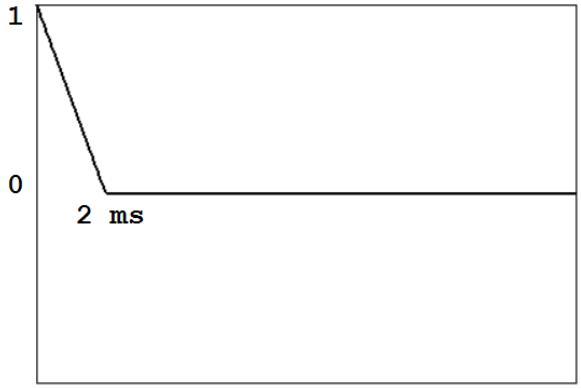


Figure 5.3: Impulse signal used to excite the bandpass filter.

The output signal of each of the ten filters is multiplied by the corresponding amplitude A_n of the mode. All ten resulting signals are added together. The signal is sent through a clipper as we did in the previous synthesis method.

5.2 Scratching Sounds

The sound produced by an object that is scraped across a rough surface can be assimilated to a succession of multiple impacts in a short time according to [Gaver, 1993a]. This paper shows that the resonant modes present in the spectrum of a struck object, are the same as when the object is scrapped. To model these modes we use the filter-based synthesis method.

In [Gaver, 1993a] and [Van Den Doel et al., 2001] the authors state that the center frequencies of the filters are scaled with respect to the contact velocity. The higher the velocity, the more the proportion of high-frequency energy increases.

We implement this by exciting the filters with a noise signal

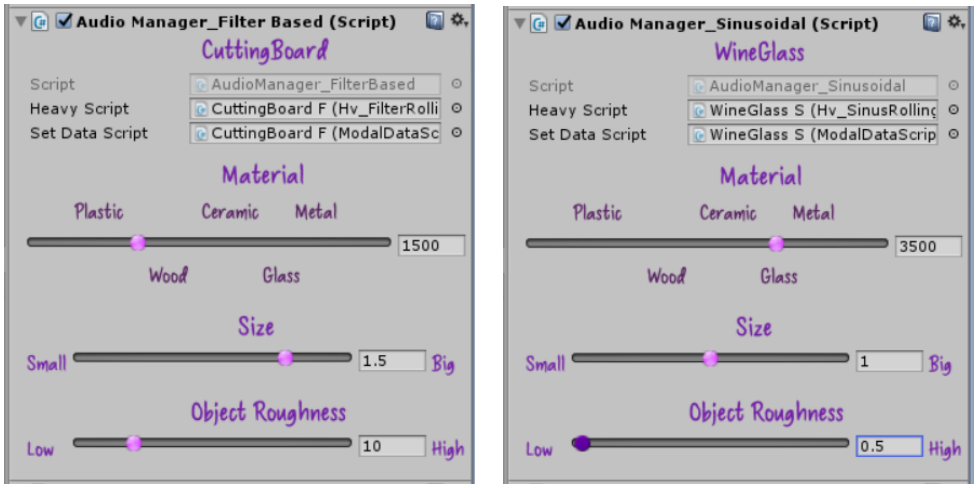
5.3 Rolling Sounds

5.4 User Interface

Here we will describe the *user interface (UI)* of the tool, where sound designers are able to choose the sound they prefer for every object.

The UI is made inside Unity®, by programming a custom inspector. *Inspector* in Unity®is a window that shows up after selecting an object, a file etc inside the platform and it displays all information relevant to it.

We also used a custom *GUISkin*, which is a set of settings about the Graphical User Interface (GUI).



(a) Filter Based **(b)** Sinusoidal

Figure 5.4: The custom inspector inside Unity platform.

When the designer wants to assign the procedural audio component on a Unity game object, he only has to select this game object and then from the Unity menu bar he can select one of the two available methods described above, as seen in figure 5.5.

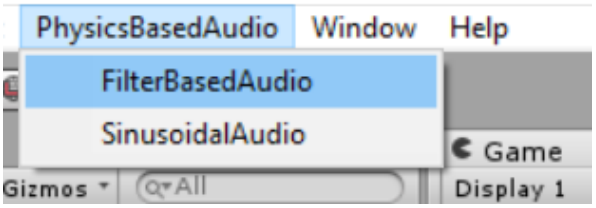


Figure 5.5: Designer can assign the audio manager from the menu bar.

Since every sound is produced from a corresponding real world object, the tool is restricted to the objects available up to this time. Therefore, the designer has to assign a **"tag"** of one of the eleven objects available on the tool (cooking pot, cup, cutting board, hug, mortar, bowl, plate, rolling pin, wine bottle, wine glass or wok). However, it is easy for her to contribute with new objects to the tool, by following the guide on appendix B.

5.4.1 Assignment of Different Materials

Different materials can be assigned to the objects made for this thesis. The designer is able to choose between *plastic, wood, ceramic, glass and metal* by adjusting the corresponding slider on the interface (see figure 5.4).

Metallic or glass sounds are more "ringy" than wooden or plastic ones that are more "thud". We achieve those sounds by changing the **Q-factor** of the **band-pass filters** used in the pd patch. Q-factor indicates the power loss in the filter. The higher the Q the less power is lost, so the resonator vibrates longer.

$$Q = 2\pi \frac{\text{maximum energy stored}}{\text{total energy lost per cycle at resonance}} \tag{5.1}$$

[D. Cory, I. Hutchinson, M. Chaniotakis, 2006]

Using trial and error method, we came out with an average value of the Q-factor for each material and we use it as the default value on the tool. Those values are shown in the table 5.1.

Material	Average Q-factor
Plastic	1000
Wood	1500
Ceramic	3000
Glass	3500
Metal	4000

Table 5.1: Default values of Q-factor for each material in the tool.

5.4.2 Changing the Size

In an application, the same object can appear in different sizes, so this thesis takes this into account. It is known that under the same excitation, the smaller the size of an object, the more high-pitched sounds it will produce, because the sound waves travel a smaller distance. Hence, we implemented a slider for the designer to choose the best sound that corresponds to the size of her object. We should note that the middle position of the slider (*size:1*) corresponds to the real object used for the data extraction.

5.4.3 Changing the Object Roughness

Another setting that sound designer is able to tweak is the object roughness. Even though our tool covers several different object materials, not every material has a uniform surface roughness when used in different objects. Adjusting a slider, the designer is able to choose a unique sound for every object of the same material.

5.5 still thinking of the title and placement of this section

5.5.1 Scaling

As mentioned above, impact sound gets more high-pitched when an object is scaled down and vice versa. To achieve a realistic scaling when the designer uses the build-in scaling feature of Unity®, the tool calculates the size of the game object on start. More specifically, a *scaling average* is calculated, taking into account all three dimensions (equation 5.2.

$$avgScale = (transform.localScale.x + transform.localScale.y + transform.localScale.z)/3; \tag{5.2}$$

avgScale is used as an adder to the *size parameter* described above. To avoid distortion in sound and to stay within the audible sound frequencies, we set a limit of adding 8.5, a number found heuristically. We consider this to be a good choice because Unity uses *meter* as the default unit and since we are mainly focusing on everyday objects, we find it rare for someone to use objects more than 8.5 times its original size.

Then, the tool checks whether a scale-up or a scale-down was executed. In the first case, we perform a normalization to 1/10th of the average scale value and we add it to the pitch multiplier **To do** reference the description of the pitch multiplier (13). We apply it to the size slider value and then to the pitch multiplier which we use to re-set the modal frequencies. To be more precise, instead of applying the actual pitch multiplier added with the average

scaling, we subtracted from 2 and then we use it ($2 - temp$ on the code below). This happens because we reversed the size slider. More specifically, the multiplier directly applied to the frequencies, increases them when it is bigger and decreases them when it is smaller. However, for convenient reasons, we wanted it to be the bigger the multiplier, the bigger the object and reverse. Since 2 is the biggest value, we normalized it to be the smallest one.

In the second case, we subtract the value from the pitch multiplier. We do not need to normalize the average scaling value, because it is already between 0 and 1. Afterwards, we follow the same procedure as above, with one difference; instead of subtracting the new pitch multiplier from 2 we add it to 1. This happens because now the biggest value of the size slider is 1 -since above this it counts as a scale-up- and we still want the reversed value for the slider, so we subtract the subtracted value, making it a plus (+).

```
1  // Scale-up
2  if (avgScale > 1f)
3  {
4    // Normalization
5      avgScale /= 10f;
6      // Add to the pitch multiplier
7      float temp = SetDataScript.multiplier +avgScale;
8      // Apply to the size slider
9      HeavyScript.SetFloatParameter(Hv_FilterRolling_AudioLib.Parameter.Size, 2-
       temp);
10     // Apply to the pitch multiplier
11     SetDataScript.multiplier = 2-temp;
12     // Re-set the modal frequencies
13     SetDataScript.SetTheFreqs();
14 }
15 // Scale-down
16 else
17 {
18   // Subtract from the pitch multiplier
19     float temp = SetDataScript.multiplier - avgScale;
20     // Apply to the size slider
21     HeavyScript.SetFloatParameter(Hv_FilterRolling_AudioLib.Parameter.Size, 1 +
       temp);
22     // Apply to the pitch multiplier
23     SetDataScript.multiplier = 1+ temp;
24     // Re-set the modal frequencies
25     SetDataScript.SetTheFreqs();
26 }
```

5.5.2 OnCollisionEnter

This is were the tool detects a collision of an object with something else. The collision could be either with the ground, another object from the tool or just an object in the scene.

The first thing that happens when a collision takes place, is to identify what kind of object is the one that collided with something. Hence, a function is called

Chapter 6

Results & Discussion

6.1 Results

6.2 Discussion

6.2.1 Bugs

- you have to press apply on prefabs
- the procedure of putting the freq and ampl data in
- it is object specific
- lack of acoustical richness that might characterize synthetic signals [Giordano and McAdams, 2006]

6.2.2 Which Synthesis Method Is Better?

6.2.3 Why our work can be used in VR/AR?

6.2.4 Did we manage to achieve what we wanted?

6.2.5 How can we improve our work?

- take into account the environment (reverberation etc)
- make objects destructible

6.2.6 Pros and Cons of Procedural Game Audio

Chapter 7

Conclusion

This is the conclusion
4-5 paragraph approx

Appendix A

Results of tests to users

Appendix B

User Guide to our product

Bibliography

[AspenCore, Inc.,] AspenCore, Inc. Passive band pass filter. http://www.electronics-tutorials.ws/filter/filter_4.html. [Online; accessed 28-April-2017].

[Autodesk, Inc., a] Autodesk, Inc. Fbx file format. https://www.autodesk.com/products/fbx/overview. [Online; accessed 2-May-2017].

[Autodesk, Inc., b] Autodesk, Inc. Maya Autodesk (version 2016). https://www.autodesk.com/products/maya/overview. [Online; accessed 20-April-2017].

[Cook, 2002] Cook, P. R. (2002). *Real Sound Synthesis for Interactive Applications*. A. K. Peters, Ltd., Natick, MA, USA.

[D. Cory, I. Hutchinson, M. Chaniotakis, 2006] D. Cory, I. Hutchinson, M. Chaniotakis (Spring 2006). 6.071j Introduction to Electronics, Signals, and Measurement. Massachusetts Institute of Technology: Mit OpenCourseWare. https://ocw.mit.edu. License: Creative Commons BY-NC-SA.

[Enzien Audio, Ltd.,] Enzien Audio, Ltd. Heavy (version r2017.02). https://enzienaudio.com/. [Online; accessed 2-May-2017].

[Farnell, 2010] Farnell, A. (2010). *Designing sound*. Mit Press.

[Gaver, 1993a] Gaver, W. W. (1993a). How do we hear in the world? explorations in ecological acoustics. *Ecological psychology*, 5(4):285–313.

[Gaver, 1993b] Gaver, W. W. (1993b). What in the world do we hear?: An ecological approach to auditory event perception. *Ecological psychology*, 5(1):1–29.

[Ge Wang,] Ge Wang. Chuck programming language. http://chuck.cs.princeton.edu/. [Online; accessed 2-May-2017].

[Giordano and McAdams, 2006] Giordano, B. L. and McAdams, S. (2006). Material identification of real impact sounds: Effects of size variation in steel, glass, wood, and plexiglass plates. *The Journal of the Acoustical Society of America*, 119(2):1171–1181.

[Lloyd et al., 2011] Lloyd, D. B., Raghuvanshi, N., and Govindaraju, N. K. (2011). Sound synthesis for impact sounds in video games. In *Symposium on Interactive 3D Graphics and Games*, pages PAGE–7. ACM.

[Miller Puckette,] Miller Puckette. Pure data programming language. http://puredata.info/. [Online; accessed 2-May-2017].

[P. Cook, J. Smith,] P. Cook, J. Smith. Physics-based sound synthesis for games and interactive systems. https://www.kadenze.com/courses/physics-based-sound-synthesis-for-games-and-interactive-systems-iv. [Online; accessed 2-May-2017].

[Ren et al., 2013] Ren, Z., Yeh, H., and Lin, M. C. (2013). Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics (TOG)*, 32(1):1.

[Series, 2014] Series, B. (2014). Method for the subjective assessment of intermediate quality level of audio systems. *International Telecommunication Union Radiocommunication Assembly*.

[Unity Technologies,] Unity Technologies. Unity® software (version 5.5.1f1 personal). https://unity3d.com/. [Online; accessed 2-May-2017].

[Van Den Doel et al., 2001] Van Den Doel, K., Kry, P. G., and Pai, D. K. (2001). Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 537–544. ACM.

To do...

- ☐ 1 (p. v): name of the recording guy
- ☐ 2 (p. 1): describe how much better it is to have procedural audio than pre-recorded sounds and on top of that physics-based sounds!! WOW!!
- ☐ 3 (p. 3): todo test
- ☑ 4 (p. 3): this is done
- ☐ 5 (p. 3): turn the bullets into text
- ☐ 6 (p. 4): Note smwhere that we're using only vibrating solid objects (not liquids).Gaver[Gaver, 1993 pg10 has info
- ☐ 7 (p. 9): maybe name this chapter smth else??
- ☐ 8 (p. 9): replace figure with a better one
- ☐ 9 (p. 11): develop more
- ☐ 10 (p. 11): describe onaudiofilterread()
- ☐ 11 (p. 15): describe headphones
- ☐ 12 (p. 16): number of participants, age, gender, normal hearing, job
- ☐ 13 (p. 21): reference the description of the pitch multiplier