Room Reverberation Simulation using Parametrised Feedback Delay Networks

Master Thesis

Daniel Strübig

Aalborg University
Sound and Music Computing M.Sc.

Electronics and IT
Aalborg University Copenhagen
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Room Reverberation Simulation using Parametrised Feedback Delay Networks

Theme:
Spatial Audio

Project Period:
Summer Semester 2020

Project Group:
Individual

Participant(s):

Daniel Strübig

Supervisor(s):
Stefania Serafin
Jesper Udesen

Copies: 1

Page Numbers: 50

Date of Completion:
June 4, 2020

Abstract:

Artificial reverberation synthesis describes filter structures that approximate room reverberation. Given a set of acoustic phenomena and parameters such as frequency absorption, reverberation time and room size, these reverberator structures emulate the given effects for early reflections and the diffuse reverberation tail. In this report, a real-time implementation of a reverberator structure is described, using delay lines and allpass filters for early reflections and a feedback delay network topology for the late reverberation tail. The output of the reverberator structure is compared against two commercially available reverberators. The results show a vast numerical difference in the system's impulse responses. Listening tests using the reverb signals in conjunction with the Unity® engine reveal that there is little perceptual difference across the reverberators. For future improvements, different test conditions and comparisons with more suitable frameworks are suggested.

Contents

0.1 List of Abbreviations and Terms

IR	Impulse response
HRIR	Head-related impulse response
BRIR	Binaural room impulse response
RIR	Room impulse response
HRTF	Head-related transfer function
dB	deciBel
ISM	Image source model
JSON	Javascript object notation
SOFA	Spatially Oriented Format for Acoustics
ITD	Interaural Time Difference
ILD	Interaural Level Difference
LTI	Linear, time-invariant

0.2 List of Symbols and Math Operations

c	Speed of Sound
RT_{60}	Reverberation Time
V	Volume
A	Surface area
ω	Radial frequency
φ	Azimuthal angle
θ	Elevational angle
$h(n)$	Impulse response
B	Interaction Matrix
$H(Z)$	Transfer Function
N	Order of reflections
d	Distance
l, w, h	Room dimensions
c	Speed of sound
$\vec{r}$	Listener location
$\vec{s}$	Source location
α	Absorption coefficient
δ	Dirac delta function
g	Impulse gain
τ	Delay

Preface

Aalborg University, June 4, 2020

Daniel Strübig
dstrub18@student.aau.dk

Chapter 1

Introduction

In the field of auralization in enclosed spaces, both perceptual credibility as well as computational efficiency are of essence [39]. Based on perceptual parameters and physical phenomena in room acoustics, a plethora of models have been invented and refined [41]. Oftentimes, this process involves operations that are computationally expensive. Due to that, numerous solutions have been developed for offline-computation, i.e. based on specified static room geometries, room impulse responses are generated for fixed receiver points [36][6].
However, due to recent hardware developments and improvements, auralization is on the verge of being implemented in real-time environments: With changing room geometry as well as source and receiver locations, the room impulse responses are generated and updated.
This report describes the development of a reverberator using feedback delay networks. In the time domain, a room impulse response can be compartmentalised into three components: The initial impulse, early reflections and a late reverberation tail [22]. The early reflections are a filter-based implementation consisting of delay lines, lowpass and allpass filters. The late reverberation tail is modeled perceptually with a feedback delay network, is parametrised such that frequency-dependent reverberation times can be controlled [20]. All implementation is carried out in the Unity® game engine [35], whereas the feedback delay network is developed using JUCE [14]. The resulting impulse responses are then evaluated in comparison with impulse responses from comparable frameworks [33][23]. Moreover, limitations, drawbacks and possible improvements of the the hybrid acoustic model are discussed.

1.1 Motivation

In the context of game development, reverberation is an integral part of creating a sense of immersion [18]. However, in large-scale projects with a large number of virtual rooms, specifying reverberation settings for each individual room can be a tedious task. Addressing this issue, the objective of this report is therefore:

1. Design a perceptually plausible reverberator for indoor room acoustics.

2. Account for a set of room acoustics parameters, such as room size, absorption characteristics and source-receiver distance.
3. Leave room for design parameters by game developers.

Given these objectives, a suitable extension of the feedback delay network is deployed [20]. This allows for sensible parametrisation of the room impulse response, which is explained in chapter 2. An extensive derivation of filter models, the feedback delay network and its parametrisation is given in chapter 3. Implementation details using the real-time audio development framework JUCE [14] and the game engine Unity® [35] are covered in chapter 4. Subsequently, the proposed reverberator is tested against 2 commercially available frameworks on its accuracy and perceptual qualities. The results and test setup are reviewed critically and this report is concluded with remarks on possible improvements, extensions and outlooks.

Chapter 2

Related Works

This section provides an overview of components and procedures to achieve auralization digitally. A set of psychoacoustic phenomena is described, along with models and simplifications that approximate these phenomena. Moreover, the properties of room impulse responses are explained and a selection of room reverb approximations are given.

For clarity, the terms **sound source** and **emitter** are used interchangeably throughout this chapter, so are the terms **listener** and **receiver**.

2.1 Reverberation

Vorländer established two principal components of auralization: The spatial impression and the local impression [38].

As a high-level abstraction, reverberation constitutes the effect that a given room has on a signal that is emitted by a sound source and reaches the receiver [39]. An established set of effects are frequency absorption by the propagation medium (air), damping, absorption by materials as well as scattering and energy loss at surfaces with sound propagating at $c = 343\frac{m}{s}$ at a room temperature of $21°C$. The influence of these properties can be captured in a **room impulse response (RIR)**, assuming that these influences are of linear behaviour. Convolving any arbitrary signal emitted by a sound source with said room impulse response constitutes the reverberant signal [39].

2.1.1 RIR Properties

The overarching property of room impulse responses its exponential decay over time. Therefore, a metric to evaluate room impulse responses is the time at which its energy decays to a certain level. The RT_{60} describes the time needed until the sound pressure level has decreased by 60 dB less, relative to its original impulse:

$$RT_{60} = 0.161\frac{V}{A} \tag{2.1}$$

with $A = \alpha S$, S being the surface area of a given room, α the absorption coefficient of said surface and V the volume of the room [39].

Similarly, Barron et.al. established a compartmentalization scheme for room impulse responses in the time domain [2]:

1. Direct impulse (1 - 5 ms)
2. Early reflections (6 - 200 ms)
3. Late reverberation tail (> 200 ms)

Described by Gardner, the early reflections of a room represent the acoustic modes that are innate to the room due to its geometric properties [8]. After a certain point, these modes overlap such that the reflections appear randomized and therefore resemble a noise-like signal [13]. Due to that, a set of approximations have been developed to recreate these properties.

2.1.2 Geometric Models

In its most basic form, reverberation can be achieved by the analysis of a three-dimensional space and inferring a set of physical properties [41]. Based on source-receiver positions as well as room dimensions and assumptions about wave propagation, geometrical models of room reverberation offer a way to derive a room's impulse response. The image-source reverberation model by Allen and Berkley reproduce reflections by mirroring a source at the room's boundaries and produces deterministic results [1], as the receiver's, emitter's and boundary positions are known. The model has laid groundwork for many extensions ever since [5], yet does not account for scattering at boundaries. Furthermore, it is assumed that waves travel along straight lines.

Similar in its approach is the concept of ray tracing. While the receiver's, emitter's and boundary positions are known, the ray tracing algorithm emits a number of rays in randomly distributed directions. Those rays, that intersect with the receiver constitute the impulse response [36]. This accuracy of the algorithm can be extended by increasing the number of rays. However, due to its stochastic approach, this method leads to increased computational time [39].

Recent advancements by Raghuvanshi et. al. incorporate a custom solver for the wave equation and allow for pre-computing the room impulse response at certain positions and interpolating between them [22]. In the light of current hardware improvements for interactive applications, it can be assumed that this approach is of relevance in applied research in the near future.

2.1.3 Algorithmic Reverberation

As stated initially, the modelling of sound propagation relies on making simplifications and assumptions about its effect on a signal. First developed by Schroeder, the series of reflections caused by a room are recreated using a series of allpass filters. This filter type has a flat frequency response, yet has an effect on the phase component of the input signal. The length of the cascaded filters correspond to the size of the room that is approximated [29]. This basic filter model has been extended by Moorer, which incorporates a lowpass filter to simulate the high frequency absorption caused by air [37].

As the frequency response of the Schroeder allpass cascade depends entirely on the individual delay line lengths, the frequency response of this reverberator is oftentimes uneven. Therefore, it is not a suitable method to model the noise-like late reverberation tail. Established by Jot and Chaigne, the feedback delay network solves this issue. By analysing and constraining the eigenfrequencies of the reverberator, a late reverberation tail with an even frequency response can be achieved [13]. The basic feedback delay network has been extended and parametrised in various ways to improve its parametrisation and pulse density [20][26]. An extensive analysis is given in chapter 3.

2.2 Binauralization

The second phenomenon established by Vorländer is Binauralization [39]. Therefore, the basic concepts and properties of binauralization are dissected and explained.

Constituted by Blauert et. al., the human anatomy adjacent to the ears is mainly responsible for the human ability to localize sounds [4]. The shape of a person's head, the position of the shoulders and the form of the pinna influence the sound waves that reach the eardrum. Assuming again that all these influences are linear, this effect can be characterised by an impulse response. As the human head is the reference in this impulse response, it is therefore referred to the head-related impulse response (HRIR). Similarly, the head-related transfer function (HRTF) represents these effects in the frequency domain. Hence, convolving any arbitrary signal with a HRIR will result in the binauralized signal.

HRIRs are usually indexed by the emitter's position relative to the listener in polar coordinates, where φ denotes the angle on the horizontal plane (azimuth), Θ the angle on the sagittal plane (elevation) and r the respective distance.

There is a plethora of influences that improve or diminish the perceptual qualities of HRIRs, summarised in system theory form by Völz [42]. Yet, two factors are of importance:

1. **Interaural level difference** (ILD): The effect, that the incoming sound pressure will be different for both ears depending on the angle between the source and the listener [3]. This is due to the distance between the source and the left and right ear respectively.
2. **Interaural time difference** (ITD): The delay between the left and the right ear when subjected to an auditory stimulus.

It is important to note that this is only a subset of all physiological and cognitive aspects of spatial hearing and mostly Blauert, Moller and Vorländer's work are of significance to this field [4] [17] [39]. Generally, the recording of an HRIR and its playback bear many factors that could diminish its perceptual plausibility [42]. Yet, a notable determining factor is the personalization of HRIRs. As explained above, the human anatomy shapes how sound is received, which is individually unique. Therefore, recent advancements in this field include simplifying and improving the personalization process of HRIRs by e.g. leveraging machine learning [32].

2.3 Frameworks and Projects

As of this date, a plethora of different frameworks and libraries exist to model impulse responses. A selection with relevance to this project is given below:

- **Evertims**: Developed by Poirot et. al., this is an extension to the 3D and physics simulation engine Blender® [7]. The early reflections are computed using the image source model described previously. The late reverberation tail is based on a feedback delay network [19]. It offers binauralization options, ambisonic rendering support and dynamic impulse response calculation based on the room geometry specified in Blender.
- **Wayverb**: This project developed by Reuben Thomas at the university of Huddersfield uses the image source model to find early reflections. The late reverberation tail is approximated by a ray tracing algorithm. Moreover, low-frequency room modes are found using a digital waveguide mesh [36]. The developed application computes the room impulse response offline for later usage.
- **RAVEN**: Implemented by Vorländer and Schröder at RWTH university in Aachen, this framework also computes the impulse response based on the image source model and stochastic ray tracing techniques [30]. It also incorporates other effects such as diffraction and allows for flexible geometry during runtime [30].

2.3.1 Summary

As a high-level abstraction, the propagation behaviour of a room can be defined by its impulse response, assuming that these influences are linear. Although some physical effects of room reverberation such as scattering at boundaries are not linear [24], many geometric models have been developed to derive a room's impulse response [1]. Simultaneously, the algorithmic models using filter theory incorporate these effects as well. Based on the Schroeder allpass filter, the plausibility of these effects have improved with parametetrised models of comb filter structures and feedback delay networks [13].

The second component, binauralization, incorporates the effects that the human anatomy has on the emitted signal. Again, these effects are assumed to be linear, so that these can be replicated by a head-related impulse response. Recent efforts include the personalization of HRIRs with leveraging modern machine learning techniques.

Chapter 3

Fundamentals

As established in section 2, the overarching premise of algorithmic reverberation is to assume that all modeled acoustic phenomena are linear and can thus be described by a linear, time-invariant system. Therefore, convolving any arbitrary signal with the system's impulse response will result in the reverberant version of that signal. In this chapter, the underlying principles of algorithmic reverberation are covered. Starting with the early prototypes and later on covering more sophisticated systems, the structure of each section consists of stating the underlying idea, its mathematical implications and the problems and shortcomings it encounters. As a general convention, f_s represents the sampling frequency, $y[n]$ the output and $x[n]$ the input of a system at a given discrete time step n. Variables named b denote feedforward coefficients, a denote feedback coefficients. ω represents the radial frequency, $T = \frac{1}{fs}$ and $j = \sqrt{-1}$.

3.1 Comb Filter

In its most simple form, the comb filter consists of a delay element and a feedback path, scaled by a coefficient. This can be expressed as follows:

$$y[n] = b_0 x[n] - a y[n - m] \tag{3.1}$$

with m being the length of the delay element. With this structure, the impulse response of this filter is infinite, as the feedback path feeds the system continuously. In the frequency domain, this filter introduces poles at multiples of m, with its transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^m}{z^m - a} \tag{3.2}$$

assuming that $b_0 = 1$.

On the unit circle, this filter can be interpreted as a system with m poles in equal distance to the origin, which are in turn dependent on a and zeros where $z^m = a$.

Similarly, this filter can be implemented as a feedforward filter by:

$$y[n] = b_0 x[n] + b_M x[n - m] \tag{3.3}$$

This, in turn, introduces zeros at the multiples of m.

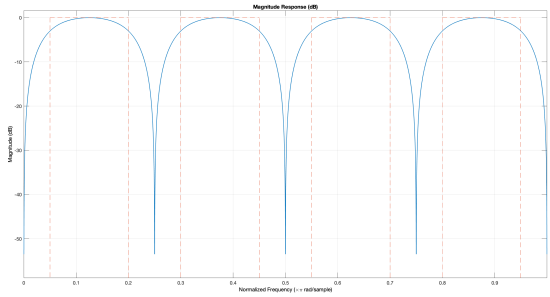


Figure 3.1: Frequency Response of a feedback comb filter

3.2 Allpass Filter

As established by Schroeder, coalescing a feedback and feedforward filter yields an allpass filter. Combining eq. 3.3 and 3.1 results in

$$y[n] = b_0 x[n] + x[n-m] - a_m y[n-m] \tag{3.4}$$

Taking the z-transform yields

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + z^{-m}}{1 + a_m z^{-m}} \tag{3.5}$$

Assuming $b_0 = a_m$ and substituting z with the complex sinusoid $e^{-j\omega T}$ results in the magnitude response [31]

$$|H(e^{-j\omega T})| = |\frac{a + e^{-j\omega MT}}{a + e^{j\omega MT}}| = 1 \tag{3.6}$$

As an abstraction, the zeros of the feedforward path cancel out the poles of the feedback path, as shown in fig. 3.3. The system's frequency response is flat as indicated in fig 3.2. It does however introduce drastic phase shifts. With this property, the concept of the allpass reverberator by Schroeder aims to model overlapping reflections in a given room. [29].

With these components at hand, the initial prototype by Schroeder and Logan consists of a structure of parallel comb filters followed by serialized allpass filters, as presented in fig. 3.4.

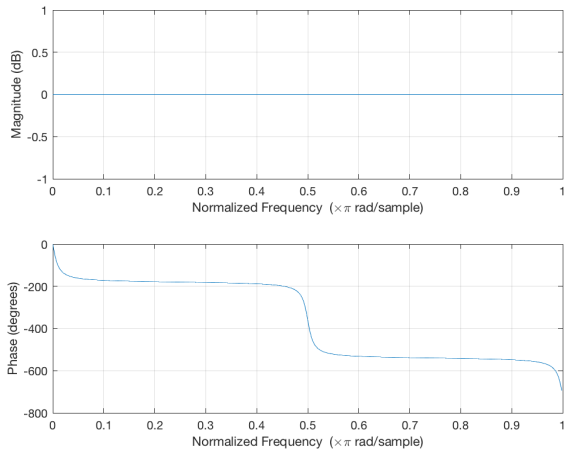


Figure 3.2: Frequency Response and Phase Response of 4th order Allpass Filter

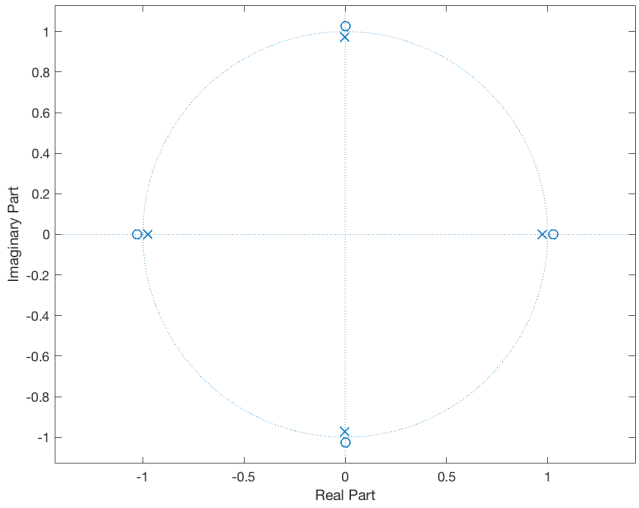


Figure 3.3: Pole-Zero plot of 4th order Allpass Filter

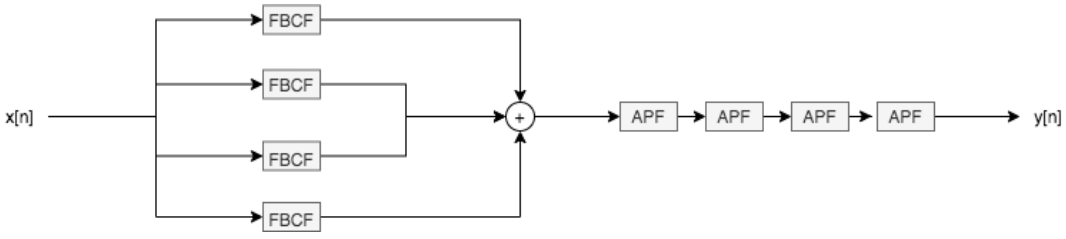


Figure 3.4: Comb filter structure proposed by Schroeder and Logan in [29]

3.3 Feedback Delay Network

Having introduced Feedback Delay Networks in section 2.1, a few questions arise:

- How to extend the feedback comb filter whose resulting impulse response fulfils the characteristics of a room impulse response described in section 2.
- How to find a sensible relation between LTI systems and perceptual qualities of reverberation such as the T_{60} reverberation time.
- How to tune the comb filter structure in such a way, that accurate frequency-dependent reverberation times can be achieved.

3.3.1 Topology

Introduced by Jot and Chaigne in 1991, the Feedback Delay Network proposes the following improvements to the aforementioned comb filter structure: Following the perceptual qualities of room impulse responses described in section 2, the pulse density has to increase linearly over time. Moreover, the frequency response of the system should be flat at all times to incorporate the noise-like perceptual quality of a room impulse response.

In the solution by Jot and Chaigne, the parallel comb filter structure remains. Yet, to increase the pulse density, a matrix in the feedback path of all comb filter is inserted. Conceptually, this allows each parallel comb filter to be connected to the remaining comb filters of the system. To ensure stability, the feedback matrix is chosen to be unitary.

3.3.2 Attenuation Filter

A second major suggestion by Jot and Chaigne is to introduce an attenuation filter to allow for a frequency-dependent reverberation time, which is explained as follows: Given a single feedback comb filter, an attenuation filter with the transfer function $A(z)$ is inserted at the end of the delay line. The choice, structure and sophistication of this attenuation filter determines, to which degree the resulting feedback delay network can be parametrised [13]. Inserting this attenuation filter results in the following transfer function of the comb filter structure:

$$H(z) = \frac{1}{1 - A(z)z^{-m}} \tag{3.7}$$

From eq. 3.7 alone, it is evident that A(z) is dependent upon the length of the delay line m.

Replacing z with the complex sinusoid, the magnitude response of the attenuation filter can be written as a function of ω in dB:

$$A(\omega) = 20\log_{10}|A(e^{j\omega})| \tag{3.8}$$

Bearing in mind that T_{60} is usually expressed in seconds, the individual gains for a frequency band with center frequency ω can be achieved by finding a certain gain γ for this specific band. This can be calculated according to [13]:

$$\gamma_{db}(\omega) = \frac{-60}{f_s T_{60}(\omega)} \tag{3.9}$$

In principal, the attenuation filter could be any arbitrary filter structure. In its original implementation, a lowpass filter was used. In recent extensions, a graphical equalizer for a frequency-dependent reverberation time was used [20]. Therefore, the reasoning and relations in the works by Prawda et.al. will be presented [20].

Recalling the delay line element in the feedback comb filter, the consequence is that longer delay line decay faster than short delay lines. From a design perspective, the desired reverberation time in seconds is usually known and should be achieved. Relating the absorptive filter to the desired reverberation time, the following condition needs to be fulfilled [20]:

$$A_{dB}(\omega) = M\gamma_{dB}(\omega) \tag{3.10}$$

3.3.3 Graphic Equalizer

As mentioned, the approach taken in [20] and implemented in this project utilizes a graphic equalizer with fixed frequency bands. Structurally, a graphic equalizer is a series of peak filters, in this case a second order peak filter as described by Orphanidis in [25]. Given N peak filters with an overall gain of g_0, the magnitude response of the entire series of peak filters can be formulated as:

$$H_{dB}(e^{j\omega}) = g_0 + \sum_{n=1}^{N} H_{dB,n}(e^{j\omega}) \tag{3.11}$$

Given in [25], a peak filter can be described by its center frequency, bandwidth and gain. When cascading peak filters, one inherent issue is the gain of one peak filter leaking of into the subsequent peak filter. Unless the peak filters are purposely designed to have a very small bandwidth, this effect is likely to take place. In the delicate case of the graphic equalizer being responsible for controlling the reverberation time of a certain frequency band, this can lead to instability. As proposed in [40], this issue can be overcome by storing the magnitude response of each individual peak filter on K additional frequency points. Given a graphic equalizer of N cascaded peak filters, the interaction matrix between all filters within the graphic equalizer structure can be stored according to [40]:

$$\mathbf{B_{k,n}} = H_{dB,n}(e^{j\omega_k})/g_{p,n} \tag{3.12}$$

Recalling the topology of the Feedback Delay Network, $g_{p,n}$ is the gain factor which is common to all graphic equalizers on each comb filter path. Having covered the graphic equalizer, this can now be used as the attenuation of the Feedback Delay Network. Recalling eq. 3.10, γ can be replaced by $\mathbf{B}$, yielding the following:

$$A(\vec{\omega}) = \mathbf{B}g_p \tag{3.13}$$

in which $A(\vec{\omega})$ is now a vector of K frequencies.

With this interaction matrix at hand, it can be ensured that additional leaking of gains between bands will not cause any stability issues. However, if neighbouring octave bands in the attenuation filter differ dramatically in their respective gain, the error between the target reverberation time and the reverberation time of the feedback delay network will be increased. In addition to the aforementioned gain optimization, the difference between the

reverberation time computed in eq. 3.10 and eq. 3.13 needs to be minimized using linear least square methods [20]. Defining $\tilde{E}$ as the relative error, this can be computed as:

$$\tilde{E} = \left|\left| \frac{1}{A(\vec{\omega})} - \frac{1}{M\gamma(\vec{\omega})} \right|\right|_2^2 \tag{3.14}$$

In summary, it can be stated that the Feedback Delay Network introduces two additional components to the comb filter structure described by Schroeder and Logan in [29]. First, the feedback paths are extended by a matrix, which allows the parallel comb filters to introduce "crosstalk" to its adjacent paths. Moreover, the introduction of an attenuation filter allows for frequency-dependent reverberation times. Depending on the choice and complexity of the attenuation filter, the individual tuning of the impulse response in individual frequency bands can be more or less extensive. As suggested by Prawda et.al in [20], a graphic equalizer in the form of cascaded peak filters can be used as the attenuation filter. To overcome leaking between bands, an interaction matrix as explained in [40] constitutes the magnitude response of the attenuation filter.

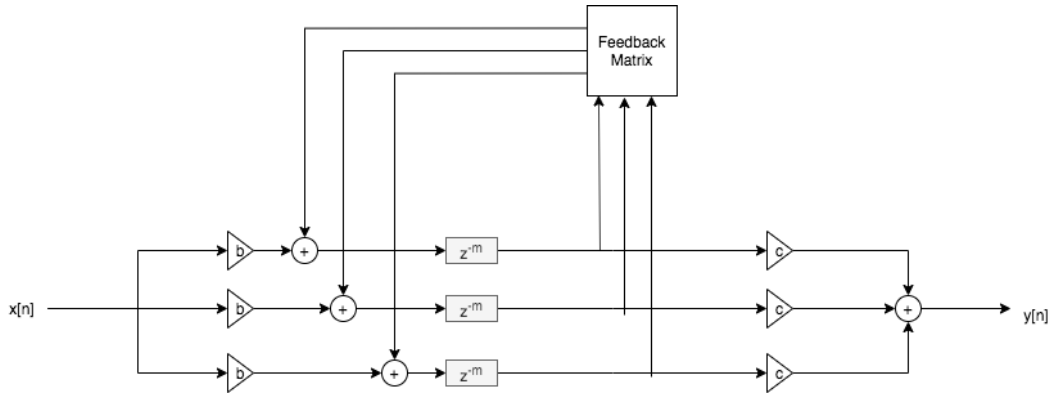


Figure 3.5: Feedback Delay Network Structure

3.4 image source Model

Although the image source Model was not directly implemented in this project, a lot of its concepts were modeled and referenced. Therefore, the theoretical background is covered in this section.

3.4.1 Terminology

As the image source Model pursues a different room approximation than the aforementioned Feedback Delay Network, some new terms and symbols are defined. This follows the original implementation by Allan in Berkley [1]:

- Room dimensions: l, w, h
- Listener location $\vec{r}$
- Source location $\vec{s}$
- Absorption coefficient $\alpha, 0 \leq \alpha \leq 1$
- Speed of sound $c = 343\frac{m}{s}$

3.4.2 Assumptions and Simplifications

At its most general level, the image source Model approximates a room impulse response with a time series of scaled and time-shifted impulses. Whenever a source emits a signal, its first order reflection can be modeled as a mirrored source at the respective boundary [1]. From there, second and higher order reflections can be computed recursively. Assuming that c is constant, the appropriate delay and volume of the impulses reaching the receiver can be computed.

To understand the implications of the image source Model on the room impulse response, the following assumptions have to be made:

- Sound propagates at straight lines (rays)
- Rays reflect perfectly at boundaries, i.e. no energy loss at boundaries due to nonlinear scattering is included in the calculation
- When a ray intersects a boundary, it spawns a secondary source at the mirrored position of the original emitter relative to the boundary

This implies, that any nonlinear effects such as scattering at boundaries are neglected by the image source Model.

3.4.3 Computation

First, the dirac delta function is given by [31]:

$$\delta(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

Given a single source at position p and a single ray that intersects a boundary at point t with normal vector n, the image source position p' can be calculated as given in [36]:

$$p' = p - 2n(n(p - t)) \tag{3.16}$$

For a single image source, the distance d to the receiver can be computed trivially:

$$d = \sqrt{(r_x - s_x)^2 + (r_y - s_y)^2 + (r_z - s_z)^2} \tag{3.17}$$

The delay τ introduced by the distance d between the source position $\vec{s}$ and receiver position $\vec{r}$ is given by [36]:

$$\tau = \frac{d}{c} \tag{3.18}$$

Therefore, the entire impulse response can be described by the sum of scaled and shifted impulses [5]:

Assuming that the boundary intercepted by ray w has an absorption coefficient α, the gain g of the impulse contributed by the image source is given by

$$g = \frac{1}{d}\alpha^{w} \tag{3.19}$$

Being able to compute g, d and τ for each image source, we can calculate the entire room impulse response as the sum of all individual impulses [5]:

$$h_{ISM}(n) = \sum_{i=1}^{I} g_i \delta(n - \tau_i) \tag{3.20}$$

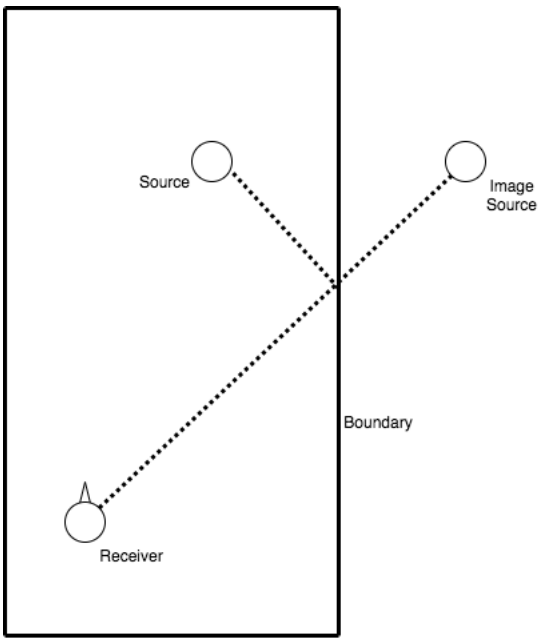


Figure 3.6: Calculation of image source positions for a single source, receiver, boundary and ray

3.4.4 Computational Cost

While it's not entirely obvious in eq. 3.20, this algorithm runs with exponential time complexity by its reflection order [5]. The approach of the model implies that for each order, each image source will have as many children as there are rays that intersect with a boundary. However, a series of validity and audibility checks are made, by which the number

of valid image sources is reduced [1]. Realtime implementations using multithreading techniques have been achieved and integrated into the Unity® game engine in [9]. While the image source Model is potentially more intuitive to understand and based on geometry, simple algorithmic reverberators such as the feedback delay network outperform it in terms of computational complexity. In recent years, attempts have been made to model the increasing pulse density inherent to the image source Model with feedback delay networks [27].

Chapter 4

Implementation

In this chapter, a rundown of all technical tools for the implementation of the reverberator are presented. As mentioned in the introduction, a plugin for parametrised room reverberation is the overarching objective of this project. Given that these have to happen in real-time, minimal and optimized computation at runtime are paramount. Therefore, a set of tools were chosen to help in this process. Furthermore, a timeline of failed and succeeded attempts and learnings are presented in a timeline-like order.

4.1 Technical Overview

4.1.1 Unity®

First released in 2005, the Unity® engine has become a popular application for creating interactive experiences [35]. It consists of a plethora of renderers, creation, build and scripting tools to prototype quickly. With its recent advancements, the engine aims to run efficiently on most architectures by parallelizing and vectorising machine code using proprietary compiler and memory management tools [35].

In this work, the most significant section of the engine is the `AudioMixer` module. With a structure that is similar to most digital audio workstations, audio emitters in the game hierachy can be connected to channels within the `AudioMixer` environment. Each channel provides volume control and an effect section, where plugins can be inserted. This is where the reverb is applied to the signal.

4.1.2 JUCE

Initially published in 2004, the JUCE framework aims to ease the creation of audio plugins and real-time audio applications [14]. It contains a comprehensive set of helper classes and tools, which both improve user interface design and algorithm design. In this context, it's been used to compile native C++ code into a proprietary plugin format to be read by Unity®.

4.1.3 Eigen

As described in Chapter 3, the solutions for a frequency dependent feedback delay network depends on solving equations using linear least squares methods. For ease of use, the C++ library Eigen has been deployed. It consists of a multitude of containers and supports, linear algebra operations, indexing operators and vectorisation of instructions [11]. As the creation of a sensible linear algebra library has been deemed as overscoped for this project, the Eigen library has proven to be a useful asset during implementation.

4.1.4 SOUL

As described later throughout this chapter, a Feedback Delay Network is a complex structure of filter elements. As a prototyping environment, the emerging domain-specific language SOUL has been utilized. With a c-like syntax, SOUL aims to ease the implementation of audio effects and abstract more complex aspects of low-level languages away without losing performance or precision [34]. As of now, a multitude of cross-compilers exist, e.g. to export SOUL code to C++. In this project, SOUL has been used to prototype peak filters, delay lines, lowpass filters and a primitive version of the Feedback Delay Network described in [13]. Example code can be found in the appendices.

4.2 Initial Implementation Plan

Recalling the objectives of this project described in chapter 1, the real-time implementation of a reverb with sensible parametrisation was defined of the main objective. With the utilization of multi-threading techniques and sensible separation of operations, the image-source model described in section 3.4 has already been implemented as a `.dll` solution for the Unity® Engine. However, in recent advancements, Unity® has put a considerable effort into parallelising operations, as for example described in the context of light rendering in [12]. Therefore, the initial premise was to employ Unity®'s native capabilities, such as its built-in raycast system to track image source positions and thus compute an impulse response based on room geometry. A prototype of the position tracker can be found below. With a number of 360 raycasts around one source object, the CPU profiler in Unity® did not show any significant performance spikes, so that from a performance standpoint, this implementation could potentially be done without any dynamic linking and explicit multi-threading instructions. However, a number of obstacles were introduced, among others the following:

- As described in section 3.4, the number of image sources grows exponentially by increasing order. Therefore, it is sensible to store each image source position in a tree-like data structure. As of now, this data structure is not yet supported by DOTS. Therefore, crow-barring the ISM into DOTS would likely make it less usable.
- The only way to store data using Unity® DOTS is sequentially. For every image-source model with a higher order, this storage paradigm would get inconvenient to work with.
- As of now, there is no way to write the elements of any buffer into a `.wav` file using Unity® DOTS. This is crucial if it is desired to later convolve signals with the

impulse response generated by the image-source model.

After consideration, the endeavour to implement the image source model natively in Unity® was abandoned. For this to succeed, it was concluded that the DOTS system would have to mature more. Nevertheless, the idea itself is worth pursuing. It could also be implemented as an offline tool, which calculates a static room impulse response from existing 3d geometry before run-time. Furthermore it lead to some simplifications and assumptions, which lead to the implementation proposed in the following section.

4.3 Early Reflections

Recalling section 3.4, the distance, delay and volume of a single image source can be calculated [5]. From there, secondary and higher-order image sources can be computed. However, following the paradigm in section 2.1 and the findings by Jot and Chaigne in [13], it can be assumed that due to the high pulse density in the early reflection phase of an impulse response, the exact temporal position of individual pulses will not be distinguishable. Therefore a filter structure similar to [29] is presented, containing the following elements:

1. A parallel path containing a single delay line element, a lowpass filter and a gain coefficient for each boundary. As the volume is dependent of a single impulse described in eq. 3.19, the lowpass filter acts as a simplified tone control parameter. Although absorption coefficients are usually given per frequency band and could potentially be controlled with a more sophisticated graphical equalizer, this would also increase the overall complexity.
2. All paths are summed and sent through 4 serialized allpass filters described in chapter 3. This is to compensate the pulse sparsity, which would otherwise be dependent on the number of the previously-described paths.

With this simple filter structure, it is both possible to control the delay introduced by the distance between the image source and the receiver as well as some basic tone control. The lowpass filter was implemented as proposed by Vadim Zavalishin in [44]. The allpass filter lengths were chosen as provided in [43] and evaluated to work well for small to medium-sized rooms. The individual delay line lengths are capped at 10000 samples.

4.4 Late Reverberation Tail

For the late reverberation tail, the feedback delay network proposed in [20] and explained in chapter 3 has been used. The attenuation filter for each path in the feedback delay network is comprised of a cascade of peak filters. Modifying the gain of the peak filters results in a parametrisable reverberation time for each specified frequency band. To be conform with the subsequent evaluation described in 5, the number of frequency bands is limited to six bands with $\vec{\omega} = \{125, 250, 500, 1000, 2000, 4000\} Hz$. It is important to know that an existing implementation by Willemsen and Prawda [20] has been utilized and modified to fit the needs of this project.
Similar to the approach taken in 4.3, the feedback delay network can be broken down into

individual processing blocks, namely peak filters, coefficients, delay lines and feedback connections. Therefore, these individual blocks were implemented as individual classes, which are then instantiated and reused. For the peak filter, the method proposed by Orphanidis in [25] was implemented. The complete structure of the reverberator, meaning the direct signal, the early reflections and the late reverberation tail can be found in figure 4.1

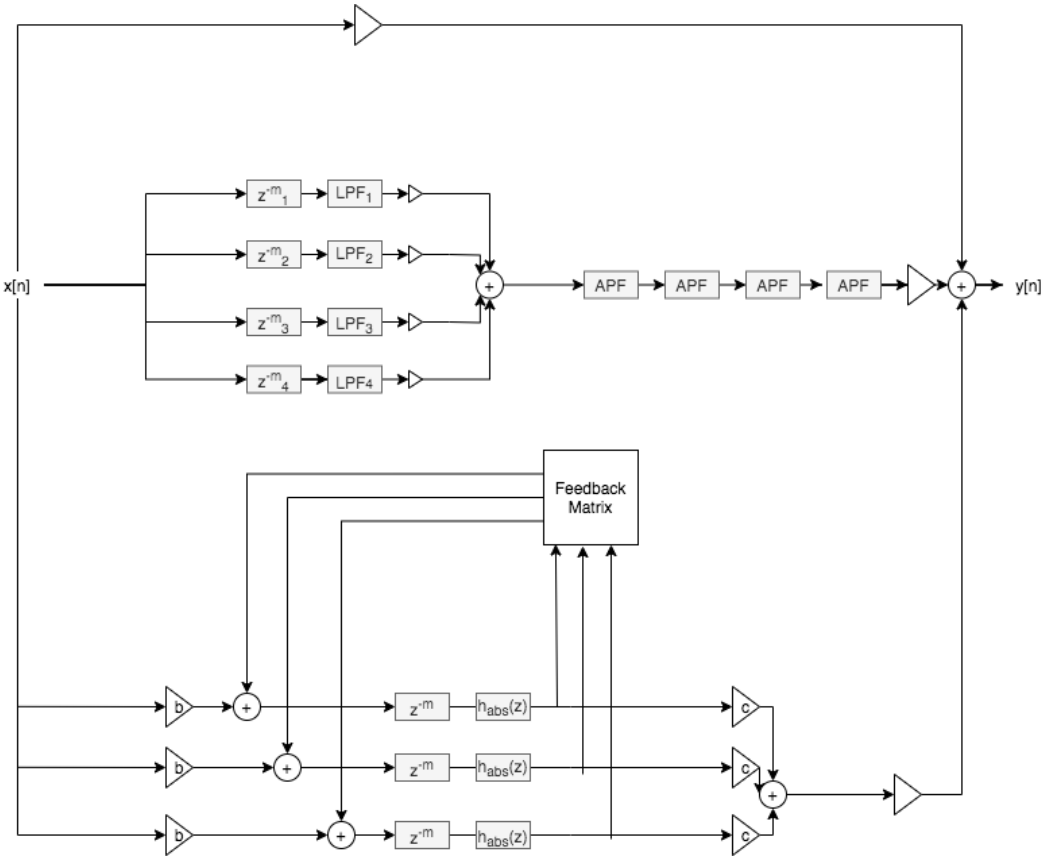


Figure 4.1: Complete Structure of the designed Reverberator

4.5 Parameter Automation in Unity®

Once compiled, the reverb plugin is ready to use in any Unity® environment. So far, the only parametrisable section of the reverb is the early reflection filter structure. To change the incoming early reflections, a `C#` script in Unity® was created which automates the delay and volume of the parameters during runtime. Additionally, the lowpass frequency for each path in the filter structure can be set before runtime to simulate different boundary materials and their absorptive behaviour.

The delay for each boundary follows the receiver position in relation to the boundary position. The farther the receiver is away from the audio source, the greater the delay will

be set. Similarly, the volume of a single reflection will decrease as the distance between the boundary and the receiver increases. In addition to the individual reflections, the sum of all early reflections are automated by the average distance of the receiver to the boundaries in a room. Furthermore, the volume of the late reverberation tail increases, as the distance between the receiver and the audio source increases, so that the direct signal of the audio source is decreased.

4.6 Technical Obstacles

Over the course of of the implementation phase of this project, a number of issues were encountered, which are explained in this section.

Recalculation of Late Reverberation Tail

The feedback delay network structure by Prawda et.al. suggests a cascaded peak filter structure as the attenuation filter for each path [20]. However, due to the potential leaking of adjacent frequency bands, the computation of the coefficients for the desired reverberation time per band relies on multiple matrix calculations per time step. To abide the objectives of this project given in chapter 1, namely real-time computation, these calculations are deemed to exceed the permitted runtime calculations. Broadly speaking, this wouldn't yield a significant disadvantage in terms of usability: In the context of an interactive application, the rooms are assumed to have a fixed size and therefore a constant reverberation tail. Unless a room needs to be resized during runtime, this trade-off is worth taking. Instead, the next best solution would be to calculate the reverberation time before the host application starts rendering any audio output. In the JUCE framework explained in section 4.1.2, there are three functions which execute sample calculations:

- `prepareToPlay()`: Executed before the host application starts rendering audio samples, e.g. when the user of a DAW presses on play.
- `processBlock()`: Executed during playback, in which an audio buffer is fetched, processed, and returned to the respective audio device.
- `releaseResources()`: Executed after the application stops rendering audio samples, e.g. when the user of a DAW presses stop.

The issue of this procedure lies in the fact, that it is the host's responsibility to call these functions at the correct point in time and has been described in numerous threads within the developer community [21]. In the case of Unity® being the host application, `preparetoPlay()` does not get called. In consequence, the desired reverberation times have to be known before compiling the plugin and for each desired room size, a complementary version of the plugin has to be exported. In the scope of the evaluation of this project, this did not pose a problem, yet in large-scale game applications, this would not presumably diminish the plugin's user-friendliness.

Frequency instabilities

Despite the measures described in chapter 3, the feedback delay network became instable at certain frequencies after long periods of running (> 1 minute). After extensive debugging

and error-checking, no apparent reason for this has become evident. Therefore, the gain coefficient of the feedback delay network out put was capped at 75 %. Furthermore, a 1st order Lowpass filter as described in [44] was inserted after the feedback delay network was inserted, whose cutoff frequency $f_c = 10000Hz$ was determined experimentally.

Parameter Scaling

During implementation, it was found out that despite the averaging of the distances between the wall and the receiver, significant jumps in the volume of the early reflection's volume occurred. The same behaviour happened for the volume of the late reverberation tail. Therefore, the actual values which drive the parameters of the plugin were scaled by values, which were found experimentally, to keep it within a the allowable range.

4.7 Summary

Concluding the implementation done in this work, the core components consist of the plugin developed in JUCE using `C++` as well as a script which automates its parameters. The reverberation times of each frequency band for the late reverberation tail remains constant during runtime, only the overall volume of the tail is automated. The early reflections are modeled as 4 parallel delay line paths with a lowpass filter and a gain coefficient. Based on the Image-Source Model explained in section 3.4, it is assumed that all impulses after the first order can be modeled by a series of allpass filters. With these simplifications at hand, the reverb can run in real-time and the early reflections are updated based on the listener position relative to the audio source and the room's boundaries.

Chapter 5

Evaluation

To classify the perceptual qualities and accuracy of the reverberator described in section 4, a 2-fold evaluation is performed. 2 freely available extensions for the Unity® engine, which achieve objectives similar to the ones in this report, are compared with the reverberator of this project. In this chapter, the frameworks, the test setup and the environment are presented.

5.1 Technical Overview

Analysing the reverb in this project in terms of its capability to make parameter changes during playback, it can be stated that the reverberation tail is fixed, yet the early reflections can update in accordance with the relative position between the receiver, source, and room boundary positions. Having this set of capabilities at hand, it is sensible to compare the reverberator to 2 presumably more accurate and sophisticated reverbs. It is important to note that while these frameworks are well-documented [33][23], the actual technical and computational schemes behind these reverberators are only partly or not published. Therefore, the descriptions of these frameworks are a combination of paper analysis and inferences that can be made based on their documentation.

5.1.1 Google Resonance

First published in 2018, Resonance Audio is a software development kit which extends existing game engines, audio engines and digital audio workstations [23]. Its features are comprised of a binaural renderer as described in [10] as well as 2 different reverberators. In Unity®, the binauralization works by attaching the scripts `ResonanceAudioListener` and `ResonanceAudioSource` to the respective objects within the virtual environment. Simple algorithmic reverberation is achieved by attaching the script `ResonanceAudioRoom` to an object. From there, a set of parameters such as the reverberation time as well as perceptual super-parameters such as "Brightness" can be tuned. Furthermore, a geometry-based reverb tail can be calculated by using the scripts `ResonanceAudioReverbProbe` and `ResonanceAudioAcousticMesh`. The reverberation tail is then computed before the scene can be played.

In this project, the reverberator developed in this project was compared to the algorithmic reverb of the Google Resonance SDK.

5.1.2 Steam Audio

Initially released in 2017, the Steam Audio Library augments the Unity® engine in a similar fashion as Google Resonance. It consists of a custom binaural renderer and a geometry-based reverberator. Within a Unity® environment, the binaural renderer is enabled by attaching the scripts `SteamAudioListener` and `SteamAudioSource` to the respective objects in the scene. Furthermore, any number of boundaries (e.g. walls, ceilings, floors, doors, windows) can be tagged with a `SteamAudioMaterial` script. The component `SteamAudioProbeBox` acts as a virtual microphone, from which the room impulse response is computed. The number of microphones in the given space can be customized, which increases the resolution at the cost of increased computation. The reverb is then precomputed ("baked") before run-time. Whether or not the system relies on a deterministic model such as the image-source model described in section 3.4 or on a stochastic model, is not known. Yet, as the reverberator computes all signals offline, it can be considered as the most sophisticated model of all addressed in this work.

5.1.3 Test Environment

Recalling the motivation and main objective of this project described in section 1.1, a sensible method to evaluate the reverberator is within an interactive game environment, which is explained as follows:

For test conduction, a simple scene in Unity® was constructed, consisting of a rectangular room with a loudspeaker placed inside it. Furthermore, a simple camera controller enabling a 1st person perspective and an `AudioListener` component was imported. The controller can be controlled using the `WASD` keys on any conventional keyboard. An `AudioSource` component was attached to the loudspeaker, which plays back a small sample of a female voice, which was trimmed to conform to the conventions proposed in the MUSHRA test setup [16]. Although a MUSHRA test was not performed in this work, its recommendations can be considered valuable in other test scenarios and was therefore utilized. The room's boundaries were completed with a `material` component which resembles wood.

With the sizes and dimensions of the room's boundaries as well as the given material in place, the individual reverberation times were calculated per band according to eq. 2.1 with absorption coefficients given by [39] and given as follows:

Frequency (Hz)	**Reverberation Time (seconds)**
125	1,26
250	1,60
500	2,07
1000	3,91
2000	3,52
4000	3,19

, resulting in an average reverberation time of 2.59 seconds.

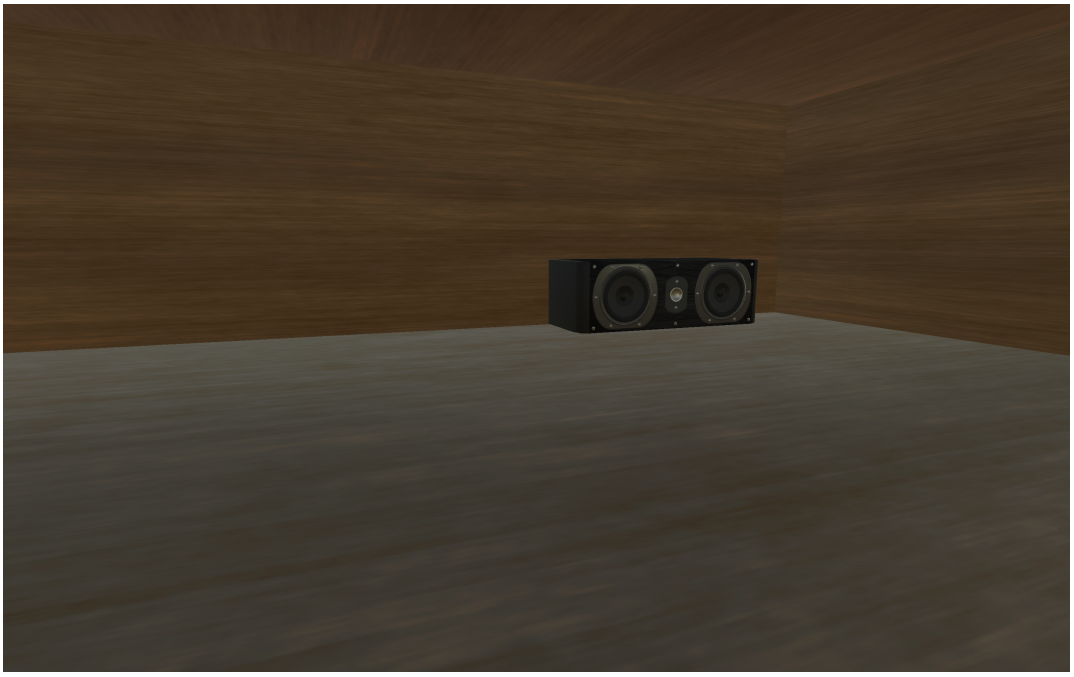


Figure 5.1: 1st person view of the virtual test environment

With this information at hand, the self-developed reverberator was tuned such that the individual reverberation times are met numerically.

The reverberator by Google Resonance does not have such a precise parametrisation, such that individual frequency bands can be tuned individually. Instead, the reverb assumes that only rectangular rooms with 6 surfaces are used. The surface materials can be chosen from a pre-defined list, in this case, the setting "Interior Wood" seemed the most fitting. Moreover, the parameters "Reflectivity", "Gain", "Brightness" and "Time" serve to tune the reverberator. The first 3 were determined experimentally, while the parameter "Time" was set to the average of the calculated reverberation time. Whether or not this value is actually mapped to the T_{60} value described in chapter 2.1 is not apparent.

The reverberator by Steam Audio has a similar black-box like behaviour. As described above, the reverb computes by analysing the geometry of the room. Scene objects that represent room boundaries are tagged with a certain script, by which the framework computes the reverb parameters are calculated offline. Similarly, material presets can be chosen from a pre-defined list. Again, choosing the material preset "Wood" was found to be the most semantically sensible in the scope of this evaluation.

5.2 Numerical Evaluation

To have a more tangible measure for comparison, the impulse responses of all 3 systems were taken and compared. The impulse signal was generated from Matlab®. Recalling Vorländers definition of the 2 aspects of auralization described in section 2, local and spatial impression respectively, it was considered useful to separate these 2 components

as much as possible. Therefore the binauralization settings in all 3 engines was set to minimum, such that, at best, no HRIR convolution would take place and thus colour the results. The resulting impulse responses were then recorded from Unity to a `.wav` file at a sampling rate of 48000 Hz. Subsequently, the impulse responses were normalized to unity gain to eliminate potential attenuation caused in the recording process. The spectrograms shown below were computed in Matlab® with a hop-size of 2048 samples.

5.3 Subjective Evaluation

Recalling the parametrisation of the reverberator described in chapter 4, the reverberator encompasses the features of setting a frequency-dependent reverberation time before run-time, early reflections updating during run-time and dynamic mixing of direct ("Dry") and reverberant ("Wet") signal parts based on source and listener positions. Moreover, the initial objectives stated in section 1.1 states the use of the reverberator within interactive environments. Therefore, the following criteria for a sensible comparison and evaluation, the following categories were defined before designing the subjective evaluation:

1. Visual and auditory congruence: The perceived size of the room matches with the perceived reverberation time.
2. Distance attenuation: The overall volume of the signal matches with the distance to the sound source.
3. Dynamic mixing of direct and reverberant signal: The mixing between direct and reverberant signal matches with the distance to the sound source.
4. Dynamic mixing of early reflections and late reverberation tail volume: The mixing between early reflections and late reverberation tail changes in a perceptually plausible manner.

These four categories should then be evaluated in three conditions, with reverb being generated from the designed reverberator, Google Resonance's reverb and Steam Audio's pre-baked, geometry-based reverb. It was chosen to evaluate these statements on a Likert scale within a range from 1 ("Strongly disagree") to 7 ("Strongly Agree").

5.3.1 Test Setup

Due to the start of the Covid-19 pandemic in March 2020, it was decided to construct a test setup that allows users to participate remotely. Therefore, the same virtual environment was exported three times, each having a different reverb engine running. The order of evaluation was then randomized, resulting in 6 different possible orders. Along with clear test instructions to ensure the user was wearing headphones, these packages were sent out manually to the participants. By giving each participant a unique identifier, the ratings could be reconstructed correctly.

Chapter 6

Results and Discussion

6.1 Numerical Results

As it is trivial to see throughout the spectrograms in figures 6.3, **??** and **??**, all reverberators fail to meet the specified average reverberation time of 2.59 seconds. In terms of frequency decay, it can be seen that the reverberator by Google Resonance introduces a steep frequency cut at 11 kHz and decays very abruptly. Similarly, the reverberator by Steam Audio has a much more equally-distributed frequency spectrum, yet decays abruptly. The reverberator developed in this project has a clear exponential decay and the frequency-dependent reverberation times are visible as horizontal lines in the power spectrum.
In the time domain, the difference in decay over time is evident. While the designed reverberator has many distinctly visible echos, Steam Audio's reverb has a much smoother tail. Google Resonance's reverb seems to decay faster than the specified reverberation time promises. To have a single measure for the similarity between the individual impulse responses, the pearson correlation coefficient was computed for the time domain signals, resulting in the following values:

Reverb 1	Reverb 2	r
Custom	Google Resonance	0.42
Custom	Steam Audio	0.17
Google Resonance	Steam Audio	0.36

Table 6.1: Correlation Coefficients between the impulse responses of the reverberators at stake

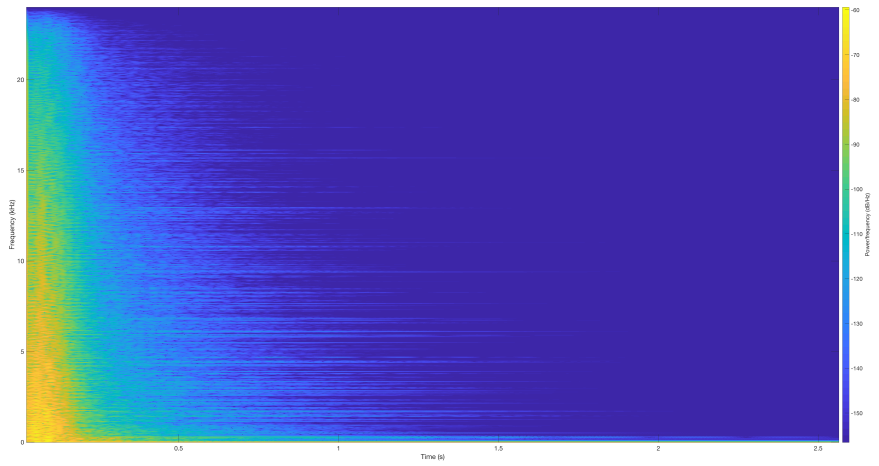


Figure 6.1: Spectrogram of impulse response of the designed reverberator

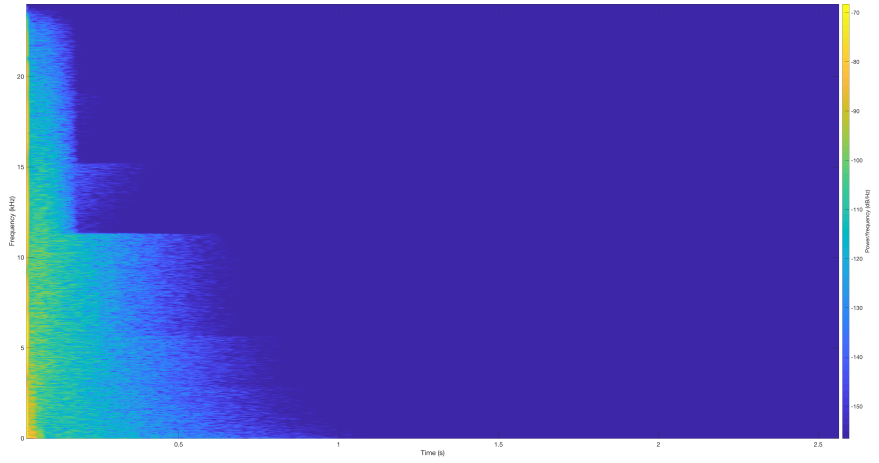


Figure 6.2: Spectrogram of impulse response of Google Resonance's reverberator

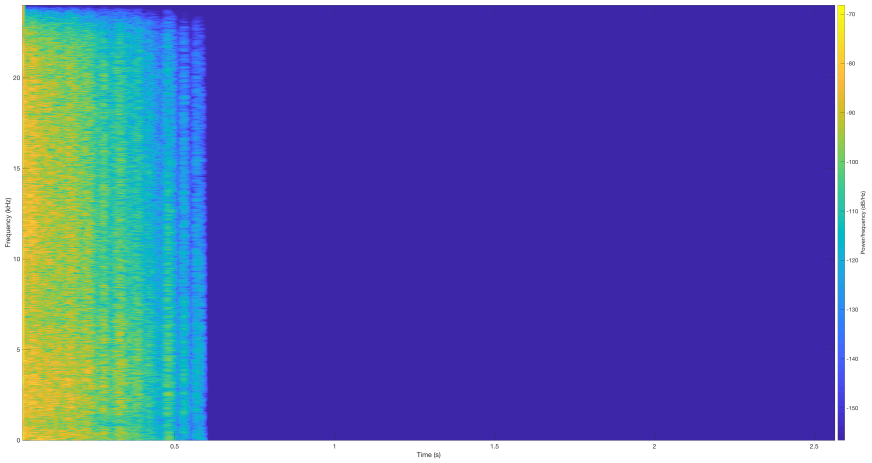


Figure 6.3: Spectrogram of impulse response of Steam Audio's reverberator

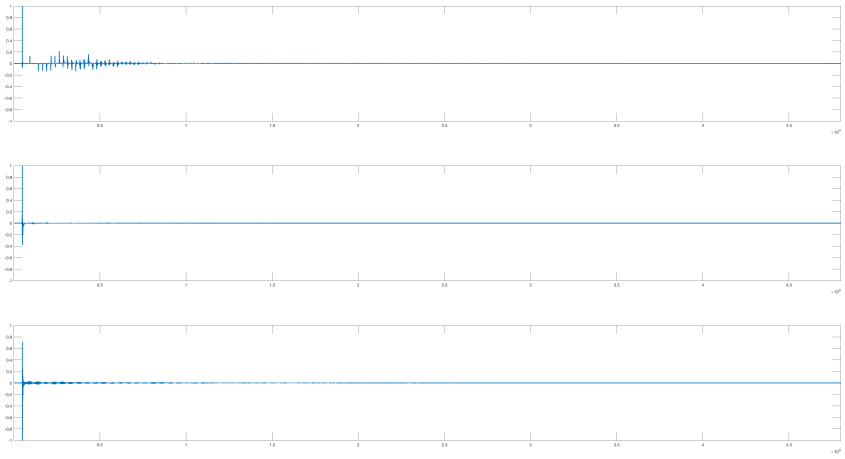


Figure 6.4: Time domain plots for the individual reverberators Top: Custom, Middle: Google Resonance, Bottom: Steam Audio

6.2 Listening Tests Results

With the given design of the questionnaire explained in section 5.3, it is likely that the data does not follow a normal distribution due to the low resolution of the rating scales. Therefore, common analysis schemes such as student's t-test or the analysis of variance would not yield valid or insightful results. It was decided to perform a Mann-Whitney U test as this assumes that the data is not normally distributed [15] and accounts for a low resolution of rating possibilities.

Reconsidering the assumptions made about the different reverberators, the hypotheses for the individual comparisons can be formulated as follows:

1. There is no significant difference across all categories in question between the means of the ratings for the designed reverb and Google Resonance's reverb.
2. There is no significant difference across all categories in question between the means of the ratings for the designed reverb and Steam Audio's reverb.
3. There is no significant difference across all categories in question between the means of the ratings for Google Resonance's reverb and Steam Audio's reverb.

The pre-determined level for a significant difference was defined as $\alpha = 0.05$ as commonly used in statistical analysis.

	Custom Reverb	Google Resonance	Steam Audio
Q1	$m = 5.2, s = 1.5$	$m = 4.8, s = 1.9$	$m = 3.9, s = 1.4$
Q2	$m = 5.0, s = 1.5$	$m = 4.7, s = 1.9$	$m = 5.0, s = 1.8$
Q3	$m = 4.8, s = 1.6$	$m = 4.5, s = 1.8$	$m = 5.0, s = 1.2$
Q4	$m = 5.0, s = 1.6$	$m = 4.9, s = 1.5$	$m = 4.7, s = 1.5$

Table 6.2: Means and standard deviations for each reverb condition and each category observed

6.2.1 Participant Data

In total, 14 persons participated in the listening tests, of which 11 were male and 3 female. No participants reported any hearing deficiencies, therefore no participant was discarded for further analysis. The mean age at the time of data collection of the participant group was 30 with a standard deviation of 5.36. No technical problems were reported in the questionnaires.

6.2.2 Comparison: Custom Reverb vs. Google Resonance

With regards to the visual and auditory congruence of the 2 reverberators, there was no significant difference in the ratings, with the U-test yielding $p_{q1\ custom\ vs\ gr} = 0.70$.

Similarly, there was no significant difference for the plausibility of the perceived distance attenuation, resulting in $p_{q2\ custom\ vs\ gr} = 0.90$.

In terms of the dynamic mixing of the direct and reverberant signal, there was no significant difference according to the ratings with $p_{q3\ custom\ vs\ gr} = 0.85$.

Lastly, there was no significant difference in the perceived dynamic mixing between the early reflections and late reverberation tail with $p_{q4\ custom\ vs\ gr} = 0.70$.

6.2.3 Comparison: Custom Reverb vs. Steam Audio

With respect to the match between visual and auditory stimuli, it can be stated that there is a significant difference between the custom reverb and Steam Audio's reverb with a probability of $p_{q1\ custom\ vs\ sa} = 0.02$. The hypothesis stating that there is no significant difference in this category is therefore rejected.

As for the perceived distance attenuation, there was no reported significant difference with $p_{q2\ custom\ vs\ sa} = 0.91$.

Regarding the dynamic mixing between the direct and reverberant signal, no significant difference was found with $p_{q2\ custom\ vs\ sa} = 0.85$.

Lastly, no significant difference was found between the perceived mixing of early reflections and late reverberation tail with $p_{q4\ custom\ vs\ sa} = 0.39$.

6.2.4 Comparison: Google Resonance vs. Steam Audio

For completeness, the 2 referenced frameworks are compared as well.

Regarding the visual and auditory congruence, the Mann-Whitney test does not yield a significant difference, with $p_{q1\ gr\ vs\ sa} = 0.14$.

Similarly, the distance attenuation of both frameworks are reportedly not significantly different, resulting in $p_{q2\ gr\ vs\ sa} = 0.65$.

With respect to the perceived mixing between direct and reverberant signal, no significant difference is found with $p_{q3\ gr\ vs\ sa} = 0.42$.

Lastly, no significant difference was found between the perceived mixing of early reflections and late reverberation tail with $p_{q4\ gr\ vs\ sa} = 0.84$.

Reverberators	Q1	Q2	Q3	Q4
Custom vs Google Resonance	0.71	0.71	0.74	0.70
Custom vs Steam Audio	**0.02**	0.91	0.85	0.39
Google Resonance vs Steam Audio	0.14	0.66	0.42	0.85

Table 6.3: Means and standard deviations for each reverb condition and each category observed

In summary, no significant difference across all categories has been reported with the exception of the visual and auditory congruence between the custom reverb and Steam Audio's pre-baked reverb. As seen in figure 6.5, the the custom reverb was evaluated to provide a more consistent spatial impression of the described room. Apart from that, the means of the rating do not differ significantly.

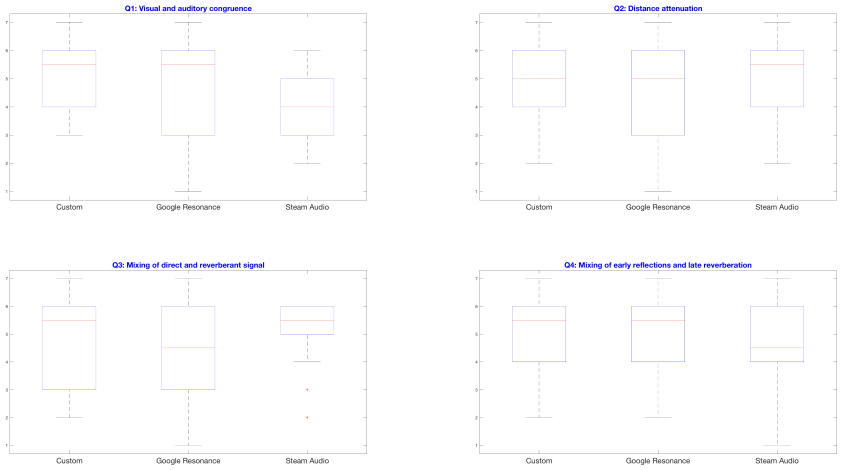


Figure 6.5: Boxplots showing the ratings of all reverberators for all questions

6.3 Discussion

Having the results at hand, these should be reviewed critically given the circumstances of the test setup and the validity of the comparison itself. First and foremost, it is important to note that numerically, the differences between the impulse responses are quite big, with a maximum correlation coefficient of 0.42. The listening test results however show that participants did not hear a difference that was big enough to reflect a significant difference after running a Mann-Whitney U test, with one exception. Having that discrepancy at hand, there are a multitude of reasons why the numerically different impulse responses are only partially reflected in the listening tests results.

Recalling the topology and objective of the reverb designed in this work, it computes both the early reflections and the late reverberation tail in different filter structures. Judging by the short length of their impulse responses, the other reverb implementations are assumed to have a vastly different implementation scheme. Especially Steam Audio's reverb seems to omit a late diffuse tail and merely cuts the signal at a given threshold. This issue is supported by informal conversations of other users of the plugins and can be found in appendix section A.1. Bearing that in mind, it can be stated that the comparison between these systems was not entirely applicable, as at least one of the compared reverb implementations fulfils a late reverberation tail only to a small degree.

Moreover, the capturing process of the impulse responses has to be addressed. Recalling the parameters and settings of both Google resonance and Steam Audio in section 5.1.1 and section 5.1.2 respectively, it can be stated both systems have different parametrisation capabilities than the feedback delay network. Steam Audio's impulse response is calculated in processes, which are not transparent and judging by the power spectrum representation, it is possible that it neglects audio samples below a certain threshold. As an improvement, it is therefore suggested to choose more similar projects for future comparisons, such as Evertims by Poirot et.al [19].

With regard to the listening test conduction, there are a number of issues at hand which need to be addressed. An immediate problem is the comparatively small sample size. In the test setup, an environment with high ecological validity was introduced. As the user is subject to multiple stimuli with rich information, the probability that other variables have influenced the decision-making process while answering the questionnaire can not be eliminated. For more reliable results, a perceptually sparse environment and a higher number of participants is suggested.

Secondly, the conditions of the user tests were vastly different for each participant and thus far from an ideal controlled testing environments. Due to the outbreak of the Covid-19 pandemic in March 2020, the tests had to be conducted remotely across different machines. Moreover, the listening environments and other technical factors such as the choice of headphones were different for each participant. Although there were no technical difficulties encountered according to self-reports, one pre-tested computer with one pair of headphones would potentially decrease the standard deviation, abide to scientific test conduction conventions and yield more reliable results.

Moreover, the overarching construct of auralization should be reconsidered. Recalling Vorländer's definition of spatial and local impression of auralized signals [39], the focus of this project was entirely on the spatial impression. The approach of room reverberation in this project does not consider the binaural aspect of auralization, i.e. identifying the distinct position of a sound source. It can therefore be argued, that instead of separating these two components, auralization should always be evaluated as a combination of these two. Thus, the binaural aspect should be considered more heavily in future endeavours.

Furthermore, it is important to review the outcome of this project by relating it to its predefined objectives defined in section 1.1, i.e. creating a perceptual plausible reverb for real-time usage which takes acoustic parameters into account and leaves some space for game developers to tune the reverberator to their needs.

The perceptual plausibility in this work was defined by evaluating 4 different aspects of reverberation, namely the auditory and visual congruence, the distance attenuation, the mixing between direct and reverberant signal and the mixing between early reflections and late reverberation tail. According to the listening tests, these criteria are met and no anomalies are reported. During development, it was found out that the late reverberation tail is still unstable at certain frequencies. So it can be stated that the reverb meets this requirement not completely, but at least partially.

Moreover, a set of acoustic parameters are considered, at least for rectangular room and the reverberation time can be set for individual frequency bands. Due to a set of technical errors explained in section 4, the reverberation time needs to be calculated before the plugin is compiled. Assuming that room sizes are not subject to change during engine playback, this trade-off was considered to be appropriate. With this functionality, the reverb is not in an ideal state, yet it fulfils the requirement at least partially.

Further parameter tuning can be achieved in the early reflections section of the reverb. As of now, the early reflections are modeled as individual delay lines with gain coefficients and a lowpass filter for modeling material absorption. For further improvements, it is suggested to replace the lowpass filter with a more sophisticated filter structure, similar to the graphic equalizer of the feedback delay network. Taking these into consideration, it can be stated that the reverb fulfils the requirements partially.

Chapter 7

Conclusion

The objectives of this report were to design a perceptually plausible reverberator for interactive virtual environments, which incorporate dynamic changing of its characteristics at run-time, accounts for room geometry and leaves some space for design parameters.

Summing up the background theory, the implementation and evaluation of the reverberator, it can be stated that these objectives have been partially fulfilled. The parameters of the early reflections, the mix between early reflections and the overall mix between direct and reverberant signal are updated based on receiver positions. A first order lowpass-filter allows for rudimentary frequency tuning of the early reflections. For rectangular rooms, the late reverberation tail needs to be pre-computed, yet a frequency-dependent reverberation time can be achieved, based on a set of absorption coefficients. Due to the computational cost for this calculation, this cannot be updated during runtime. However, a dynamic reverberation tail would mean that rooms are resized during run-time. This seems a reasonable trade-off given the scope of the project.

Subsequent comparisons in a single room setup with other commercially available reverbs indicate that there is a vast difference between the impulse responses of the systems, with a correlation coefficient of < 0.5. Listening tests enquiring about the perceptual plausibility of the reverb show that there is no significant difference perceptually when compared to two other reverberators apart from one exception. With this discrepancy, immediate improvements to the project would include a more exhaustive comparison to other, more similar reverberation systems, testing in a perceptually sparse environment, and conducting the listening tests in a controlled environment.

Thus far, the designed reverb incorporates only linear effects of room reverberation. Therefore, further immediate improvements to the system include the incorporation of diffraction and shadowing effects caused by room geometry. Furthermore, the reverb is only applicable to rectangular rooms. With these shortcomings, the perceptual qualities of the reverb seem plausible, yet it lacks some ease of use. However, as the objective of this project is to create a reverb that can function in real-time contexts, the lack of nonlinear effects seems appropriate. Described in [27] and [28], the modeling of nonlinear reverberation effects in feedback delay networks seems applicable. Therefore, it can be stated that the objectives defined in this report are partially met and it is hoped that this project lays a proper foundation for the implementation of reverbs with sensible parametrisation in the context of real-time environments.

Bibliography

[1] Jont B. Allen and David A. Berkley. "Image method for efficiently simulating small-room acoustics". In: *The Journal of the Acoustical Society of America* 65.4 (1979), pp. 943–950. DOI: 10.1121/1.382599.

[2] Mike Barron. "The subjective effects of first reflections in concert halls—The need for lateral reflections". In: *Journal of Sound and Vibration* 15 (Apr. 1971), pp. 475–494. DOI: 10.1016/0022-460X(71)90406-8.

[3] Jens Blauert. "Sound Localization in the Median Plane". In: *Acta Acustica united with Acustica* 22 (Nov. 1969).

[4] Jens Blauert. "Spatial hearing: the psychophysics of human sound localization". In: Jan. 2001.

[5] Fabian Brinkmann, Vera Erbes, and Stefan Weinzierl. "Extending the closed form image source model for source directivity". In: Mar. 2018.

[6] Fabian Brinkmann and Stefan Weinzierl. "AKtools - the open software toolbox for signal acquisition, processing, and inspection in acoustics." In: (Feb. 2017).

[7] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[8] William Gardner. "Reverberation Algorithms". In: Apr. 2006, pp. 85–131. DOI: 10.1007/0-306-47042-X_3.

[9] Konstantinos Gkanos. *Interactive, real-time room acoustics simulation*. English. 2019.

[10] Marcin Gorzel et al. "Efficient Encoding and Decoding of Binaural Sound with Resonance Audio". In: Mar. 2019.

[11] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[12] Eric Heitz, Stephen Hill, and Morgan McGuire. "Combining analytic direct illumination and stochastic shadows". In: May 2018, pp. 1–11. ISBN: 978-1-4503-5705-0. DOI: 10.1145/3190834.3190852.

[13] Jean-Marc Jot and Antoine Chaigne. "Digital delay networks for designing artificial reverberators". In: (Jan. 1991).

[14] *JUCE | JUCE*. en. Library Catalog: juce.com. URL: https://juce.com/ (visited on 03/27/2020).

[15] H. B. Mann and D. R. Whitney. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". In: *The Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60. ISSN: 00034851. URL: http://www.jstor.org/stable/2236101.

[16] Catarina Mendonça and Symeon Delikaris-Manias. "Statistical tests with MUSHRA data". In: May 2018.

[17] Henrik Møller. "Fundamentals of binaural technology". In: *Applied Acoustics* 36.3 (1992), pp. 171 –218. ISSN: 0003-682X. DOI: https://doi.org/10.1016/0003-682X(92)90046-U. URL: http://www.sciencedirect.com/science/article/pii/0003682X9290046U.

[18] Natasa Paterson et al. "Spatial audio and reverberation in an augmented reality game sound design". In: *Proceedings of the AES International Conference* (Jan. 2011).

[19] David Poirier-Quinot, Markus Noisternig, and Brian F. G. Katz. "EVERTims: Open source framework for real-time auralization in VR". In: *Audio Mostly Conference*. 2017.

[20] Karolina Prawda, Vesa Välimäki, and Sebastian Schlecht. "Improved Reverberation Time Control For Feedback Delay Networks". In: Sept. 2019.

[21] *prepareToPlay and releaseResources are not getting called?* JUCE. Aug. 15, 2010. URL: https://forum.juce.com/t/preparetoplay-and-releaseresources-are-not-getting-called/5562 (visited on 05/21/2020).

[22] Nikunj Raghuvanshi and John Snyder. "Parametric directional coding for precomputed sound propagation". In: *ACM Transactions on Graphics* 37.4 (July 30, 2018), pp. 1–14. ISSN: 07300301. DOI: 10.1145/3197517.3201339. URL: http://dl.acm.org/citation.cfm?doid=3197517.3201339 (visited on 10/17/2019).

[23] *Resonance Audio*. URL: https://resonance-audio.github.io/resonance-audio/ (visited on 05/24/2020).

[24] Jens Rindel. *Book Review: Room Acoustics, sixth edition, by Heinrich Kuttruff*. Aug. 2017.

[25] Orphanidis S. "Introduction to signal processing, by S.J. Orfanidis, Prentice Hall Signal Processing Series, Prentice Hall, Upper Saddle River, New Jersey, 1996 - Book review". In: *Control Engineering Practice* 4 (Dec. 1996). DOI: 10.1016/S0967-0661(96)90009-X.

[26] Sebastian Schlecht and Emanuël Habets. "Dense Reverberation With Delay Feedback Matrices". In: Oct. 2019.

[27] Sebastian Schlecht and Emanuël Habets. "Dense Reverberation With Delay Feedback Matrices". In: Oct. 2019.

[28] Sebastian Schlecht and Emanuël Habets. "Scattering in Feedback Delay Networks". In: (Dec. 2019).

[29] Manfred R. Schroeder and Benjamin F. Logan. "'Colorless' Artificial Reverberation". In: *J. Audio Eng. Soc* 9.3 (1961), pp. 192–197. URL: http://www.aes.org/e-lib/browse.cfm?elib=465.

[30] Dirk Schröder and Michael Vorlaender. "RAVEN: A real-time framework for the Auralization of interactive virtual environments". In: *Proceedings of Forum Acusticum* (Jan. 2011), pp. 1541–1546.

[31] Julius O. Smith. *Physical Audio Signal Processing*. http://ccrma.stanford.edu/~jos/pasp/.

[32] Simone Spagnol. "Auditory Model Based Subsetting of Head-Related Transfer Function Datasets". In: May 2020, pp. 391–395. DOI: 10.1109/ICASSP40776.2020.9053360.

[33] *Steam Audio*. URL: https://valvesoftware.github.io/steam-audio/ (visited on 05/24/2020).

[34] Julian Storer. *SOUL - The Future of Audio Coding*. URL: https://soul.dev (visited on 05/19/2020).

[35] Unity Technologies. *Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations*. en. Library Catalog: unity.com. URL: https://unity.com/ (visited on 03/20/2020).

[36] Matthew Reuben Thomas. "Wayverb: A Graphical Tool for Hybrid Room Acoustics Simulation". PhD thesis. University of Huddersfield, 2017.

[37] N. Toma, M. Topa, and E. Szopos. "Aspects of reverberation algorithms". In: *International Symposium on Signals, Circuits and Systems, 2005. ISSCS 2005.* Vol. 2. 2005, 577–580 Vol. 2. DOI: 10.1109/ISSCS.2005.1511306.

[38] Michael VORLÄNDER. "Virtual Acoustics". In: *Archives of Acoustics* 39.3 (2014).

[39] Michael Vorländer. *Auralization: fundamentals of acoustics, modelling, simulation, algorithms and acoustic virtual reality*. 1st ed. OCLC: ocn171111969. Berlin: Springer, 2008. 335 pp. ISBN: 978-3-540-48829-3.

[40] Vesa Välimäki and Juho Liski. "Accurate Cascade Graphic Equalizer". In: *IEEE Signal Processing Letters* PP (Dec. 2016), pp. 1–1. DOI: 10.1109/LSP.2016.2645280.

[41] Vesa Välimäki et al. “More Than 50 Years of Artificial Reverberation”. In: Feb. 2016.

[42] Florian Völk. “System Theory of Binaural Synthesis”. In: *Audio Engineering Society Convention 131*. 2011. URL: http://www.aes.org/e-lib/browse.cfm?elib=16093.

[43] Bruce Wiggins and Mark Dring. “AmbiFreeVerb 2—Development of a 3D Ambisonic Reverb with Spatial Warping and Variable Scattering”. In: July 2016.

[44] Vadim Zavalishin. *The art of VA filter design*. URL: https://www.native-instruments.com/fileadmin/ni_media/downloads/pdf/VAFilterDesign_1.1.1.pdf (visited on 05/20/2020).

Appendix A

Appendix

A.1 Conversations about Steam Audio Impulse Responses

Jesper Udesen

Steam Audio plugin in Unity

To: Daniel Ströbig, Cc: Stefania Serafin

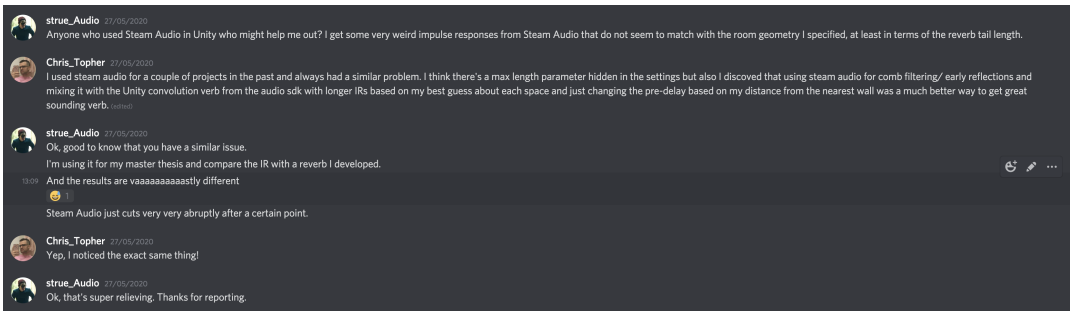
28. May 2020 at 16:11

Details

Hi Daniel

As promised, I checked the Steam audio plugin in Unity and I think the reverberations looks a bit non-physical when I look at the room impulse responses but it is difficult to say for sure. I have attached a figure with the left and right ear responses from a speaker in a room to each ear of the player (the HRTF's are also convolved). The wall material is "rock" and the room size is ~10mx10mx10m and the distance to the speaker is ~5m

BTW: when I listen to the corresponding audio (not the MLS noise but some music in the virtual room) it sound OK and realistic.



strue_Audio 27/05/2020
Anyone who used Steam Audio in Unity who might help me out? I get some very weird impulse responses from Steam Audio that do not seem to match with the room geometry I specified, at least in terms of the reverb tail length.

Chris_Topher 27/05/2020
I used steam audio for a couple of projects in the past and always had a similar problem. I think there's a max length parameter hidden in the settings but also I discoved that using steam audio for comb filtering/ early reflections and mixing it with the Unity convolution verb from the audio sdk with longer IRs based on my best guess about each space and just changing the pre-delay based on my distance from the nearest wall was a much better way to get great sounding verb. (edited)

strue_Audio 27/05/2020
Ok, good to know that you have a similar issue.
I'm using it for my master thesis and compare the IR with a reverb I developed.
13:09 And the results are vaaaaaaaaastly different
1
Steam Audio just cuts very very abruptly after a certain point.

Chris_Topher 27/05/2020
Yep, I noticed the exact same thing!

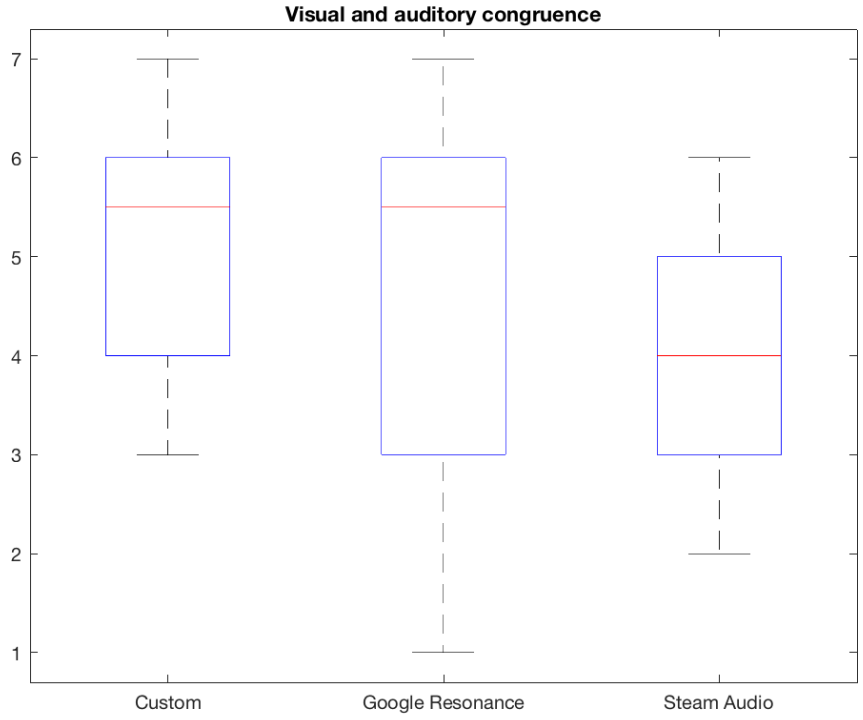
strue_Audio 27/05/2020
Ok, that's super relieving. Thanks for reporting.

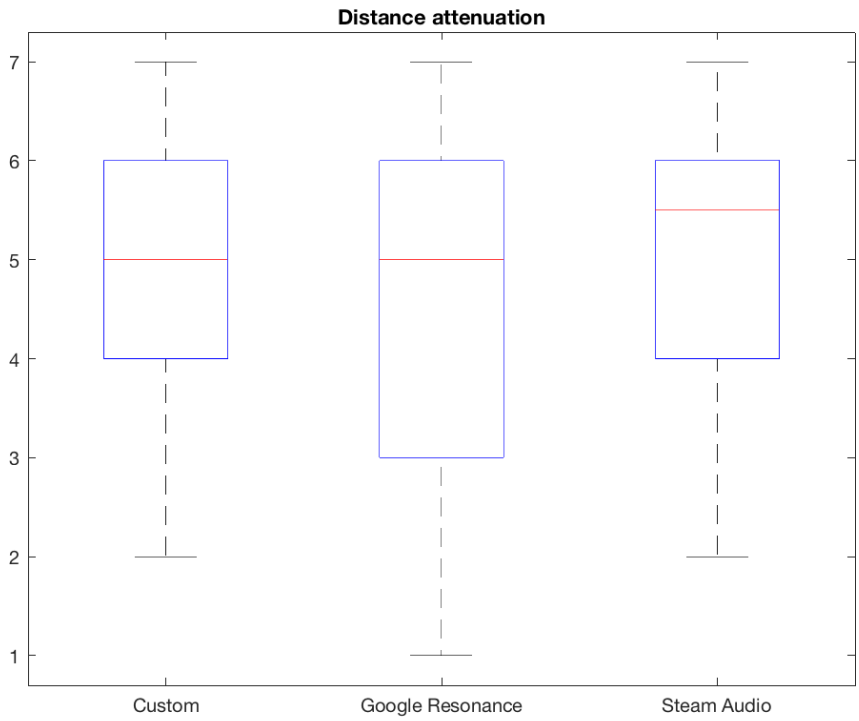
A.2 Example Code of SOUL: Peak Filter Implementation with fixed center frequency and bandwidth

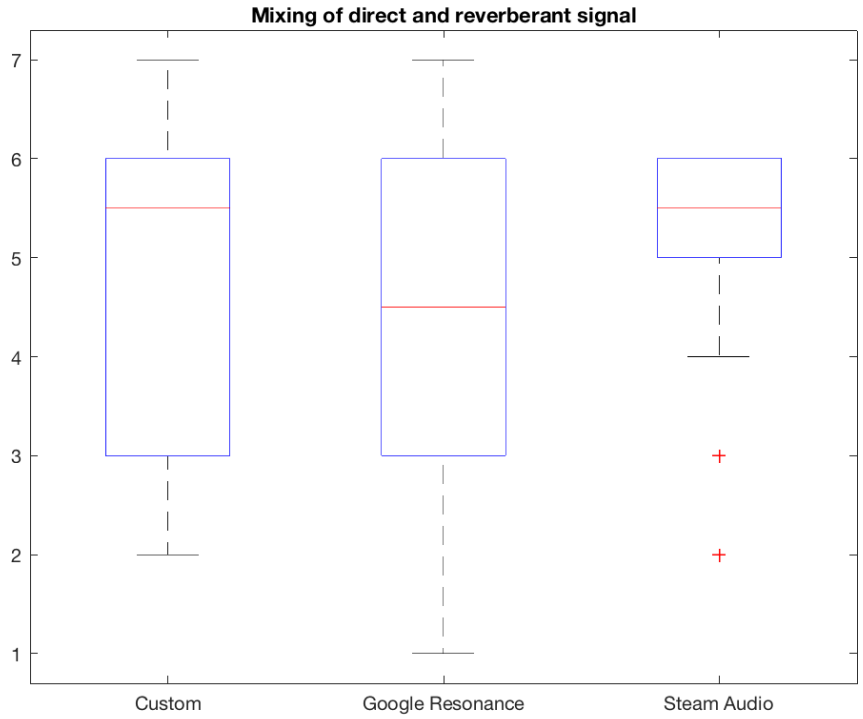
```
    processor PeakFilter (const float peakFreq, const float deltaFreq)
{
    input stream float audioIn;
    output stream float audioOut;

    const int fs = 44100;


    const float omega = (2 * pi * peakFreq) / fs;
    const float deltaOmega = (2 * pi * deltaFreq) / fs;

    const float beta = tan(deltaOmega / 2);

    const float b = 1 / (1 + beta);


    float coeff_yMin1 = 2 * b * cos(omega);
    float coeff_yMin2 = -2 * b + 1;


    const float coeff_x = 1 - b;
    const float coeff_xMin1 = 0.0f;
    const float coeff_xMin2 = -1 + b;

    float x, xMin1, xMin2, yMin1, yMin2, out;

    void run()
    {
        loop
        {

            x = audioIn;

            out =   coeff_x * x +
                    coeff_xMin1 * xMin1 +
                    coeff_xMin2 * xMin2 +
                    coeff_yMin1 * yMin1 +
                    coeff_yMin2 * yMin2;

            audioOut << out;

            yMin2 = yMin1;
            yMin1 = out;
```

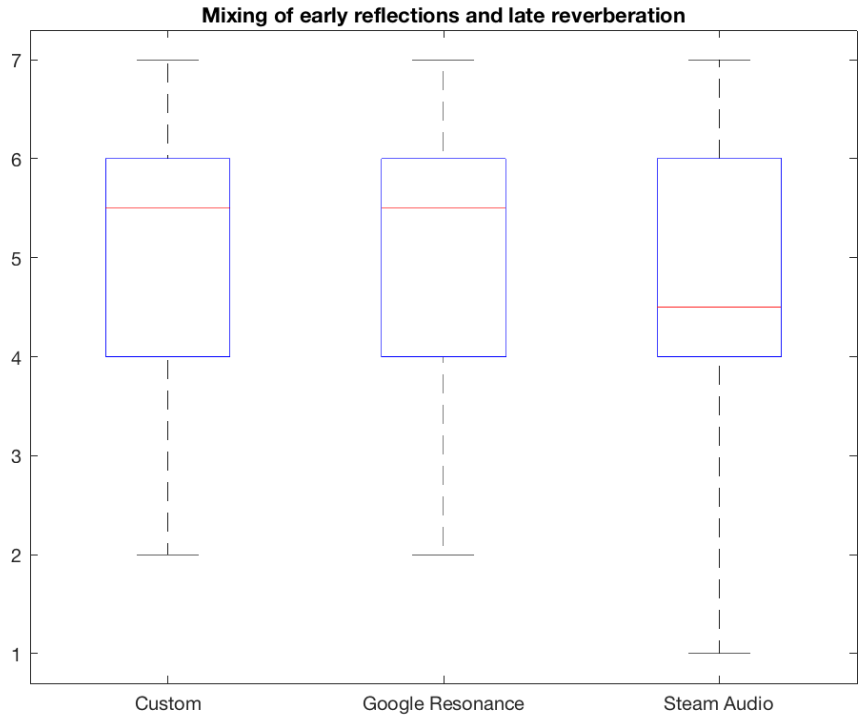
```
            xMin1 = x;
            xMin2 = xMin1;


            advance();
        }
    }
}
```

A.3 Boxplots of listening test ratings for each question and reverberator




Distance attenuation
7
6
5
4
3
2
1
Custom
Google Resonance
Steam Audio


Mixing of direct and reverberant signal
7
6
5
4
3
2
1
Custom
Google Resonance
Steam Audio


Mixing of early reflections and late reverberation
7
6
5
4
3
2
1
Custom
Google Resonance
Steam Audio

A.4 Spectrograms for all Impulse Responses

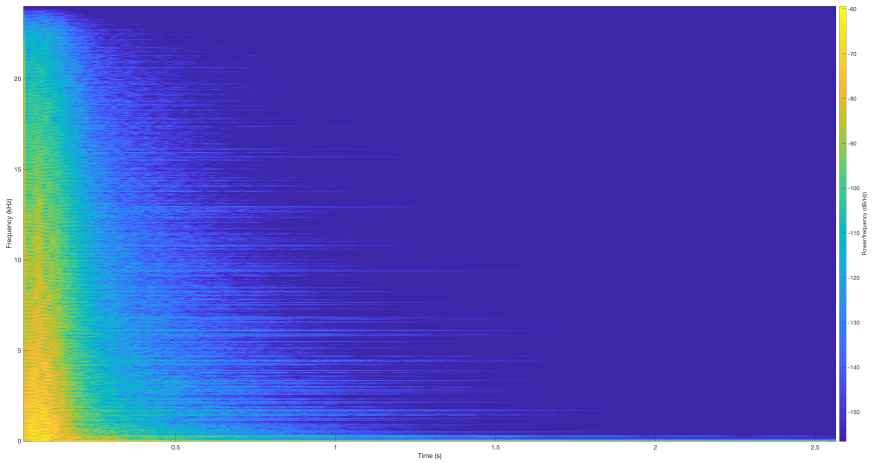


Figure A.1: Spectrogram of the impulse response of the designed reverberator

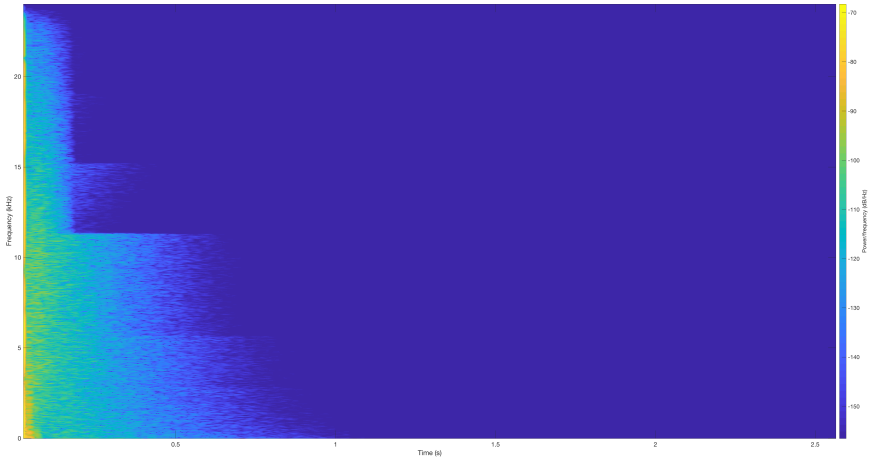


Figure A.2: Spectrogram of the impulse response of Google Resonance's algorithmic reverb

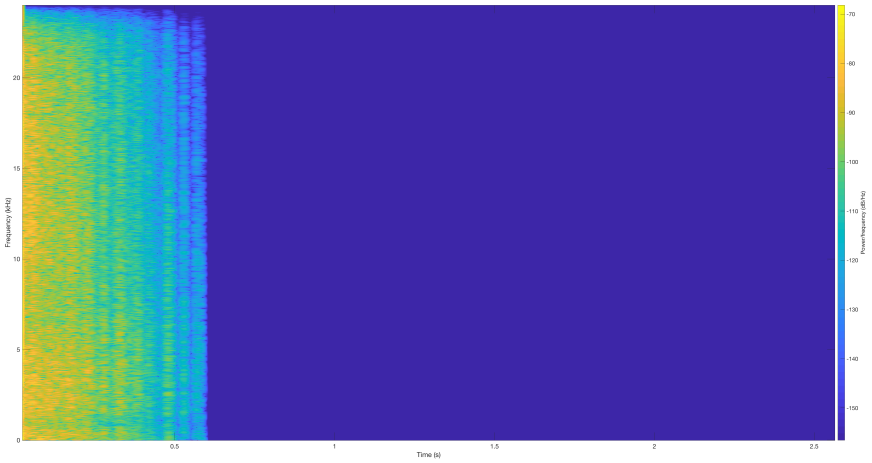


Figure A.3: Spectrogram of the impulse response of Steam Audio's geometry-based reverb

A.5 JUCE Plugin Code

The plugin code of the reverb is found in the folder path `CODE/PLUGIN`. The plugin can then be built from the `.jucer` file and the appropriate IDE.

A.6 Unity Environment

The Unity environment is found in the folder path `CODE/UNITY`. Under `Assets/Scenes`, there are 6 scenes available:

- `NATIVE_SCENE`: Scene containing the designed reverberator.
- `GR_SCENE`: Scene containing Google Resonance's implementation.
- `SA_SCENE`: Scene containing Steam Audio's implementation.
- `NATIVE_SCENE_OBJ`: Scene for capturing the designed reverberator's impulse response.
- `GR_SCENE_OBJ`: Scene for capturing Google Resonance's impulse response.
- `SA_SCENE_OBJ`: Scene capturing Steam Audio's impulse response.

  Under `Assets/Scripts`, the respective scripts can be found.

A.7 User Test Data

The data of the listening test rating can be found in `DOCS`