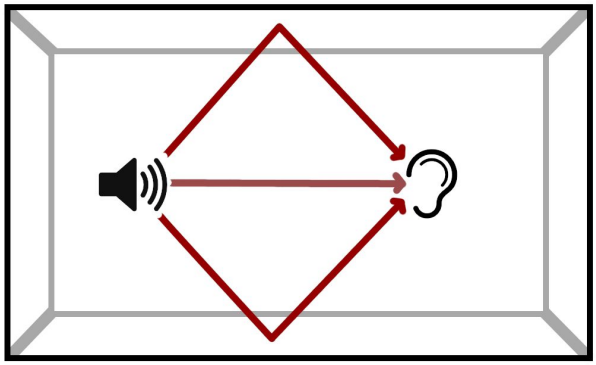
Real-time listening experience of feedback-delay network reverberation using geometrical acoustics

Sound - & music computing thesis - Anders Havmøller Laursen, 2019



Abstract

Acoustical modelling is complex and often face difficulties when having to run in real time, when both source and listener are moving dynamically and independently. Large amounts of calculations are required at run-time in order to produce plausible results. In order to mitigate the amount of calculations required at run-time, this paper investigates the use of radiance-transfer method to calculate relevant geometrical statistics, which can then be used to set the parameters of a feedback-delay network according to said statistics. Feedback delay networks are known for their low computational cost, and generally good overall quality and are thus popular in real-time applications. This project seeks to find an order of feedback-delay network that sits at a good middle ground between computational cost, and subjective quality experience.

A MURSHA inspired test revealed that significant differences in participant experiences are present when comparing lower order systems. Testing also revealed that it becomes gradually less impactful to the listening experience, when comparing higher order systems, beyond 8th order. Further testing is required for the overall listening experience of the system itself compared to other systems.

Keywords: *Signal processing, real-time reverberation, geometric acoustics, acoustics modelling, 3D game development, virtual architecture, MURSHA*

Table of contents

1. Introduction

The reflection of sound between surfaces in an environment is termed *reverberation*. This prolongs the sounds made within an environment, as the sound energy is dispersed within it. Reverberation differs from a simple echo, in that multiple reflections arrive to the listener at different times and directions. Reverberation carries with it spatial information, like shape and size of an environment. It also carries information about objects present and surface materials (*Välimäki et al, 2012*).

Due to the finite speed of sound (*343 m/s at room temperature*), the reflections happen over time and thus arrive in *stages*. These stages are typically broken into *early* - and *late reflections*. Along with the arrival of these reflections, the *direct sound* also arrives at the listener (*Savioja, Svensson 2015*) A sound will arrive at the listener after an initial travel time, termed T_0, and is then followed by the early reflections from the closest surfaces in the space. The direct path gives information about direction from listener to source, and the early reflections gives information about the surrounding geometry and shape. Late reflections are generally heard as a set of increasingly dense echoes, and carry information about the size of the space (*Bai, Richard, Daudet, 2015* & *Välimäki et al, 2012*). These late reflections decay over time and the the it takes for the reflections of an impulse response to decrease in strength by 60 db, is termed T_{60} and is the *decay time* of the reverberation (*Eaton, Gaubitch, Moore, Naylor, 2016* & *Välimäki et al, 2012*).

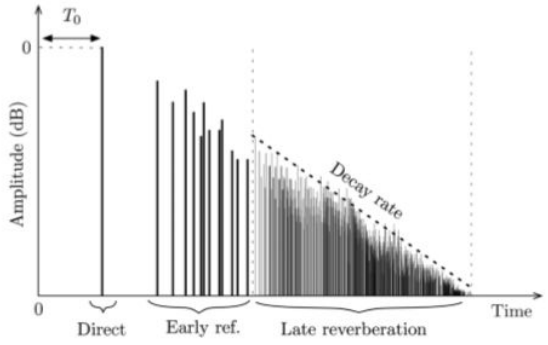


Figure 1.1: Simplified plot of reverberation reflection amplitude over time. Note the density of the late reflections (Välimäki et al, 2012).

Reverberation has been artificially replicated since the sixties, and is now of great interest when constructing concert halls, classrooms, railway stations and even homes. The process of listening to a

space before it is built, is known as *auralization* (*Savioja, Svensson 2015, Kleiner et al, 1993*). Reverberation algorithms have also gained popularity for modelling instrument bodies, like guitars, since the behaviour of mechanical vibrations in such instruments share a lot of physical features with room acoustics (*Välimäki et al, 2012*).

Some of the most utilized reverberation algorithms are still set according to personal experience or taste of the user though, which means that while high quality reverberation is achievable in simpler systems, their physical accuracy is lacking. However, the systems that are capable of accurately modelling reverberation from given 3D geometry are much more computationally expensive (*Bai, Richard, Daudet, 2015*).

Thus, this project seeks to implement a reverberation modelling algorithm that takes advantage of the speed of simpler reverberation systems, while utilizing the accuracy of the more complex reverberation modelling systems. This could be useful in real-time applications and could lead to a greater sense of presence for users. The general goal of this project is then to figure out how much designers can afford to reduce the reverberation system complexity and memory requirements, before the subjective listening experience suffers significantly.

1.1 Reader guide

This paper assumes that the reader has some familiarity with digital signal processing mathematics and algorithms.

The sections of this paper are as follows:

- Section 2 will go over a range of different approaches to artificial reverberation, and a few methods for modelling geometrical acoustics.
- Section 3 will provide a few state-of-the-art examples of how companies like Microsoft and Facebook's Oculus model acoustics in virtual environments.
- Section 4 provides an overview of the implemented system used in the final testing.
- Section 5 will describe the testing procedure and how it was designed
- Section 6 describes testing results, how the results were analyzed, what the data revealed and finally an overall conclusion.

2. Artificial reverberation

Reverberation is often used as an effect in different forms of media that utilize sound, like film, games, and of course music. In the early 20th century, this effect was often obtained using reverberation rooms, constructed specifically to produce reverberation for recording of music. Other techniques, like plate - and spring reverberation, were also invented. These techniques, while producing high-quality reverberation, were quite limiting. The effect was dictated by a pre-built environment, and maintenance and reliability of plate - and spring reverb was troublesome. The need for flexible and reliable artificial reverberation was clear (*Välimäki et al, 2012*). For digital reverberation, the *feedback comb filter*, proposed by Schroeder and Logan (*1961*) is one of the earliest systems that artificially reproduces reverberation. In 71, Gerzon proposed a multi-channel delay network fed through a feedback matrix. This work was later extended into what we today know as *Feedback-delay networks* (described in detail in section 2.2).

The work done by Schroeder lead to the first commercially available artificial reverberation device, the Lexington Delta T-101. During the 70's a range of digital reverberation devices were commercially released. In 79, Allen and Berkeley proposed the *image-source* method for room-acoustics generation. This image-source method is still used for virtual acoustics today, to calculate early reflection delay times and arrival angles. ISM will be described in detail in section 2.5

From this point, different forms of reverb will be described. Firstly, a simple structure composed of a single delay known as a *comb-filter* is introduced. The next section will then expand upon this structure to create the *feedback-delay network* structure. Following this, the field of geometrical acoustic modelling is introduced and some other algorithms are also described.

2.1 Comb filters

Feedback comb filters consist of a single delay line and attenuation path, which feeds back into the system recursively. The comb filter is named after the shape of its frequency response curve. This structure is shown in *figure 2.1*.

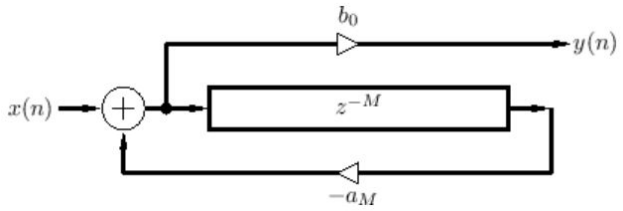


Figure 2.1: Diagram of a feedback comb filter. Here, the z^{-M} block is the delay line, and b_0 and $-a_M$ are the attenuation factors (Smith, Rocchesso 1996).

The filter output *y(n)*, where *n* is the sample number, is given by the following equation:

$$y(n) = b_0 x(n) - a_M y(n - M)$$

Equation 2.1

Where *x(n)* is the input, and b_o is an output attenuation. The attenuation coefficient $-a_M$ is generally supposed to be less than 1 in magnitude, in order to ensure stability.

Feedback comb filters produce discrete decaying echoes, when the delay time *M* is large enough to separate reflections in time (*Välimäki et al, 2012*). These echoes do not become denser over time, due to the structures utilization of only one delay line. Thus, each reflection will always feed back into the system every *M* samples.

2.2 Feedback-delay networks (FDN)

A feedback-delay network unlike the comb-filter, consists of multiple delay lines and a *transformation matrix*. The sound energy travelling through the feedback loop are *re-routed* to disperse sound energy between the different delay lines (*Välimäki et al, 2012*). This *transformation* of the sound energy is performed by the transformation matrix *A*, as seen in the top of figure 2.2. Delay line lengths are often also chosen as mutually prime lengths (*Bai,Richard,Daudet, 2015*).

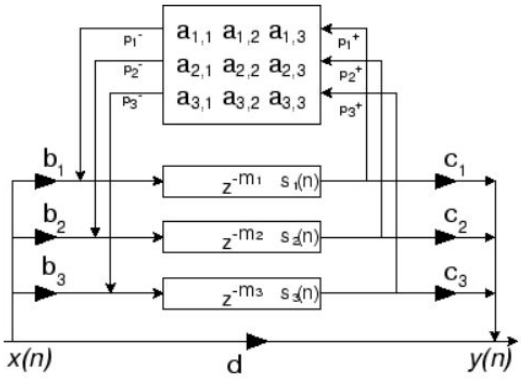


Figure 2.2: Diagram of an FDN reverb (Smith, 1996).

Figure 2.2 shows a diagram of a 3rd order FDN reverb system. The *order* of an FDN reverb is given by the number of delay lines in the network. Here, the z^{-m} blocks are the delays with output $s(n)$. The d, p_x, b_x, and c_x values are attenuations that make the system's output decay over time (*Smith, 1996*). FDN reverbs of this structure are given by the following equations

$$y(n) = \sum_{i=1}^{N} c_i s_i(n) + dx(n)$$

$$s_i(n+m_i) = \sum_{j=1}^{N} a_{i,j} s_j(n) + b_i x(n)$$

Equations 2.2 and 2.3

Where N is the number of delay lines, and $s_i(n)$ are the outputs from the terminations of delay line i. In vectorized form, the system can also be described by the following

$$\begin{bmatrix} x_1(n) \\ x_2(n) \\ x_3(n) \end{bmatrix} = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & g_3 \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} x_1(n-M_1) \\ x_2(n-M_2) \\ x_3(n-M_3) \end{bmatrix} + \begin{bmatrix} u_1(n) \\ u_2(n) \\ u_3(n) \end{bmatrix}$$

Equation 2.4

$$\begin{bmatrix} y_1(n) \\ y_2(n) \\ y_3(n) \end{bmatrix} = \begin{bmatrix} x_1(n-M_1) \\ x_2(n-M_2) \\ x_3(n-M_3) \end{bmatrix}$$

Equation 2.5

Where *u(n)* are the raw inputs, and *x(n)* are the inputs to the delay lines. The attenuation factors along the feedback loop are replaced with a single attenuation matrix *G* (*Smith, 1996*).

FDN reverberation are a common form of reverberation systems, and are popular due to their high quality and computational efficiency. As mentioned in section 1, these reverberation systems often have their parameters set manually through experience or preference (*Bai, Richard, Daudet, 2015*).

The delay lengths are often chosen around the *mean free path* within the space that is being simulated. This mean free path is given by:

$$\overline{d} = 4\frac{V}{S} \qquad \text{(mean free path)}$$

Equation 2.6

Where *V* is the volume of the space, and *S* is the surface area within that space. Using this equation, the average sample length of each delay can be found as follows:

$$\frac{\overline{d}}{cT} = \frac{1}{N}\sum_{i=1}^{N} M_i$$

Equation 2.7

Here, *c* is the speed of sound, *T* is the sampling period. *N* is the order of the FDN system and M_i are the delay lengths (*Smith, 1996*).

2.3 convolutional reverberation

Convolution reverberation simply works through the use of a recorded, or generated impulse response from some real or virtual environment. The formula for convolving one signal *x* with an impulse response *h*, yielding the output signal *y* is as follows:

$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{m=+\infty} x[m] \cdot h[n-m]$$

Equation 2.8: The general formula for convolution (Park, 2009)

Using such an operation, it is possible to fully reproduce the acoustics of said environment. However, though simple in concept, the computational cost of a convolution operation is quite high compared to other methods. Indeed, if a signal is convolved with an impulse response of 1 seconds length with a sample rate of 48 kHz, then 48000 multiplications and summations are required for every sample of the output for each channel of the system. This load increases linearly with impulse response length.

In order to conduct real-time reverberation using convolution, fast fourier transform is useful. This is because convolution can be done by multiplying the DFTs of each signal being convolved, and then applying the inverse DFT. Even when using this form of convolution, some delay is inevitable, and computational cost remains quite high (*Välimäki et al, 2012*).

2.4 Geometrical acoustics modelling

Geometrical acoustics aim to replicate impulse responses of real spaces, and is thus of great interest to this project.

Early implementations of geometrical acoustics considered the propagation of sound as rays that move perpendicular to the wavefronts of the sound, and would often reflect specularly off the surfaces in the given environment, but reflections of sound in the real world is mostly diffuse. Today, the mathematics used in 3D graphics for *global illumination* are often used. This field is commonly known as *radiosity*, and models the propagation of light over some number of reflections through an environment. Using this technique does require some alterations, since sound is of finite speed (*Savioja, Svensson 2015*).

In geometrical acoustics, the simplest case of a reflection is when considering a sound wave that encounters an infinite and perfectly smooth surface. A *local reaction model* is often assumed when conduction geometrical acoustic calculations. This means that the interaction between the air and the surface is a one-way interaction (*Savioja, Svensson 2015*).

The propagation of sound is often done by some sort of reflection tracing across a series of surfaces within an environment. The techniques differ in the geometrical shapes of the propagation, the calculations of the angles of reflection, and accuracy. In general, each technique works through a sender-receiver modelling, in which paths of reflection through the environment are determined, and often sorted to include the most relevant paths of reflection. The paths that arrive at the listener are then used to determine the final reverberation, through a range of data. This data could include the length of the path,

the number and angles of reflections, the materials encountered during propagation, and angle of arrival to the listener.

In the following sections, a range of different techniques are described. In practice, hybrid methods of such techniques are often used in geometrical acoustics modelling (*Bai, 2016*).

2.5 Image source rendering

An elegant and simple rendering technique for geometrical acoustics is the *image source* technique. This technique is also known as the *mirror source* solution, since it works by mirroring the sound source across each surface in the given environment. These reflections can then be regarded as secondary sources which are then treated in the same way recursively, until some condition is met. These mirrored sources are known as *virtual sources*. Such a termination condition could be a certain number of reflections, or a desired final signal length. A 2D example is given in figure 2.3.

The image source method is regarded as being highly accurate for box-shaped rooms, thanks to their 90-degree corners. These 90-degree corners eliminate the importance of diffraction in such cases.

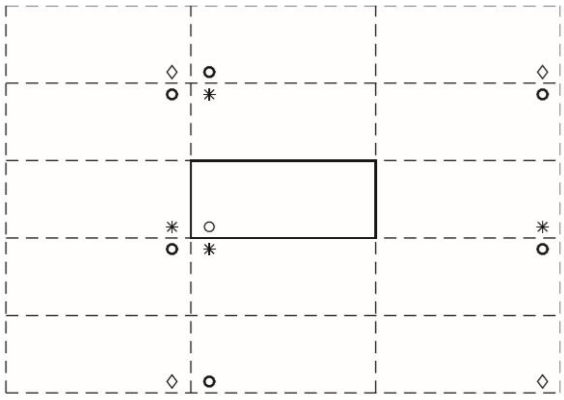


Figure 2.3: A top-down view of a shoebox shaped room, with the original source depicted as an o. First-order mirror sources are shown as stars, second-order sources are bold o's, and third order sources are squares (Savioja, Svensson 2015).

The number of reflections when using the image source method for K reflections is given by:

$$\sum_{k=1}^{K} N(N-1)^{k-1}$$

Equation 2.9

Where *N* is the number of surfaces. In the case of shoebox shaped rooms however, some reflections can be disregarded due to equality.

Image source calculations can be seen as a tree-structure, with the original sound source at its root, and each reflection makes up a branch. Impulse responses can be constructed through a summing operations for each branches overall contribution to the output signal.

For rooms of arbitrary geometry, it is possible to reduce the amount of mirrored sources that are valid to the listener. First, mirroring is only done from one side of surfaces, and surfaces behind the reflector in question are not needed. Secondly, specular reflection paths to the listener are created using the mirror sources. If this reflection path intersects any other reflector, it is discarded, since it is then not visible to the listener.

This method is regarded to be very efficient in finding early reflection paths, but leads to errors when looking for higher order reflections, due to the termination of the reflection paths whenever a mirrored source is obstructed from the listeners perspective, even though it may again be visible after higher order reflections. The exponential growth of mirrored sources also expensive, when calculating higher-order reflections, making it unfit for the modelling of late reflections in real-time applications (*Savioja, Svensson 2015*). It does however work efficiently for the modelling of early reflections at real-time speeds in both box-shaped rooms, and rooms of arbitrary geometry (*Bai, 2016*).

2.6 Path tracing

Path tracing algorithms work by extending a series of '*rays*' or '*paths*' from the source of the sound, and reflecting them off of surfaces until some number of these rays intersect with a volume that represents the position of the listener (*Funkhauser et. al, 2000*). When rays intersect the receiver volume, they are not terminated. Often, rays are allowed to continue their propagation through the environment in order to achieve higher-order reflections (*Savioja, Svensson 2015*).

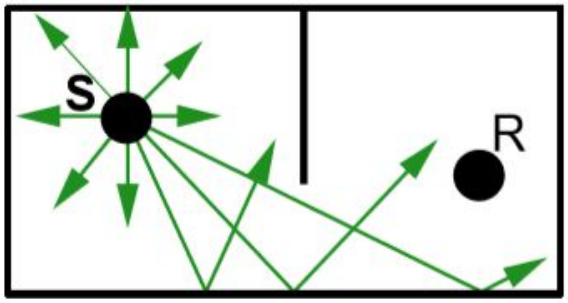


Figure 2.4: Simple overview of a ray tracing algorithm, where S is the source and R is the receiver (Funkhauser et.al, 2000).

There are several different types of path-tracing algorithms. While some of them treat reflections completely specularly, other implementations add randomness to the direction of the reflection, others create several new rays at each reflection, and some simply trace the direction at each reflection point directly to the listener
(*Savioja, Svensson 2015*). The following figure 2.5 depicts some of these versions of a path tracing algorithm. In (a), rays are treated in a way similar to the image source method by looking for specular reflection. In (b), ray reflections create several new rays, to model the diffuse reflections that occur in real-world reflections. In (c), a random direction is chosen for each reflection point. Finally, in (d), paths are traced from each specular reflection directly to the listener.

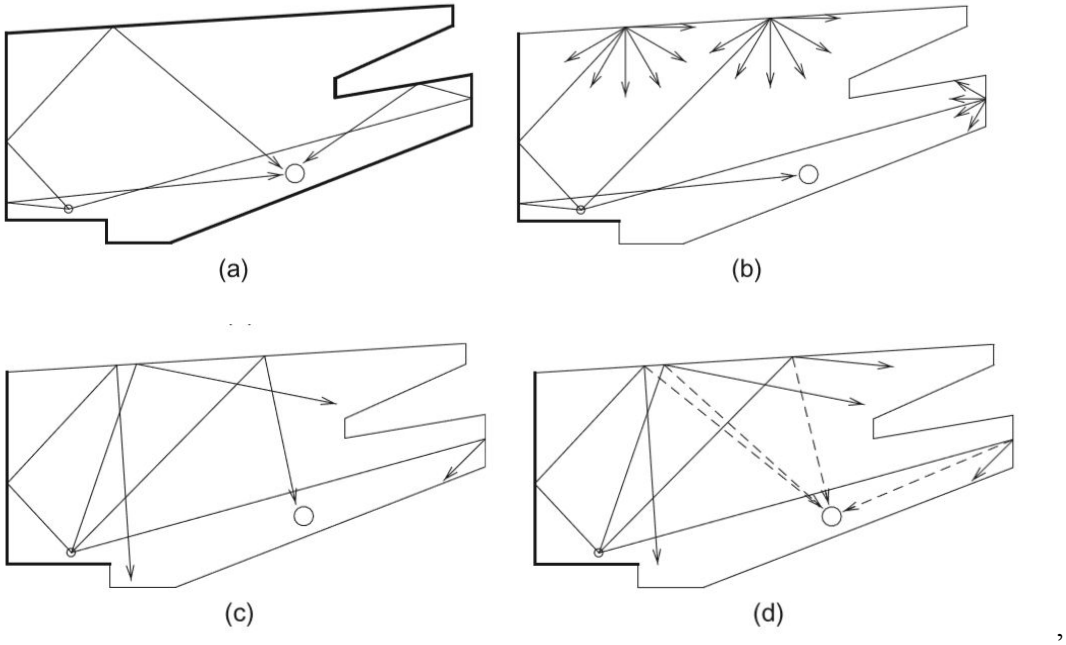


Figure 2.5: Overview of different path-tracing reflection models.

The path termination condition can be handled in multiple ways. One way is to let each path carry information about the amplitude of energy that it is transporting in some number of frequency bands, and terminate the ray once the overall energy falls below a given constant. Another termination condition is known as the 'russian roulette' method, where each reflection of a path has a certain random chance to terminate the path, based on material absorptivity. These approaches are considered to produce more accurate modelling of reflections than the image source method, because they can be treated as diffuse reflections, rather than purely specular (*Savioja, Svensson 2015*).

While the simplicity of ray-tracing algorithms are an advantage, the nature of their functionality may miss the most important paths, or due to the volumetric nature of the receiver it may not be perfectly accurate. To help in this regard, a large number of rays can be generated, but this requires higher amounts of computation. The volume size of the listener may also be increased, but this sacrifices accuracy. Furthermore, the greater the distance between sender and receiver, the greater the amount of necessary paths to calculate (*Funkhauser et. al, 2000, Savioja, Svensson 2015*).

2.7 Beam tracing

Beam tracing works similarly to path tracing, where the distribution of energy is traced from source to receiver through some amount of reflections. It is also similar to the image source technique described in section 2.5, since it mirrors the source across each surface. In general, the algorithm works by tracing beams through the environment, emanating from the source. The beams can have different shapes, and their cross-sections are often either rectangular, or triangles. These shapes can however change according to the geometry of their projections onto surfaces in the environment.

When beams intersect with surfaces in the environment, the beam is clipped to remove the occlusion region. Reflections are then created by a mirroring of the transmission frustum across the intersected surface (*Funkhauser et. al, 2000*). This is depicted in figure 2.6 below. Beam tracing is often either an extension of path tracing, or of image source rendering.

Reflections are continued until some criteria are reached, for example, when the listener is within the beams field. When a full series of beam reflections has been calculated, absorption characteristics are calculated for each reflection, based on materials intercepted and distances between reflection areas (*Monks, 1999*).

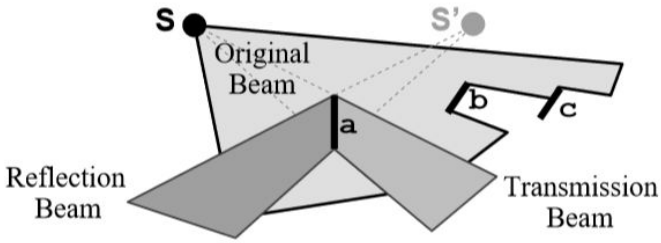


Figure 2.6: Beam tracing is done by mirroring the source S over surface a. The virtual source S' is then used to construct the subsequent beam reflected off of a (Funkhauser et. al, 2000).

a. as an extension of path tracing

In the case where beam tracing is an extension of path tracing, it works by tracing a series of beams, that work like volumetric frustums. These frustums are defined by their borders, which are treated as individual rays that encompass each frustum. If these rays intersect two or more surfaces, then the beam is split according to the number of surfaces encountered. The splitting of the beams produce more accurate results, but this technique is slower, since this leads to a higher number of beams than was originally projected from the source.

One technique is known as frustum-tracing, where frustum-shaped beams are propagated through the environment, and retain their shapes after reflections occur. While this does sacrifice accuracy, and may misrepresent some reflections, this technique is often used in real-time applications (*Savioja, Svensson 2015*).

b. as an extension of image source technique

The general aim of extending the image source technique, is to limit the number of virtual sources and thus increase efficiency of the overall algorithm. The general use of this form of beam tracing ensures that the number of beams remain constant, rather than splitting into more beams as propagation continues.

The image source beam tracing technique creates the first-order reflections similarly to the default image source technique. Here, a range of reflections have already been disregarded due to the image source techniques elimination process of invalid virtual sources. Then, beams are generated from these virtual sources, by using the reflecting surface that they represent as a border that defines the edges of the beam.

The beam tracing technique is considered more accurate than path tracing in arbitrary geometry, since each beam can represent an infinite number of rays within its region (*Savioja, Svensson 2015*).

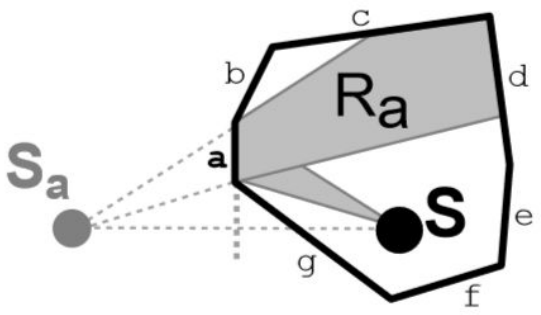


Figure 2.7: The virtual source S_a is used to construct the reflection beam R_a. R_a then only intersects surfaces c and d. This means that we can disregard all other surfaces for this particular beams subsequent reflections (Funkhauser et. al, 2000).

2.8 Diffractions in geometrical acoustics

Diffraction modelling is considered quite challenging for path tracing. Here, virtual portals are set up in the environments at valid edges in the geometry. When a path is traced through such a portal, it's trajectory is then modified based on it's distance to the edge in question.

In beam tracing, the locations of diffracting edges are often pre-computed. When one of these edges are present within the area of a beam, a new frustum is made which models the diffraction (*Taylor, et al, 2009*).

2.9 Radiance transfer method (RTM)

RTM is an algorithm that models the dispersion of energy between surfaces through diffuse reflections. This is done by discretizing the environment in question into some number of small surfaces, called *patches*. Each of these patches then calculate their relationship with each other patch. This relationship is known as a '*form factor*', and is given by the following equation (*Bai, Richard, Daudet, 2015 & Nettle, 1999*).

$$F_{i-j} = \frac{\cos\,\theta_i \cos\,\theta_j}{\pi r^2} H_{ij}\, dA_j$$

Equation 2.10

Here, Fij is the form factor between patches i and j. θ_i and θ_j are the angles between the normal vectors of the two patches, and a vector drawn between their centers. H_{ij} is a visibility factor, and A_j is the ratio of the surface areas of each of the patches being compared. The interaction is also shown in detail in figure 2.8.

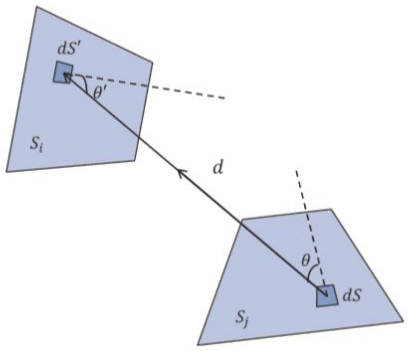


Figure 2.8: The form factor between patches is calculated using the θ angles (*Bai, Richard, Daudet, 2015*).

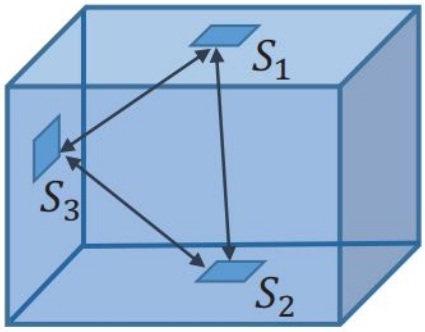


Figure 2.9: RTM works by calculating the form factor between each patch in the environment (*Bai, Richard, Daudet, 2015*)

The initial distribution of energy to each patch is known as the '*shooting matrix*'. Each patches radiation density after *n* reflections is given by

$$I_i^{(n)}(t) = I_i^{(n-1)}(t) + \rho_i \sum_{j=1, j\neq i}^{M} F_{i,j} I_j^{(n-1)}(t - \frac{r_{i,j}}{c})$$

Equation 2.11

Here, $r_{i,j}$ is the average distance between patches *i* and *j*, and $F_{i,j}$ is the form factor, which is given in equation 2.9, and c is the speed of sound. Then, the total radiation density of patch *i* at time *t* is given by

$$I_i(t) = \sum_{n=0}^{\infty} I_i^{(n)}(t)$$

Equation 2.12

Where $I_i^{(0)}(t)$ is the initial radiation density, as distributed directly from the sound source (*Bai, Richard, Daudet, 2015*).

RTM is generally used to calculate a '*room impulse response*' (RIR), and works in 3 steps:

1. *Distribute sound energy to each patch.*
2. *Let the sound energy propagate between the patches*
3. *After some number of reflections, the total radiation from each patch is gathered at the listener*

RTM is most often used to model the dispersion of light in 3D graphics, but has proven an accurate tool for modelling the dispersion of sound energy as well (*Bai, Richard, Daudet, 2015*). The main difference being, that while the propagation of light is modelled as an instantaneous, sound travels at a finite speed.

2.10 RTM to FDN

It is possible to set the parameters based on statistical operations of data obtained using the RTM algorithm. This has proven an accurate way of simulating late reverberation, while taking advantage of the computational efficiency of FDN reverbs, whose parameters are often set manually (*Bai, Richard, Daudet, 2015*).

Bai, Richard and Daudet (*2015*) proposed a system that did this. Using the distribution of distances between patches in and environment when using RTM, they were able to designate the lengths of some number of delay lengths for an FDN reverb. Using the form factors and material properties, they were

able to assign accurate parameters to the attenuations and transformation matrix of the FDN (*Bai, Richard, Daudet, 2015*).

Briefly, the delay lengths of an FDN reverb are set using RTM, by determining the most distances between all patches in the environment, and looking at the distribution of all patch-to-patch distances. In this way, the most prominent geometric information can be acquired using some clustering technique on the generated histogram of delay lengths . The transform matrix indices are are set according to a hadamard matrix, by an even energy grouping scheme of the form factors that creates N groups of form factors, where N is the number of delay lines in the FDN reverb being set up. By looking at the summed product of each groups form factors, it is possible to determine the amount of sound energy being transferred from one group to the other. Filtering coefficients are determined through the reflection coefficients of the materials of the surfaces. A more detailed overview of this process is given with relevant equations in the implementation section 4.

This however only accounts for the sound reflections after the sound has already been dispersed to the surfaces, and thus does not replicate the order-1 reflections accurately. For this to be possible, an extended FDN reverb structure was proposed, where *pre- and post-mix delay lines* are set up to account for the travel time of the sound from the source to disperse itself to the grouped patches, and the travel time for these late reflections to reach the listener. These pre- and post-mix delays can be set by determining average distance from both sender and receiver to each group of patches.

2.11 Room-to-room energy transfer

In the cases where the virtual geometry being modelled by geometrical acoustics is complex, and can be divided into several rooms, separated by doors or other openings, it is considered advantageous to render each room individually. Each rooms impulse response is modelled and then connected into a filtering tree which reflects the layout of the rooms (*Savioja, Svensson 2015*).

When a sound is played in one room, the acoustics of that room are applied. Then, the radiation exchange from this room into another room, where the listener is located, is calculated. Finally, this radiation is then altered by the acoustics of the room containing the listener. The exchange of energy from one room, to an adjacent room through an opening between them is fortunately quite simple. This can be modelled by creating virtual secondary point sources at the center of each element of geometry. It has been shown, that point sources used in this way produce results with the same accuracy as plane-sources (*Schröeder,*

Vorländer, 2007). This way, all that is needed is the sound reduction coefficient *R*. Each transmission path then governed by one of three transfer functions H_{ij} , H_{si} , and H_{rj}. In figure x.x, these transfer functions can be seen to describe the acoustics of room 0, the exchange of energy between geometric elements, and room 1.

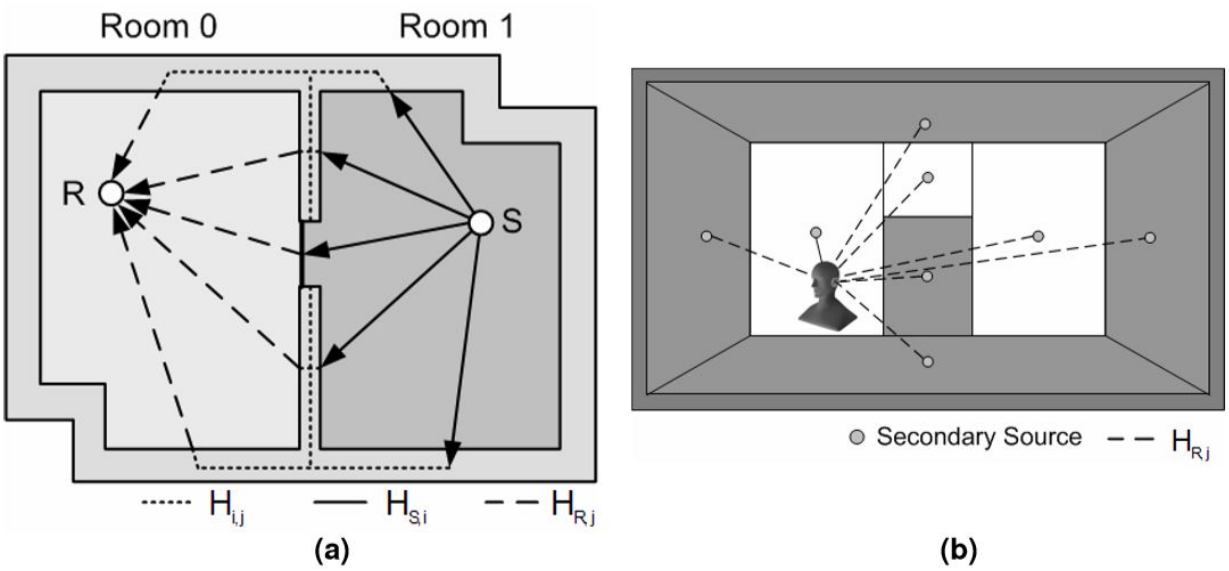


Figure 2.10: Using some number of virtual point sources and transfer functions, the propagation between rooms can be modelled in real-time with good accuracy (Schröeder, Vorländer, 2007)

The full transfer function of the room-to-room propagation, would then be given by the following equation, where *M* is the number of transfer functions in the full environment:

$$H = \sum_{j=0}^{\hat{N}} \left(\sum_{i=0}^{M} H_{S,i} \cdot H_{i,j} \right) \cdot H_{R,j}$$

Equation 2.13

The transfer function H_{ij} in figure 2.10 describes the transfer of sound energy between the different structural elements. This transfer function can be thought of as a filtering operation where the transmission coefficient *t* is related to *R*, by the following:

$$\tau_{i,j} = 10^{-R_{i,j}/10}$$

Equation 2.14

The signal from source S in another room from the listener, can be produced using the virtual sources S_j by:

$$S_j = S \cdot \sum_{i=0}^{M} H_{S,i} \cdot H_{i,j} \ , 0 < j < N$$

Equation 2.15

Of course, some restraints apply for a system like this. The number of point sources needs to be managed according to allocated computing power, and thus some accuracy is lost. In many cases however, the contribution of energy from the point sources located within solid geometry is negligible, and the system can produce good results when using only the virtual sound source located in the opening between the rooms (*Schröeder, Vorländer, 2007*).

3. State of the art

This section briefly looks at some current technology used in geometrical acoustics modelling, and how their techniques become capable of real-time reverberation rendering.

3.1 Microsoft Acoustics

Microsoft has released their own geometrical reverberation modelling for the Unity and Unreal game engines. This is known as *Project Acoustics*, and it is a wave-based approach that pre-bakes the acoustic characteristics of an environment. The baking step is very computationally expensive, but it helps ensure that the expensive computation is done before run-time. Then, during real-time, a range of *blend probes* positioned around the environment are used to determine what the final reverberation will sound like. The system allows for some developer control, so that an artistic freedom is maintained. For example, the reflectivity of surfaces can be set manually to reduce, or strengthen the reverberation.

Project Acoustics takes several physical phenomena into account, such as occlusion, portalling, and obstruction of sound, and is used in games such as Microsoft's own Gears Of War 4 (*Godin et. al., 2019 & Microsoft, 2016*).

3.2 Oculus acoustics

Oculus models geometrical acoustics by fitting a number of 'shoebox' enclosures onto a given environment. This model assumes that all surfaces reflect equally, that there are no occlusions beyond what the shoeboxes can represent, and that all the wall elements of the shoeboxes are orthogonally aligned. This is a highly simplified acoustic model, but manages to be much more effective in terms of performance than, for example, convolutional reverberation (*Oculus, 2019*).

4. Implementation

This section describes the RTM-FDN reverberation algorithm built by the author and used in testing. It is based on the work of Bai, Richard and Daudet (*2015*) as described in section 2.10.

This section will be a step-by-step walkthrough of the implemented RTM-FDN reverberation algorithm as it is of time of writing. These are the steps of the RTM-FDN algorithm in general terms:

1. Divide the geometry of the environment in question into patches.

2. Calculate form factors for all possible reflections and distribution of delay lengths.

3. Sort form factors according to an even-energy grouping scheme. Assign them to *N* groups, where *N* is the number of FDN delay lines.

4. Find the entries of the FDN transform matrix. Then the delay attenuation coefficients, the filter coefficients and the group-wise mean delay lengths.

5. Square root the entries of the FDN transform matrix, and manage signs according to the Hadamard matrix form.

6. Adjust delay lengths for optimal sound quality.

These steps are followed chronologically through the rest of the algorithm, which will be described in the remainder of this section.

The RTM-FDN algorithm was implemented in the Unity game engine. Here, walls and floors were made using simple square 3D surfaces known as *quads*. Each quad would have a *SurfacePatcher* script attached to them.
At run-time, the surfacePatcher script would divide each quad into some number of equal-size patches. The resolution of the patches can also be freely set by the user.

Another script, named *RTM_handler* would then find each of the surfaces in the environment and conduct the RTM-FDN calculations. These calculations were done in a series of steps.
The first step was to calculate all patch-to-patch form factors and distances. To calculate form factors, equation 9 is used much like in a traditional RTM algorithm.

In order to do this, a 6-layered nested loop structure was used. These loops would loop over each surface, the x - and y coordinates of each patch on that surface, each other surface, and finally the x - and y coordinates of patches on the other surface. For each interaction, information is saved using a custom class, named *PatchInteraction*. All PatchInteractions are saved to an array, after their respective values are calculated.
Following these calculations, the interactions are then sorted according to an even-energy grouping scheme described by Bai et. al. (*2015*). This scheme is based on the use of a hadamard transformation

matrix, where the transfer of energy from one delays output to another is of the same magnitude. With this in mind, the sorting of interactions can be done based on the transported energy of each interaction, which is described by their respective form factors. The grouping scheme firstly sorts all of the interactions in descending order, and then splits the entire list of interactions into a number of bins, each of which contains *N* form factors, where *N* is the order of the system we are setting up. The grouping is then done in a circular fashion as shown in the following diagram:

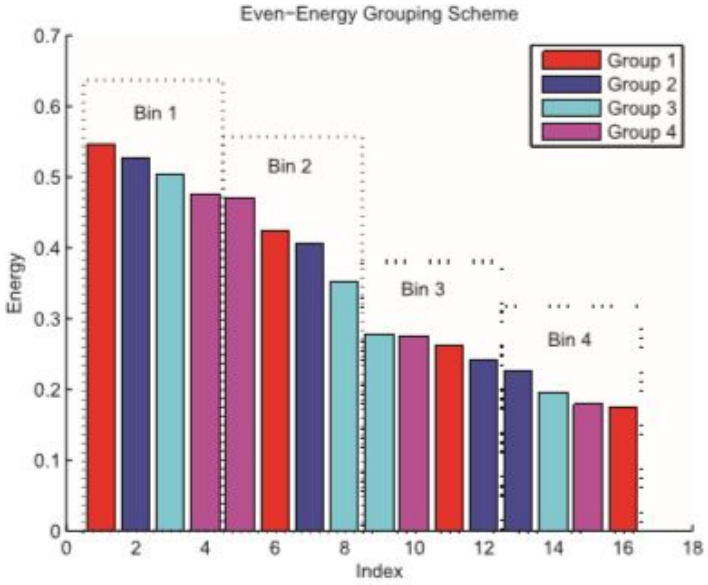


Figure 4.1: The even-energy grouping scheme used to designate interactions into N groups, where N is the system order.

This is done by a simple modulo indexing operation that sorts each interaction into a 2D list of the *PatchInteraction* class. The mean integer sample number for each groups delay times is then found and logged. These are the sample number delay times used in the delays of the FDN structure.

The calculations of attenuations is done by considering the formfactor of each interaction and the reflection coefficients of the surfaces in question. This gives us the following equation:

$$\bar{\theta}_n^s = \sum_{i \to j \in \wp_n} F_{i,j} \theta_i^s$$

Equation 4.1

Where θ^s_n is the average reflection coefficient in frequency band *s* in group *n* of patches. θ^s_i is then the reflection coefficient for patch *i*. Using similar logic, we can calculate the first order filter coefficients for each group by taking a ratio of zero-frequency - and nyquist frequency reflection coefficients as follows:

$$b_{\mathrm{p}} = \frac{\bar{\theta}^{0}{}_{\mathrm{n}} - \bar{\theta}^{\mathrm{fs}}{}_{\mathrm{n}}}{\bar{\theta}^{0}{}_{\mathrm{n}} + \bar{\theta}^{\mathrm{fs}}{}_{\mathrm{n}}}$$

Equation 4.2

Where Θ^{fs}_n is the mean reflection coefficient of group *n* at the nyquist frequency, and Θ^{0}_n is the mean reflection coefficient of group *n* at zero frequency.

In order to then calculate the transform matrix, we then look at how much energy one group transports to another. To do this, we can look at the total energy transported by group *m*, and how much of this energy is transferred from *m* to group *n*. Since the magnitude of each index in the matrix should never exceed a magnitude of 1, a ratio between transported and total energy is used. Thus the matrix entry a_{mn} can be calculated as follows:

$$a_{mn} = \frac{\ell_{m,n}}{\ell_m}$$

Equation 4.3

Where l_{mn} is the energy transported from group *m* to group *n*, and l_m is the total energy within group *m*. These values can be calculated as follows:

$$\ell_{m,n} = \sum_{i \to j \in \wp_m} \sum_{j \to k \in \wp_n} F_{i,j} F_{j,k}.$$

Equation 4.4

$$\ell_m = \sum_{i \to j \in \wp_m} F_{i,j},$$

Equation 4.5

The entries a_{mn} in the matrix are then square-rooted and their signs are switched according the to hadamard matrix. According to the conservation of energy, and with ideal precision of form factor calculations:

$$\sum_{n=1}^{N} a_{mn} = 1.$$

Equation 4.6

For example, an order 8 system should have each matrix entry be around ⅛. The system implemented produces values within the range of 0.124 and 0.1255. These small inaccuracies can be attributed to the simplified and less accurate calculation of the form factors in the previous steps.

Due to the even-energy grouping scheme, the distribution of delay distances between patches is practically equal to one another as well. In order to set delay lengths for better reverberation quality, some inspiration from regular FDN networks is taken. Here, the base delay length is set according to the mean free path, and subsequent delays have their lengths set according to a 1:1.5 scaling. Delay lengths are also set to prime numbers in order to ensure that reflections will not overlap, which improves sound quality (*Bai, Richard, Daudet, 2015*).

Since the RTM-FDN reverberation calculates the delays and attenuations from patch to patch, the reflections modelled by this method are from 2nd order reflections and onward. In order to model 1st order reflections, the ISM method described in section 2.5 was used. To find the specular reflection paths between each surface, the custom *Reflector* script was made. This script would find the positions of source and listener within the 3D space, and then find the paths between each of these objects and the surface center. This gives the reflector two 3D vectors that describe the positions of source and listener in local space. These vectors would then be specularly reflected and multiplied by -1 in order to define the virtual source and listener positions on the other side of the surface. Using two raycasts, the reflector then determines if the paths are free from other obstacles. If the paths are free, then there is a valid specular reflection path. The total length of the reflection path is then used to determine the delay time in samples, and the surface is used to determine filtering and attenuation.

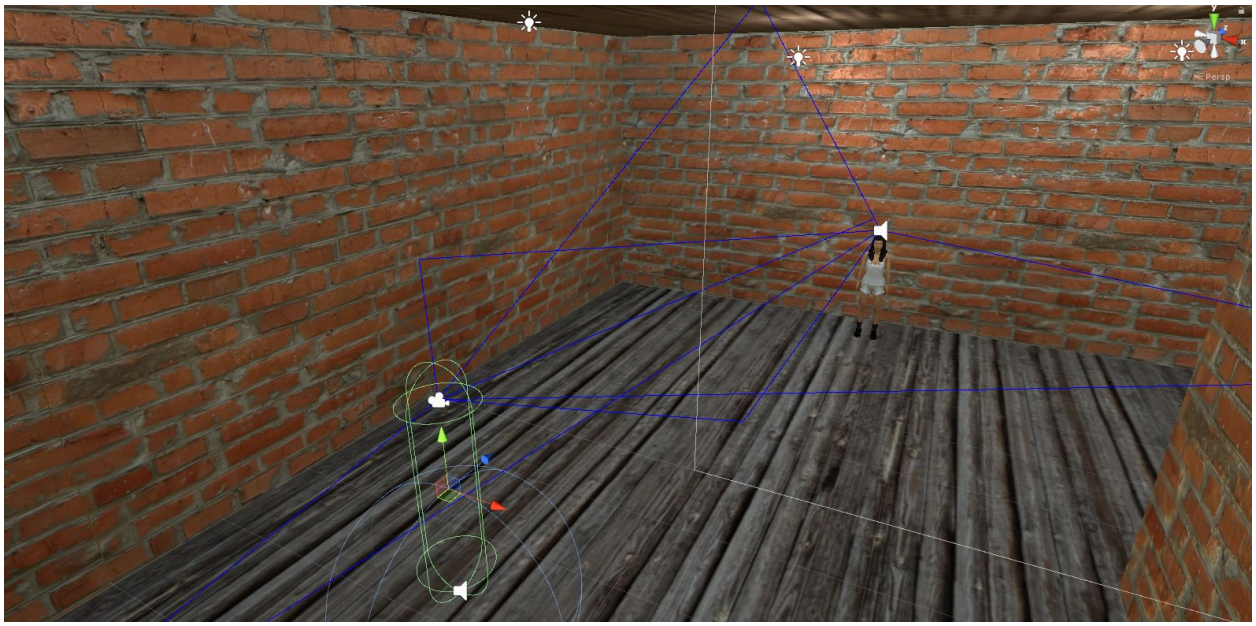


Illustration 4.1: The ISM paths are displayed here using the blue lines.

In order to produce the reverberation effect itself, a script called *FDN_reverb* was attached to the listener object in the scene. This script used Unity's *OnAudioFilterRead* method to process sound data in the scene, and calculating a final output. This script used circular buffers of float values as the delays of the system. For each channel of audio, the sound in the scene would be fed into these circular buffers, and read back as a vector. This vector would then be multiplied by the designed transformation matrix and filtered according to the calculations done in the RTM calculations. ISM reflections are handled independently by another series of circular buffers and filters, and added to the result. The total output is then played back to the user.

5. Method

Geometrical acoustics works off of some resolution, especially when in real-time. Other real-time geometrical acoustic models are mostly intended to improve plausibility and by extension user presence. Therefore, some subjective measure of believability of the reverberation is needed.
For real-time applications, such as games, the illusion is mostly considered more important than realism . Testing believability of some aspect of a real-time simulation is done with some derivation of a *MUSHRA test*. This kind of test aims to measure the subjective experience of a sounds quality, and is mostly used to find out how users experience the quality of different kinds of audio compression. In this way, this testing procedure holds many advantages in determining a less biased measure of the users listening experience.

MUSHRA stands for '*multiple-stimulus with hidden reference and anchors*' and is usually used to test the listening experience of compressed audio. The *multiple stimulus* refers to the different versions of an audio signal that test participants will listen to during the test. During the test, all different versions will be freely available to listeners to switch between and compare. This helps listeners compare between the different versions more easily. When dealing with compressed audio, users would also listen to the original uncompressed sound before the test, in order to ensure that they know what the sound is supposed to sound like with no compression. In this case, when testing an artificial reverberation algorithm, a 16 high-order system is used as this reference. This reference signal is also one of the settings available during the test itself, and should always be rated quite highly compared to the other settings. If this is not the case, then the listener may be deemed unreliable.

The *anchors* of a MUSHRA test are added to ensure that the whole range of the rating scale is used. One anchor is used to ensure that the lowest end of the scale is used, and should be the lowest quality signal. Another anchor is used to ensure that the middle of the rating scale is used and should be of average quality, compared to all other available settings. These anchors are usually low-pass filtered versions of the reference sound. In this case however, the subjective experience of reverberation is tested, not the experience of a compression algorithm. What this test seeks to find is an acceptable middle ground for the experience of reverberation quality, and computational expense in a real-time setting. For this reason, it was chosen to simply use a 2nd-order system for the low anchor, and an 8th-order system as the middle anchor. Since the system is not yet ready to compute an impulse response, as would be done with a traditional RTM system, a 20th-order system is used as reference.

The rating scale that users would use to rate the listening experience is a 0 to 100 integer scale, divided into 5 portions: 0 to 20 as ‘bad’, 20 to 40 as ‘poor’, 40 to 60 as ‘fair’, 60 to 80 as ‘good’, and 80 to 100 as ‘excellent’.

In this case, instead of different compression techniques of an existing audio signal, the aim is to find a good compromise between computational expense, and maintenance of the illusion of real spatial sound for a given room. Therefore, the testing of the RTM-FDN-ISM system was done by altering the order of the system itself to achieve multiple different experimental conditions. Users would be seated in front of a classical computer display and be wearing a set of noise-cancelling headphones (where? What do they need to know before starting? What are the classical settings of a Mushra? How do we collect and analyze results? Listeners need access to ‘what the sound is really like’ when dealing with compression - we simply use a very high-order system. We need to rate something other than ‘sound quality’ - maybe ‘how much the sound sounds like it is happening in the given space?’ - ‘spatial quality?’).

They would then be asked to rate a range of different reverberation settings on a 0-100 integer rating scale per setting. Users would be able to freely switch between setting modes and control when a short sound clip was played. The sound clip used was of an anechoic voice signal of 8 seconds in length.

In order for the testing procedure to remain relevant to its intended use, the environment was modelled in 3D, and surfaces textured using photographic 3D textures with normal maps. The source that would produce the sound clip was implemented as a female character, in order to visualize where the sound was coming from. A simple mouth-movement script was also made, to help make it clear who was speaking, when the clip was played. The mouth movements were based on the average amplitude of windowed portions of the sound, and the characters jaw would move up and down accordingly.

Illustration 4.2: the GUI used during testing. The green square indicates which reverb the user has currently selected. The orange button in the upper right lets users commit their ratings, which would then be saved on the computer as a txt file.

The interface that participants would use consisted of a mouse-controlled series of sliders, each of which were tied to a certain FDN reverb. They would be able to freely switch between FDN reverbs, and grade them according to the reference. The reference would be playable at all times during the test, allowing users to easily compare each reverb with it. Each slider would be tied to a random reverb, so that the sequence was eliminated as a potential bias, as per MURSHA testing recommendations. By pressing the ‘e’ key, users would be able to play the clip through their currently selected reverb, and by pressing the ‘f’ key, they could listen to the reference Users would also be able to look around freely within the environment. Their movement was however restricted.

6. Results

Of the 20 participants that underwent the MURSHA test, 5 of them were deemed unreliable due to their high rating of the lowest anchor setting. These 5 participants rated the 2nd order reverberation as 40, or above. This left a total sample size of 15 participants.

6.1 MURSHA test results

The distributions of scores given to each order of FDN reverberation can be seen in the boxplots below. Here, the order-20 reverb is the reference, and the order-2 reverb is the low anchor. As we can see, the full range of scores were utilized, and there appears to be a steady increase in general scores according to the order of the reverb.

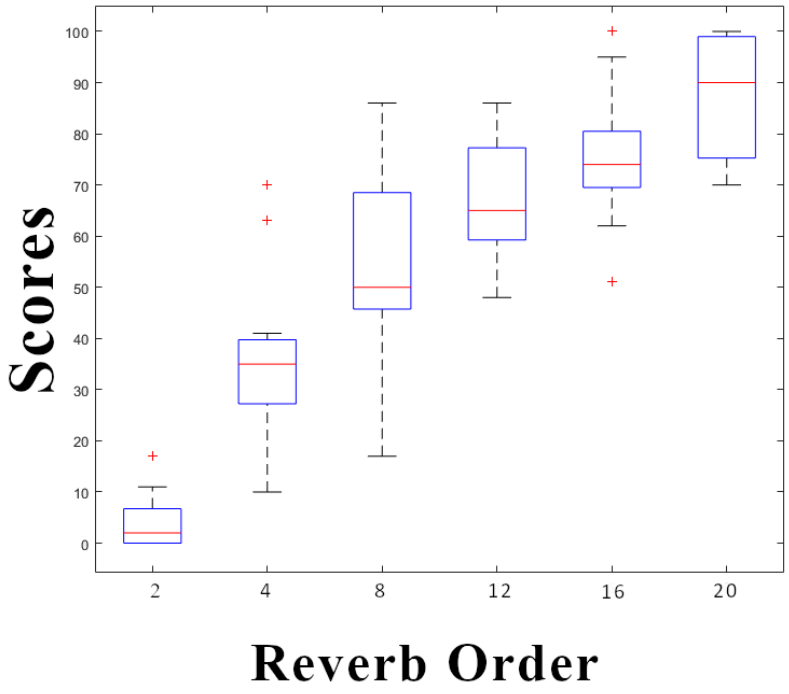


Figure 6.1: Boxplots of distributions of assigned scores for each order of FDN reverb

6.2 Analysis of results

In order to analyse the distributions of scores for each reverb, Matlab's *multcompare* method was used. This method uses a one-way ANOVA test and multiple-comparisons procedures, like Tukey's HSD

(*honestly significant difference*) criterion. This type of data analysis is founded on the MURSHA test recommendation by ITU (*2015*). The resulting test matrix *c* is as follows:

Group a	Group b	Lower confidence interval	Estimate	Upper confidence interval	p
1.0000	2.0000	-44.9640	-31.2667	-17.5693	0.0000
1.0000	3.0000	-64.4974	-50.8000	-37.1026	0.0000
1.0000	4.0000	-75.6974	-62.0000	-48.3026	0.0000
1.0000	5.0000	-85.4974	-71.8000	-58.1026	0.0000
1.0000	6.0000	-96.6974	-83.0000	-69.3026	0.0000
2.0000	3.0000	-33.2307	-19.5333	-5.8360	0.0011
2.0000	4.0000	-44.4307	-30.7333	-17.0360	0.0000
2.0000	5.0000	-54.2307	-40.5333	-26.8360	0.0000
2.0000	6.0000	-65.4307	-51.7333	-38.0360	0.0000
3.0000	4.0000	-24.8974	-11.2000	2.4974	0.1735
3.0000	5.0000	-34.6974	-21.0000	-7.3026	0.0003
3.0000	6.0000	-45.8974	-32.2000	-18.5026	0.0000
4.0000	5.0000	-23.4974	-9.8000	3.8974	0.3042
4.0000	6.0000	-34.6974	-21.0000	-7.3026	0.0003
5.0000	6.0000	-24.8974	-11.2000	2.4974	0.1735

Here, the first two columns indicate the groups *a* and *b*, being compared in each row of the test matrix. The groups are numbered as follows:

- Group 1 is the score distribution for the 2nd order FDN reverb
- Group 2 is the score distribution for the 4th order FDN reverb
- Group 3 is the score distribution for the 8th order FDN reverb
- Group 4 is the score distribution for the 12th order FDN reverb
- Group 5 is the score distribution for the 16th order FDN reverb
- Group 6 is the score distribution for the 20th order FDN reverb

From this, it is possible to say that not all groups show statistically significant differences between each other, at a 95% confidence level. Specifically, the higher the order of systems being compared, the less confidence can be maintained in their statistically significant differences. Looking at the boxplots in figure 6.1, we can also see a clear overlap of the distributions among the higher order system scores.

This aligns with the results of Bai, Richard and Daudet (*2015*), that indicate diminishing returns in subjective experience of reverberation quality for higher order FDN reverberation, with statistically significant decrease in scores with FDN reverbs of the 8th order and below. This means that 8th order systems are a good benchmark minimum order, if good reverberation quality is to be maintained with this system.

6.3 Discussion

Even though the results of the test seem to reinforce the research upon which this project was based, there are some issues that may have affected the testing procedure, and by extension the results. The testing was conducted through convenient sampling at the Copenhagen campus of Aalborg University. While the testing was done with noise-cancelling headphones, the surrounding environment was not controlled according to MURSHA standards. Testing was done during the early hours of the day when less noise was present in the halls, and testing was ceased during the lunch break to avoid the worst noise. Testing then continued in the group rooms of the medialogy bachelor students. The change in environment may also have been a factor. During testing, it was also found that a user had altered the output volume of the computer used for testing, which may have caused subsequent testers to have a harder time hearing the audio clearly.

Furthermore, while MURSHA recommends participants to be experienced listeners, there was no screening process to infer how experienced the participants were. Out of the 20 participants that completed the test, 5 of them were deemed unreliable. They may have misunderstood the task or the interface. This is considered the most likely, since many participants in general asked for clarification of both task and GUI before starting the test itself. Furthermore, testing was only conducted for a single virtual environment.

While the results indicate a general scale of experienced quality in relation to system order, the test cannot indicate how *convincing* the listening experience was in the given virtual environment. Further testing of a different nature would be required to conclude this part of the listening experience. For this, a classic RTM systems generated impulse response could be used to construct a reference signal through convolution, and this signal could then be used in a somewhat similar MURSHA test.

6.4 Future work

There are some acoustical details that are not considered in the current build of both the FDN and ISM systems. Specifically, the pre- and post-mix delays in the FDN reverb are currently hard-coded, which means that they are only accurate for a single given setup of source and listener. This is also the case for the early reflection paths calculated with the ISM. Furthermore, the early reflections are not properly spatialized, meaning that the system is currently only capable of producing a mono output for all reflections. The ratio of amplitudes between the dry and wet signals are also not properly informed.

6.5 Conclusion

This project sought to find a way to inform the choice of system order in an FDN reverberation algorithm, taking computational expense and listening into account. The system was adjusted to virtual 3D geometry with RTM, and complemented by early reflections modelled through ISM.

The MURSHA-inspired test that was conducted revealed that FDN systems of order 8 is a good minimum for the listening experience, and that the subjective experienced quality of the reverberation is negatively impacted much more clearly between systems of order 4 and 2, than with orders 8, 12, 16 and 20. Further testing is however needed to assess the subjective listening experience of the system compared to other systems and compared to real-world acoustics. To do this, the system needs further functionality, and the reference signal used for testing should be computed through classical RTM calculations of impulse response, to be used in a convolution reverb.

Bibliography

1. Välimäki V., Parker J., Savioja L., Smith J., Abel J., 2012, *'Fifty Years of Artificial Reverberation', IEEE/ACM Transactions On Audio, Speech, And Language Processing, Vol. 20, No. 5*

2. Bai H., Richard G., Daudet L., 2015, *'Late Reverberation Synthesis: From Radiance Transfer to Feedback Delay Networks', IEEE/ACM Transactions On Audio, Speech, And Language Processing, Vol. 23, No. 12*

3. Eaton J., Gaubitch N., Moore A., Naylor P., 2016, *'Estimation of Room Acoustic Parameters: The ACE Challenge', IEEE/ACM Transactions On Audio, Speech, And Language Processing, Vol. 24, No. 10*

4. M. R. Schroeder and B. F. Logan, 1961, *"Colorless artificial reverberation," J. Audio Eng. Soc., vol. 9, No. 3*

5. Smith J., 2010, "*Physical Audio Signal Processing for Virtual Musical Instruments and Audio Effects*", *Center for Computer Research in Music and Acoustics, Stanford University*

6. Smith J., Rocchesso D., 1996, *'Circulant and Elliptic Feedback Delay Networks for Artificial Reverberation', IEEE/ACM Transactions On Audio, Speech, And Language Processing, Vol 5, No. 1*

7. Nettle P., 1999, *'Radiosity in English II: Form Factor Calculation' accessed 25/1 2019, https://www.gamedev.net/articles/programming/graphics/radiosity-in-english-ii-form-factor-calculation-r653/*

8. Funkhauser T., Carlbom I., Elko G., Pingali G., Sondhi M., West J., *'A Beam Tracing Approach to Acoustic Modelling for Interactive Virtual Environments', 2000, Bell Laboratories*

9. Kajiya J., *'The Rendering Equation', 1986, California Institute of Technology*

10. Veach E., Guibas L., *'Metropolis Light Transport', 1997, Computer Science Department, Stanford University*

11. Monks, '*Audioptimization: Goal-Based Acoustics Design*', *1999, Cornell University*

12. Unity, 2018: '*https://docs.unity3d.com/Manual/UnderstandingFrustum.htmlI*', accessed 11/4 2019.

13. Savioja L., Svensson P., '*Overview of geometrical room acoustics modelling techniques*', *2015, The Journal of the Acoustical Society of America 138*

14. Kleiner M., Dalenbäck B., Svensson P., '*Auralization - An overview*', *1993*, *J. Audio Eng Soc. 41(11)*

15. Taylor M., Chandak A., Ren Z., Lauterbach C., Manocha D., 2009, '*Fast Edge Diffraction For Sound Propagation In Complex Virtual Environments*', *Proceedings of the EAA Auralization Symposiom*

16. Schröder D., Vorländer M., 2007, '*Hybrid method for room acoustic simulation in real-time*', *in Proceedings of the 19th International Congress on Acoustics*

17. Livingstone D., 2006, '*Turing's Test and Believable AI in Games*', *ACM Computers in entertainment, Vol. 4, University of Paisley*

18. Bai H., 2016, '*Hybrid models for acoustic reverberation*', *Télécom ParisTech*

19. Mason A.J, 2002, '*The MUSHRA audio subjective test method*'

20. ITU, 2015, '*Method for the subjective assessment of intermediate quality level of audio systems*', *The ITU Radiocommunication Assembly, Geneva.*

21. Tae Hong Park, 2009, '*Introduction to Digital Signal Processing*', *World Scientific*

22. Godin K., Christiani T., Sharkey K., Harvey B., Cross N., Strande H., accessed 10/8 2019, '*What Is Project Acoustics?*, *https://docs.microsoft.com/en-us/azure/cognitive-services/acoustics/what-is-acoustics* , *Microsoft*

23. Oculus, accessed 12/8 2019, '*Environmental Modelling*', *Oculus developer page: https://developer.oculus.com/documentation/audiosdk/latest/concepts/audio-intro-env-modeling/*