Creating a visual model of audio quality in rooms/venues

By

Joseph Davison

A report on project work carried out for the degree of

BSC (Hons) Music Technology and Audio systems

1st Supervisor Ian Gibson

2nd Supervisor XXXX XXXXXX

xx/xx/22

# Abstract

Understanding room acoustics intuitively is quite hard for humans to do especially as you change environment variables…

Contents

[Abstract 2](#_Toc121373619)

[Table of Figures 4](#_Toc121373620)

[Introduction 5](#_Toc121373621)

[Background Research 6](#_Toc121373622)

[Spatial model 6](#_Toc121373623)

[Defining an Index Score 10](#_Toc121373624)

[Visualisation of the Index Score 12](#_Toc121373625)

[Plan of Work 15](#_Toc121373626)

[Deployment 15](#_Toc121373627)

[Back-end Design 15](#_Toc121373628)

[Prototype Code Examples 16](#_Toc121373629)

[Creating an Algorithm 18](#_Toc121373630)

[Design Ideas and Optimisations 18](#_Toc121373631)

[Prototype Code Example 19](#_Toc121373632)

[Visualiser 20](#_Toc121373633)

[System Overview 20](#_Toc121373634)

[Plan 21](#_Toc121373635)

[References 22](#_Toc121373636)

[Appendix 23](#_Toc121373637)

# Table of Figures

[Figure 1: Visualisation of the reflected rooms. 8](#_Toc121373510)

[Figure 2: Example of an EQ plugin with a spectrogram. 12](#_Toc121373511)

[Figure 3: Colour showing loudness of level monitors 14](#_Toc121373512)

[Figure 4: API calls from node.js file 16](#_Toc121373513)

[Figure 5: placeholder C++ backend algorithm 17](#_Toc121373514)

[Figure 6: very early prototype algorithm 19](#_Toc121373515)

[Figure 7: system overview flow chart 20](#_Toc121373516)

[Figure 8: deadlines timetable 21](#_Toc121373517)

# Introduction

This paper outlines research conducted prior to the production of a web-based software solution aimed at audio engineers who are looking to model a venue, room, or other fully enclosed 3D space to gain information about it. This product will be public access and available via an internet browser, where users can upload information, wait in a queue for it to be processed, then once it is processed, they can download a file which will allow them to visualise audio quality throughout the venue. This queue system will allow a wider range of individuals to have access to the product, as all intensive computation is performed externally, however, it does mean that code running on the server side will have to be extremely fast and efficient otherwise users might be waiting hours or days for their results, which, should that happen, the product will be seen as a failure.

The first chapter of this report discusses literature surrounding the topics of spatial audio modelling, qualitative measurements of audio in a 3D space, and methods of visualising music and audio. research methodology, data collection…

The second chapter of this report discusses the design choices made from the results of the research as well as a system overview and plan of work. research methodology, data collection…

The outcome of this project should be a system that is both fast and efficient, which can accurately measure and then display the audio quality of a room in real-time while conveying as much information as possible in a comprehensible manner.

# Background Research

## Spatial model

To calculate audio within a space obviously a method for modelling audio in a 3D space is needed, many solutions to this problem exist with a large range of complexity, however, with complexity comes both computation cost as well as implementation time, because of this, certain higher-level models will have to be left out of consideration for the final system.

Open air/no reflections/simple space? Why I’m choosing enclosed?

One method is described in *Parametric Directional Coding for Precomputed Sound Propagation,* N. Raghuvanshi & J. Snyder (2018). In this paper Raghuvanshi and Snyder outline a system for precomputing large virtual spaces so that a binaural head related impulse response (HRIR) can later be calculated in real time as a listening position is arbitrarily moved around anywhere in the space. The main point of their paper was to create a system for realistic audio in video games (Microsoft, n.d.) meaning the precomputation time is not a problem due to encoder only needing to be “baked” once per level and then can be shipped out with the game, in their results Raghuvanshi and Snyder state on a single 8-core machine the pre-calculations can be performed in between 20 minutes and 5-6 hours depending on input parameters. While these calculations can be run in parallel allowing more powerful modern multi-core chips to complete the same calculations in less time, this is still unreasonable for a system which is being designed for general access where each new user might add hours of computation time to the queue.

The paper *A Universal Deep Room Acoustics Estimator*, P. S. Lopez, P. Callens & M. Cernak (2021) while not specifically about propagation modelling, is still worth mentioning. This paper proposes a method for using a neural network to predict signal to noise ratio (SNR) along with 5 room acoustic parameters (RT60, C50, C80, DRR, and STI) from an input reverberant and noisy speech signal. Their results showed that the model could follow the ideal estimation reasonably close with some variance. This brings up an important question of what role artificial intelligence could play in the implementation of the model, a similar system which trains using 3D maps of a room and room impulse responses (RIR) could be quite successful in predicting reverb given a specific input sample, however, rather than finding impulse responses (IR) in a diffuse room, the point of this research is to create a system that can model the frequency spectrum at every given point in a room, this means that if real world data were to be used it would have to be hundreds of measurements and the exact locations of those measurements across multiple hundreds of venues which would be difficult to gather individually, in their paper Lopez, Callens & Cernak do reveal a few open source data sets used for their training, however, as stated above, this system would need a slightly different set which cannot be found open source or otherwise. One alternate method of training could be to use a system like the one outlined in Raghuvanshi & Snyder (2018) to pre-compute large spaces, take virtual measurements within that space and then feed that resulting data set into a neural network, however, implementation time for that method might exceed the bounds of this project and simply from a user experience stand point very few people will have a perfect 3D map complete with complicated absorption coefficients for whichever venue they might want to model.

Another method is described in *A Simple Method for Calculating the Distribution of Sound Pressure Levels within an Enclosure,* B. M. Gibbs & D. K. Jones (1972). This paper outlines a method for modelling sound propagation by imagining a 3D array of tessellated cuboids surrounding a real room with a sound source and a receiver where each adjacent box mirrors its surrounding boxes therefore representing what a wavefront would see as it reflects around the room, however, the wavefront is moving outward into the imaginary rooms instead of reflecting. This allows for simple wave calculations to be used to model the propagation as if the source is in completely free space moving radially outwards, If you calculate the distance between each of the reflected imaginary points and the source you can gather delay time and if you calculate which walls the wave has passed through to get to a point then you can figure out a chain of absorption coefficients to multiply the final signal by.

A picture containing outdoor object

Description automatically generated

Figure : Visualisation of the reflected rooms. The smaller dot is the source, the larger dot is the receiver, and the blank circles are the imaginary reflected points.

This method is much simpler to implement and should be able to run much faster than the other methods discussed in this paper while giving fairly accurate results for a given space, however, no room is a perfect cuboid and rooms will not only contain static objects within it, which might scatter or absorb audio, but also dynamic objects, such as humans. J. S. Bradley (1996) shows there is a simple linear relationship between the absorption coefficient of a crowd and the area of ground covered, this means that should a user be able to predict attendance the system could incorporate the effect of crowds upon the reverberation time, unfortunately crowds also create their own noise which to simulate an additional sound source could possibly be included in the centre of the room emitting some form of white noise like “crowd signal” however including this would greatly increase run time by possibly half if an additional source needs to added to the original two. Furthermore, crowds will grow and diminish through time, even during a show, meaning they are ultimately impossible to predict. Additionally, the further the shape of the users venue strays from the ideal cuboid, the further the model will stray from reality, meaning there is a possibility for users to receive wildly different results from what they see in real life and some might not even be able to approximate their venue to a cube in the first place, for example large concert halls which might have irregular walls or rounded edges, this will cause the pool of possible users of the system to be decreased.

Possibly include research from the paper discussing modern usage of Gibbs, Jones method– they discuss air absorption and uneven absorption coefficients

The method outlined in Gibbs & Jones (1972) is clearly not perfect but what it loses in accuracy it does make up for with simplicity and computation time which is a far more important quality in this system, therefore this method has been chosen to be the basis upon which work will be done to add functionality and refine it into a final system, a plan for this work will be discussed in the next chapter.

Representing the room in this manner allows for a very simple equation to be used to find the resulting function at each imaginary point by finding the length from the source and therefore delay time τ.

Xn(t) = (α1 \* α2 \* …) \* αair \* F(t - τ)

Where αx is the absorption coefficient of each wall the wave passes through, αair is the magnitude lost to the inverse square law, t is time, X(t) is function at an imaginary point with n being the point number, and F(t - τ) is the source function delayed by time gathered from distance to source.

Then summing each of these functions gives the resulting real function at that specific point

Y(t) = SUM 0-n Xn(t)

One can then perform a Fourier transformation on to get the frequency domain at regular intervals.

yHAT(f) = INTEGRAL Y(t) dt

if you do this for…

## Defining an Index Score

The point of this project is to create a visual model of audio quality throughout a room and to do that we must define a formula by which to judge audio quality at each given point.

Firstly, there is a lot to be gained from looking at other similar ideas like the speech intelligibility index (SII), which is a simple method for predicting specifically how intelligible speech is within a given space. The two main inputs for this index are audibility which is a signal to noise function for a frequency band and a frequency importance function which gives a weighting to each band for how important that specific band is to the interpretation of the speech, as outlined in (ANSI/ASA, 1997). Obviously, this index is very heavily based on speech and how different frequency bands are more important than others for comprehension, unfortunately, isn’t true with music as the totality of the spectrum is important, the general idea is you want the listener to receive a signal where the whole spectrum is as close as possible to the output plus possible desired room reverb, it could be argued that perhaps a greater focus might be wanted for typical instrument bands, however, that will still cover a very large area. SII is clearly built for speech, but the basic outline of a combination of parameters, normalised to a 0 – 1 scale, can easily be carried forward into the creation of the formula.

Secondly, industry standards in room acoustics should be considered to approach this problem, for example, in their paper *A Universal Deep Room Acoustics Estimator,* Lopez, Callens & Cernak (2021) use Reverberation Time (RT60), Direct to reverberant ratio (DRR), Clarity (C50 & C80), and Speech Transmission Index (STI) as they are “basic acoustic parameters that characterize the environment well”, all of these parameters would be useful in giving insight into a room’s behaviour however, again, fewer calculations are better due to the focus on run time, therefore only RT60, DRR, and C80 will be Considered. The use of these standards should hopefully mean that it is intuitive to understand for engineers who would already be familiar with such ideas, however, it is also important that an untrained person can get a good idea of quality, so simplicity is key.

Finally, the frequency domain should be considered. Because two sound sources are being used, even if all walls were 100% absorbent, there will be constructive and destructive interference between waves, alternatively, if the walls were highly reflective it would then also be important to be able to model phenomena like standing waves. one possible way to include this could be calculating the correlation between the frequency domain of the receiver and both left and right sources or perhaps left and right receivers separately with corresponding sources, however, the total calculations necessary should be kept to a minimum for this algorithm due to the importance of low computation time.

Although possibly subject to change due to computation time optimisations, the basic formula will look something like this:

Y =

## Visualisation of the Index Score

When creating a visual model, the main aim is to make it as intuitive as possible, so the end user must spend as little time to decipher the results as possible. A quick look at other visualisation methods across the music industry shows that, while there is a lot of divergence, there are a few specific patterns that often show up.

One of these is the idea that the visualisation will dynamically move along with the music, like how a digital EQ might show a spectrogram of the output, this is important as music by its nature is somewhat ephemeral, you cannot understand the music if you experience it all at once, therefore, the visualisation must develop alongside the music in the same way the acoustics of the room will develop.

A picture containing chart

Description automatically generated

Figure : Example of an EQ plugin with a spectrogram.

Another common attribute of visualisation systems is interactivity, an example of this can again be seen in EQ plugins, Fig 1 above shows toggles over the spectrogram which the user can use to form the EQ curve which is then reflected immediately by the spectrogram as well as within the audio playback. Although there is an argument to be made against the use of visual feedback when mixing a track, that if you rely more on your ears alone you can avoid bias coming from the visualiser allowing you to truly get the best sounding curve, it is still obvious that the visual input/output for these systems makes it much easier to use and comprehend data especially to a new or untrained user. The visualisation system this paper is trying to zero in on will not have to worry about controlling audio output, conversely, its main function is to try and convey large amounts of data to a user in a form that they can understand and an interactive or draggable map might allow the user to see all aspects of the room that could be missed if it was simply static as well as allowing them to focus in on specific areas.

Another common pattern in music visualisation and design in general is the use of colour to convey information about something, one example of the use of colour is in level monitors where quieter signals are displayed as green, louder signals become more yellow/orange, and once the signal begins clipping it will display as red. The use of colour to explain the state of something is quite a fundamental part of visual design and so its inclusion in this system is undisputable, the visualiser will most likely use a similar colour layout to the level monitors, a 0 or bad value will be displayed as red, low to medium values will be displayed as orange and medium to good values will be displayed as green.

A screenshot of a computer

Description automatically generated with low confidence

Figure : Colour showing loudness of level monitors, (University of Washington 2022)

The point of including these design patterns is to try to make the system blend in with other visualisation systems hopefully providing a sense of familiarity to a user, improving both the initial acclimatisation as well as the overall efficiency of displaying room characteristics.

Ultimately, the visualisation should be successful if these conditions are met:

* It is dynamic and can change along with audio.
* It is intuitive and simple to understand by use of typical design choices i.e., colour.
* It allows the user full comprehension of the 3D space.

# Plan of Work

## Deployment

### Back-end Design

The main user access point for this system will be a web page which will double as both the file upload and queue system as well as the visualisation player. The server providing this web page will use the node.js runtime and will serve both browser documents like HTML and JavaScript and handle API calls for file transfers, node has been chosen simply because of the execution speed and concurrency advantages over PHP with Apache allowing for a smoother user experience. The node.js server will use nginx as a reverse proxy to interact with the worldwide web, this provides little benefit with low user volume, other than caching static content, however, should the system require scaling in the future, nginx can handle drastically more concurrent requests than legacy web servers as well as perform load balancing if provided with multiple endpoint destinations.

### Prototype Code Examples

To supplement research, an early website design was created to know what the methods of deployment of this system might look like, however, unlike the algorithm prototypes, the below code is likely very close to the final product.



Figure : API calls from node.js file

The figure above shows all API calls available to the browser in the current prototype system with the content of the middleware functions removed to aid readability. Each URL is sensibly named for the function it serves and “/:username/” corresponds to where the browser will insert the users username In the API call.



Figure : placeholder C++ backend algorithm

The figure above shows the placeholder C++ program which checks for file uploads via a MySQL queue table and then create a new random file and delete the uploaded file. The functions selectInt() and selectString() take SQL string inputs and return whatever type they say in their names, and ticketReady() sets the MySQL queue table row to ready so that the node.js server knows it can download the output file to the user.

Complete examples of the code can be found in the project GitHub repository (Davison, 2022).

## Creating an Algorithm

### Design Ideas and Optimisations

The main consideration for this algorithm is compute time, creating an algorithm that can run quickly relies on two main factors, initial design choices and subsequent code optimisation. The first design choice to be made when creating software is the paradigm within which to write code; Functional Programming has one major advantage in the area of speed due to the referential transparency of functions (if a == b then f(a) == f(b)) which allows for parallelism (Amiana, 2022), functional programming can also provide reduced memory allocation and usage when compared to something like Object Oriented Programming (OOP) which could increase speed, however, that depends more heavily on implementation rather than paradigm. Secondly, on the topic of parallelism, rather than typical multi-threaded CPU processing, GPU acceleration can be used to decimate the amount of data each thread needs to compute, drastically speeding up the system.

Finally, for design, the language used to write a program can have a massive impact on its efficiency and run time, the programming language Python is notorious for its trade off compute speed for ease of implementation and simple syntax, whereas slightly more low-level languages like C and C++ are able to effectively compile straight down into machine code allowing for very fast running executables with the trade-off being complexity, leading to increased production time and risk of problems like memory leaks. Due to the importance of speed the algorithm will initially be written in C++ despite the additional complexity that must be navigated, however, further implementations might be created in other languages so they can be “drag raced” to find which runs quickest, notably, rust will be considered for this additional testing due to its memory efficiency and comparative run speed to C and C++.

Optimisations in code are difficult to consider before writing, however, with such large amounts of data it will likely be possible to interpolate or compress it at some point in the process either before processing, speeding up the compute time, or after which will save on memory and disk space as well as bandwidth when transferring data and allowing playback of the visualisation file to be smoother.

### Prototype Code Example

In addition to the research, a few simple early implementations of this algorithm were created to guide the literature review.



Figure : very early prototype algorithm

The figure above shows the main function for the prototype algorithm. This program will look very different to the finished product, however, the basic idea is there; for each point in the room x, y, z, get the time taken for audio to get to it, then get the amplitude of the signal at that point. The main big differences of this program are that this does not take an audio file input and doesn’t consider reflections or any other extra room parameters.

Complete examples of the code can be found in the project GitHub repository (Davison, 2022).

## Visualiser

Rendering of the visualiser will be performed using JavaScript and the open-source library webGL due to its high performance and the fact it can run in browser so no extra software or drivers would be required to be distributed. The development of this visualisation system will occur alongside the development of the main server-side algorithm or more specifically the output of the server-side algorithm, as the output file configuration will have to be known before the rendering system will be built. Its requirements are that it is interactive and that each frame can be fully rendered quickly.

## System Overview

The basic flow of the system will look something like this:

Diagram

Description automatically generated

Figure : system overview flow chart

## Plan

Below is a table outlining deadlines for the development of the system, heavier workloads have been included earlier in the pipeline allowing for greater agility if problems arise allowing for slightly less critical tasks to be removed e.g., testing Rust against C++ or web page aesthetics. In a worst-case scenario, there is also the ability to truncate some of the important tasks by removing extra functionality, for example, implementing CPU parallelisation rather than completely refactoring for GPU computation or not implementing data interpolation. Additionally work on the report will be naturally added throughout production as areas get completed and the complete report 2 point is simply to represent refinement and adjustments before submission.

|  |  |
| --- | --- |
| December/January | C++  Implement Gibbs & Jones (1972) model  Implement index formula  Parallelisation / GPU acceleration |
| March | Web page  Update web page inputs/MySQL tables for required input parameters  Create blank interactive visualiser in webGL  C++  File output format  Output file compression  Web page  Working visualiser in webGL |
| April | C++  Output data interpolation  General optimisation  Port to Rust and compare  Web page  Make web page look pretty  Complete report 2 |
| May | Submission date: 1/5/2023 |

Figure : deadlines timetable

# References

Raghuvanshi, N. & Snyder, J. (2018).Parametric Directional Coding for Precomputed Sound Propagation. *ACM Transactions on Graphics*, 37(4), 1-14, Article 108. https://doi.org/10.1145/3197517.3201339

Microsoft. (n.d.) *Project Triton*. Retrieved December 2, 2022, from https://www.microsoft.com/en-us/research/project/project-triton/

*Gibbs, B.M. & Jones, D.K. (1972).* A Simple Image Method for Calculating the Distribution of Sound Pressure Levels within an Enclosure*. Acta Acustica united with Acustica,* 26(1), 24-32(9).

Bradley, J.S. (1996). The sound absorption of occupied auditorium seating. *Journal of the Acoustical Society of America*, 99, 990-995. https://doi.org/10.1121/1.414562

ANSI/ASA. (2020). *Methods For Calculation of The Speech Intelligibility Index* (Standard No. S3.5-1997 (R2020)). Retrieved from https://webstore.ansi.org/standards/asa/ansiasas31997r2020

Lopez, P. S., Callens, P. & Cernak, M. (2021). A Universal Deep Room Acoustics Estimator*. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. https://doi.org/10.48550/arXiv.2109.14436

University of Washington. (2022). *Ableton Live Online Curriculum*. Retrieved December 5, 2022, from https://itconnect.uw.edu/tools-services-support/teaching learning/workshops/online-tutorials/digital-audio-online-curriculum/ableton-live-online curriculum/

Amiana, D. (2022, April 8). *OOP vs Functional vs Procedural. Scaler Topics*. https://www.scaler.com/topics/java/oop-vs-functional-vs-procedural/

Davison, J. (2022). *Individual Project Repository.* GitHub Repository. https://github.com/Nocturnalx/individual\_project

# Appendix