Implementation of 2D Ray Tracing in Csound for Convolution Reverberation

Presented by:

Dharanipathi Rathna Kumar

A thesis submitted for the degree of:

Creative Music Technologies, Master of Arts

Maynooth University
National University of Ireland Maynooth

Department of Music

Maynooth University, NUI Maynooth

Co. Kildare, Ireland

Submission: **September 2018**

Head of Department:

Prof. Christopher Morris, BMus, MA, PhD

Research Supervisor:

Dr. Iain McCurdy, PhD

Table of Contents

List of Figures

Abstract

This project discusses methods for creating 2D ray tracing-based room reverberation algorithm and its effectiveness and its limitations. Modern digital reverbs are typically either algorithmic or convolution based. There are advantages and disadvantages for both these types. The main disadvantage of the convolution-based reverb is that the Impulse response used for the convolution is static. There isn't much the user can change. For different room dimensions, different impulse response has to be used. This project focuses on this limitation of the convolution-based reverb. With the help of 2D ray tracing different impulse response files for different dimensions of a room can be created. With this method the user has a parametric control to define various parameters while creating the impulse response. In this project we begin with a general introduction about reverberation and its types. And then look at the mathematical algorithm used for tracing a ray in a 2D plane. And its step by step implementation in Csound. We then test the application by providing different set of input parameters and then comparing the output impulse response file with the impulse response of various rooms and with the impulse response of various artificial algorithmic reverb opcodes and plugins. finally, limitations of this method and steps to improve the existing algorithm and ideas for future work are discussed.

Acknowledgement

I would like to express my profound gratitude to my thesis advisor Dr. Iain McCurdy from the music department at Maynooth University for his patience, continuous support and constant encouragement. His guidance helped me throughout the time of the research and documentation of this thesis. This accomplishment would not have been possible without him.

I would also like to thank my friends Niall O'Connor and Damien McEvoy for their support and motivation throughout the time of the research.

Last but not the least I am also grateful to the members of Maynooth University's Music department faculties Dr. Gordon Delap and Prof. Victor Lazzarini, my family and my friends for supporting me throughout my masters. Thank you all.

1. Introduction

1.1 Motivation

Reverberation is an acoustic phenomenon that we hear every day. Reverberation is essential to music because it adds a sense of space to it. Music without reverberation makes it unnatural and not very pleasing since almost everything we hear has reverberation in it. Artificial reverberation has been an important subject of interest in the field of commercial music recording. Since all recording studios are acoustically treated, use of artificial reverb is essential. Artificial reverberation has come a long way since its beginning but in terms of methods and standards.

I was always fascinated with delays, echoes and reverbs in the audio digital processing world. the main idea for the thesis project came to me while I was working on my second semester musical signal processing project. I developed a convolution-based Echo plugin called Tap Tap echo using Csound and cabbage. The main principle behind that plugin was to create an impulse response with a single echo written into it and convolute it with the incoming audio signal. Though it was a convolution-based plugin, the user had a parametric control over the echo timing. For various echo time, instead of using different Impulse response file, the program modifies the existing Impulse response file.

The idea of creating an impulse response based on the user's requirements was interesting for me. Convolution reverbs has always been of an interest to me due to its realistic reverberation of actual spaces. But the need to change the impulse response for each and every different setting was a major drawback of convolution reverbs and another drawback was that if you want to model reverberation of different rooms, you have to record the impulse response each and every time. This could be avoided if one could synthesize the impulse response based on the room dimensions and properties. I wanted to implement the echo plugin idea on a reverb application. But it was not as easy as the previous one. The echo plugin needed just one echo to be written on the impulse response, but for a reverberation impulse response you will need thousands of echoes at different locations each with different levels.

1.2 Scope of the project

The scope of the project is to implement a 2D ray tracing algorithm in Csound to model a room in 2D and create an impulse response function table which can then be convoluted with the incoming audio signal to achieve the desired reverberation. The project is limited to 2D because the primary goal of this project is to establish the concept of ray tracing in Csound so that it can be a base for future developments. The mathematical implementation of 2D ray tracing is based on simple linear algebra concepts.

The write up gives an in-depth explanation of the algorithm and the implementation of the algorithm in Csound. The mathematics in the algorithm are simple enough to understand. By the end of this write up, the reader will have a good understanding of the approach handled in this application and a codebase to further develop upon.

1.3 Organization of the Report

Chapter 2, “Background” will help the reader to get familiar with the concept of reverberation. The chapter begins with general introduction to room reverberation and then moves on to various types of artificial reverberations. It gives the reader brief idea about convolution reverberation, impulse response and ray tracing. It also gives a brief explanation about the advantages and disadvantages of various reverberation techniques.

Chapter 3, “Explanation of the Algorithm” will explain the mathematics behind 2D ray tracing. It goes through the algorithm step by step, explaining concepts like line equation, unit circle, intersection of two lines and parametric form of line. It explains how to find the intersection of the ray with the wall, and how to find its angle after reflection.

Chapter 4, “Implementation of Algorithm in Csound” will go through the Csound code, explaining the construction of the application. It explains about the conditional statements used, user defined opcode, the process of calculating absorption, reflection, and diffusion. It gives an overview of the GUI, explaining various parameters that can be controlled.

Chapter 5, “Testing of the application” will go through analysis of various impulse responses created using the application. Both the time domain and frequency domain of the impulse response are examined and compared with impulse response taken from real spaces.

Chapter 6, “limitations and future work” is a critical overview of the algorithm and the application. It talks about the scope for future improvements and suggestions to improve the algorithm.

Chapter 7, “conclusion” talks through my thoughts about the whole project and the lessons I learnt in this creative process.

2. Background

2.1 Reverberation

Reverberation is a natural acoustic phenomenon that occurs due to reflection of sound in enclosed spaces. Since most of our surrounding reflect sound, we hear reverb on a daily basis. Since we are so used to hearing sounds that are accompanied by reverberation, any sound without reverberation will sound unnatural to us. Reverberation gives us the sense of space. It helps our ears to identify where we are spatially by perceiving this mixture of reflections. A large room will have a different reverberation compared to a small room. An open space will have very less reverberation since there are less objects to reflect upon and reach the lister back. There are different stages in reverberation.

2.1.1 Direct sound

The sound waves which travels directly from the source to the listener's ears without reflecting on any boundaries in a space are called direct sound. This is equal to sound waves traveling in free field, where no reflection occurs.

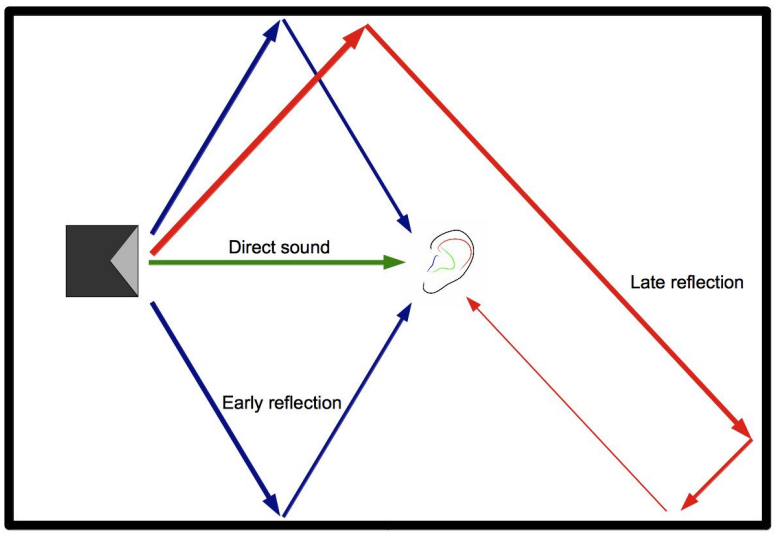

Figure 2.1 direct sound

2.1.2 Pre-Delay

The delay time between the direct sound and the first reflected sound is called Pre-delay. Pre-delay is very critical because it helps the listener in perceiving the room size.

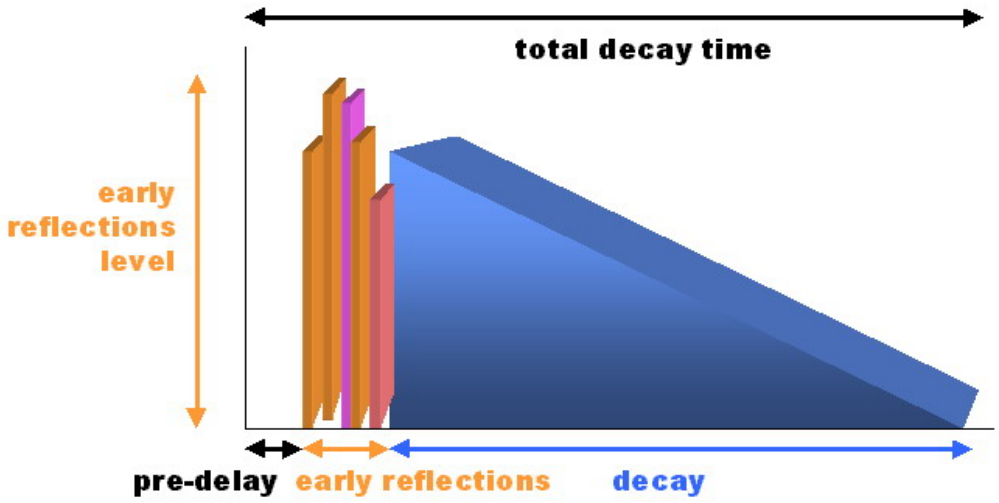

Figure 2.2 Pre-delay

2.1.3 Early Reflections

Sound waves that are bounced of the boundaries just once before reaching the listener are called Early Reflections. Shortly after the direct sound, early reflections are heard. Typically early reflections reach the listener somewhere between 5ms to 50ms. These are reflections caused by the surface of the room such as floors, walls and also by the objects that are in the room. Early reflections provide sound localization cues to the listener about the characteristics of the space. Early reflections will have less intensity compared to the direct sound. This is due to the fact that sound energy dissipates while travelling and also the surface where it gets reflects, absorbs some energy.

2.1.4 Late Reflections

Sound waves that are bounced more than once before reaching the listener are called Late reflections. These are the dense reflections that reach the listener of the early reflection. Typically late reflections reach the listener after 50ms. The intensity of the waves are further reduced due to multiple reflections.

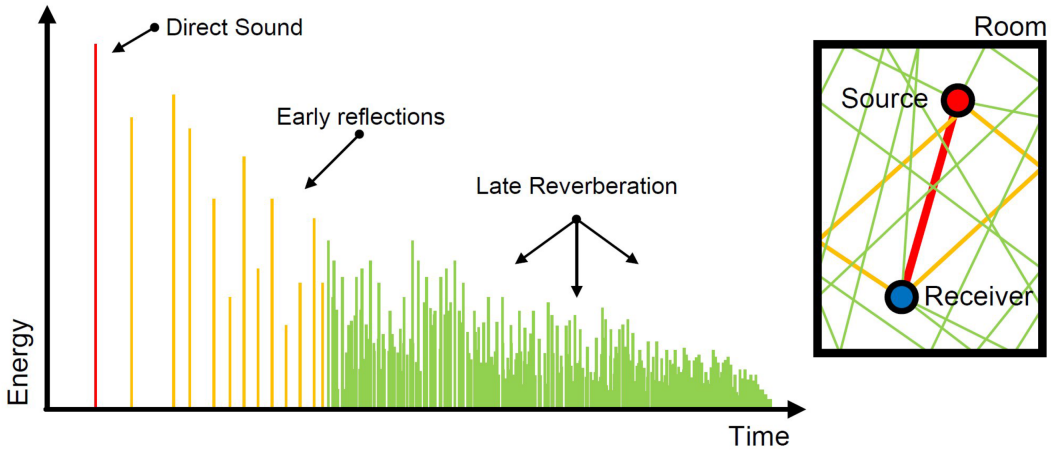

Figure 2.3 Late reflections

2.2 Different types of Artificial Reverbs

Artificial reverberation has been an important topic since the beginning of studio recording technology. Since most of the studios are acoustically dead, the recordings were extremely dry with almost negligible amount of reverberation. To counter this issue, artificial reverberation were introduced.

2.2.1 Spring Reverbs

Analogue soring reverb was one of the first artificial reverberation units. When an analogue sound signal is passed through a spring with a transducer at the other end, it added a sense of reverberation to the signal. spring reverbs were used in semi-professional recordings and are incorporated into guitar amplifiers due to their cost effectiveness and size.

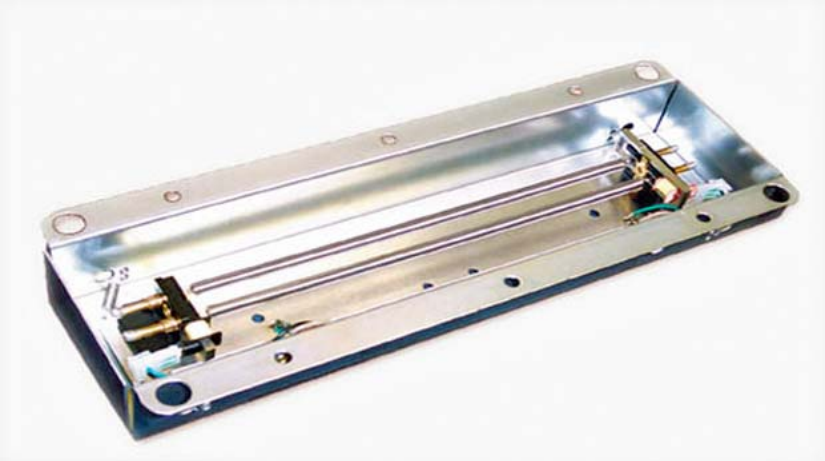

Figure 2.4 Spring reverb

2.2.2 Plate Reverbs

The first major breakthrough in artificial reverberation was the invention of plate reverb. Reverberation was added to the signal by passing it through an isolated vibrating large plate of sheet metal. A transducer picks up the vibration of the metal plate. The reverberation produced were very dense with no distinguishable early reflections. Due to this, the reverberation from the plate reverb didn't have any spatial identity. The popularity of plate reverb paved way for modern digital algorithms being written to emulate it.

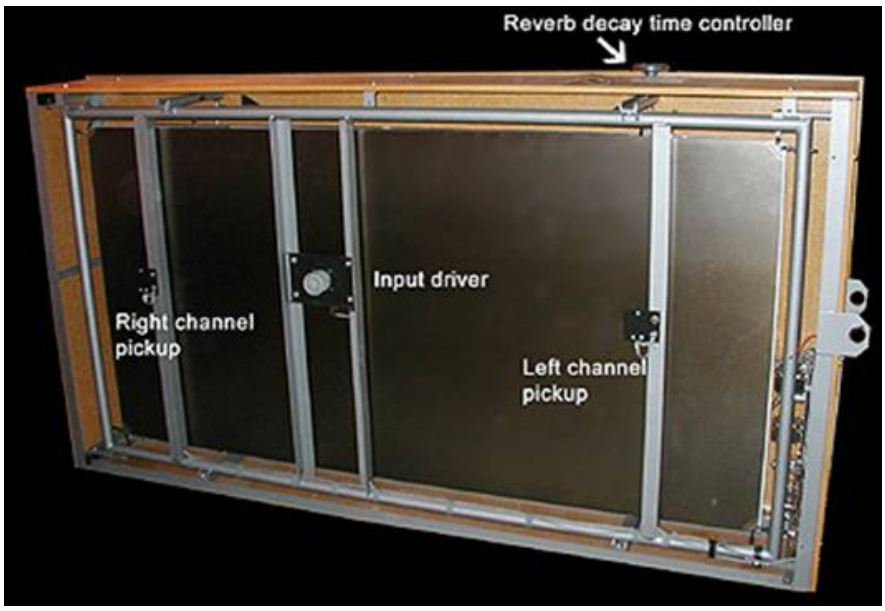

Figure 2.5 Plate reverb

2.2.3 Digital Reverbs

The advancement of science and technology in the field of digital audio resulted in the production of digital reverbs using signal processing algorithms. There are different ways to produce reverberation digitally. The most common methods are algorithmic reverbs and convolution based reverbs.

2.2.4 Algorithmic Reverbs

Algorithmic reverbs sends audio through a network of delays which attempt to imitate the interaction of sound with the room surfaces. Algorithmic reverbs are also far less computation expensive when compared to convolution reverbs. Algorithmic reverbs are highly configurable. But they do not provide the realism that can be obtained from convolution. Schroeder's reverberation algorithm was based on combination of comb filters and all pass filters. Comb filters are nothing but delays with feedback. When this delay is added with the original signal, it causes constructive and destructive interference. To get a flat frequency response, this signal is then sent into a series of all pass filters. All pass filter is a modified comb filter with a feedforward loop. Schroeder used four comb filters in parallel feeding into two all pass filters in series in his original algorithm. The limitations of Schroeder algorithm are poor response to short sounds and low echo density resulting in unrealistic reverberation.

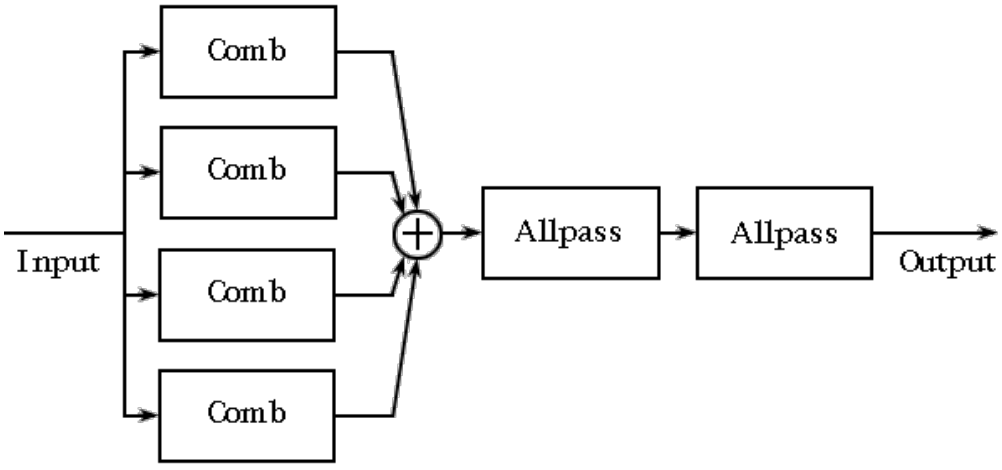

Figure 2.6 Schroeder reverb algorithm

2.2.5 Feedback Delay Network Reverbs

Feedback delay network reverbs employ parallel delays similar to that of parallel comb filters used in Schroeder reverb but in this case they employ a combined feedback signal rather than individual feedback signals around each delay unit. FND reverbs are more realistic compared to Schroeder reverb due since its echo density increases as the reverb tail progresses.

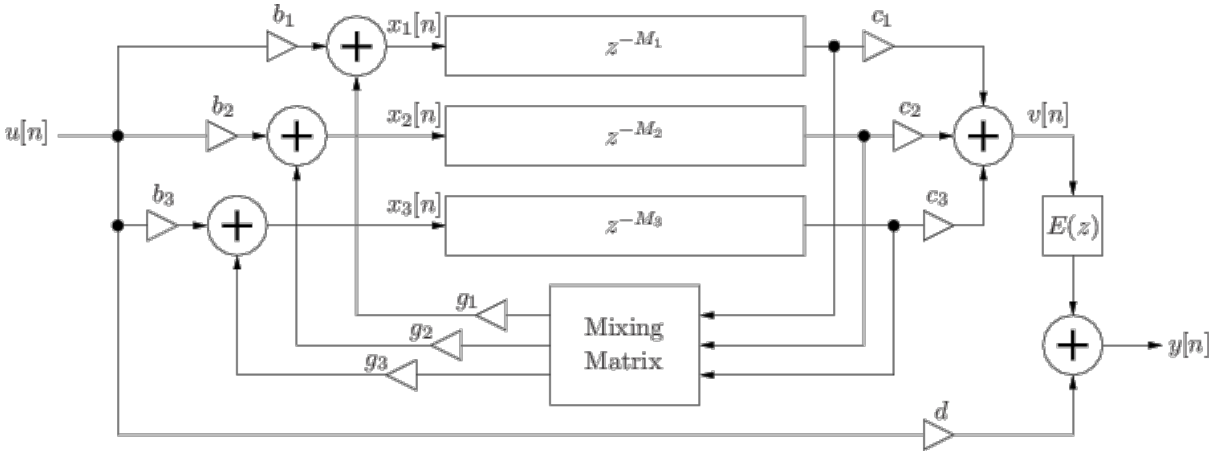

Figure 2.7 Feedback delay network structure

2.2.6 Convolution reverbs

Convolution is a digital signal processing technique which is essentially a mathematical way to combine two signals to form a third signal. convolution is basically multiplying every sample of an input signal by every sample of an impulse signal. convolution reverbs use impulse response of real spaces to apply reverb to an audio. An impulse response contains the information of how an impulse will decay in a specific space. By convolving the impulse response with an input signal, an output signal is obtained which is how the input signal would have decayed in that specific space.

2.3 Limitations of Convolution reverbs

Even though convolution reverbs are realistic and great for simulating reverberation of a real space, the impulse response is static and it is difficult to interact with the impulse response to adjust them to suit a particular requirement. It is not possible to have a control over the characteristics of the space. Latency is involved with convolution too. For every different room dimensions and characteristics, one needs a different impulse response. And to get an impulse response the user has to record the space using sine sweep or a transient. It is not possible to emulate a space if the user cannot get there physically to record the impulse response.

2.4 Ray Tracing

Ray tracing is a technique used in computer graphics and games to render an image by tracing the path of the light and then simulate the way the light interacts with the virtual objects it hits in the computer-generated space. This method produces high level of realism in computer graphics. Ray tracing of capable of simulating light properties such as reflection, refraction and diffusion. Ray tracing works by tracing a ray from an imaginary eye through each pixel in a virtual screen and the checks whether the ray intersects with any objects in the scene, if it does, then it calculates the amount of light at that point, examines the properties of the object and then with this information it calculates the final colour of the pixel.

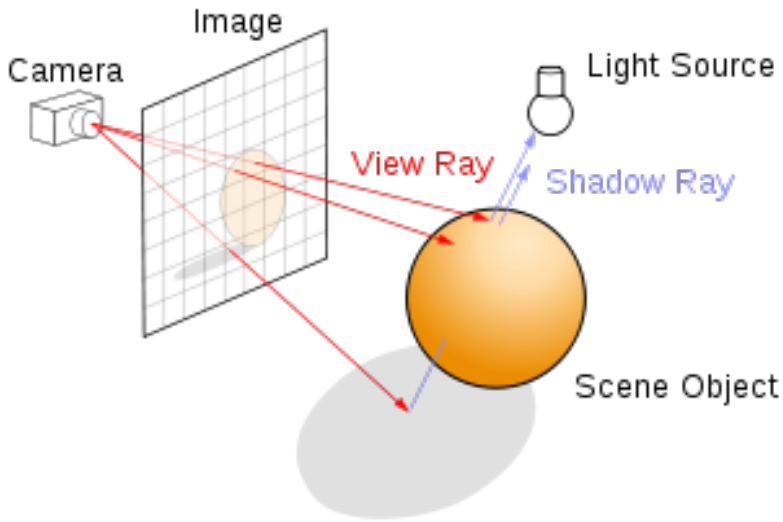

Figure 2.8 Ray tracing

3. Explanation of 2D ray tracing Algorithm

Since sound behaves more or less like light waves, we could use ray tracing method to find how sound travels within a space and also calculate its properties like reflection, absorption and diffusion. In this method sound from a source can be defined by a number of finite rays. These rays travel at the speed of sound and are reflected after hitting the room boundaries. Every time the ray hits a boundary, its intensity decreases since there is some amount of sound absorption during collision. If the ray crosses the listener, then the data from the ray is collected and used to create an impulse response.

To demonstrate this in a simpler way and also to reduce complexity, I am going to consider few things,
Implementing ray tracing in 2D and
The shape of the room is going to be a rectangle.

The most fundamental calculation for tracing a ray in 2D is ray-line intersection, since the walls are mere lines in 2D. when you want to trace a ray you need a mathematical way to answer few questions.
Where does a ray intersect the wall (lines in 2D)?
At what angle the wave reflects?
Where is the source and the listener?
And finally how far has the ray travelled before reaching the listener?
Now we will focus on the first sub problem, the ray - line intersection. We need a mathematical way to compute intersection points. To develop the maths, we first introduce a coordinate system. The maths we need comes from looking at the algebra of intersecting a ray with a line segment.
For the sake of simplicity, let us consider the length and breadth of the 2D room to be 10m.

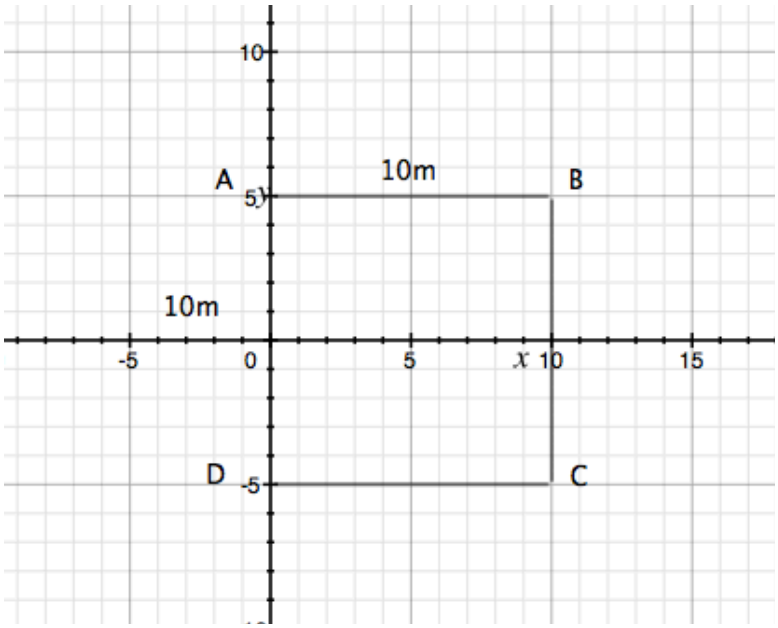

Figure 3.1 Room in 2D plane

Once we have our coordinate system, we can write our line AB as line equation. We could also use slope intercept form (y = mx + c) but since our walls are straight lines and they don't have a slope, it is better to write them as line equation to avoid division by zero.

Line equation:

$$ax + by + c = 0$$

That can be written as

$$(y_1 - y_2)x + (x_2 - x_1)y + (x_1y_2 - x_2y_1) = 0$$

When we substitute (0 , 5) and (10 , 5) to find line equation of line AB, we get

$$y = 5$$

Subsequently you can find the line equation of the remaining lines BC, CD and DA using the same equation.

AB : $y = 5$ BC : $x = 10$ CD : $y = -5$ DA : $x = 0$

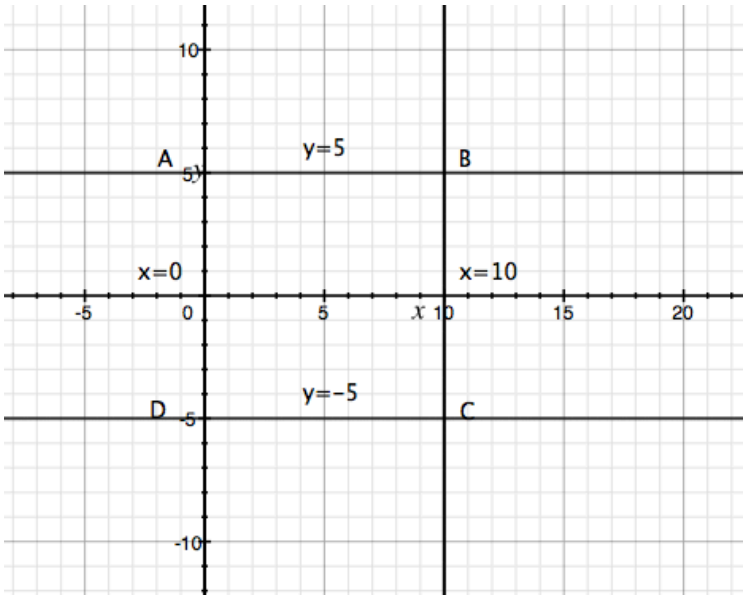

Figure 3.2 Line equations

Since we have the line equation of the four walls, now we need to find the line equation of the ray so that we can find the intersection point using these two equation.

Before we start, we have to get the coordinates of our source. Let us consider our source as origin i.e. (0 , 0) for now. And the angle at which the ray travels as 45°.

We need at least two points in the ray to write a line equation. All we got now is one point and an angle. To solve this issue, we could use Unit circle. The unit circle is nothing but a circle with a radius of 1 unit.

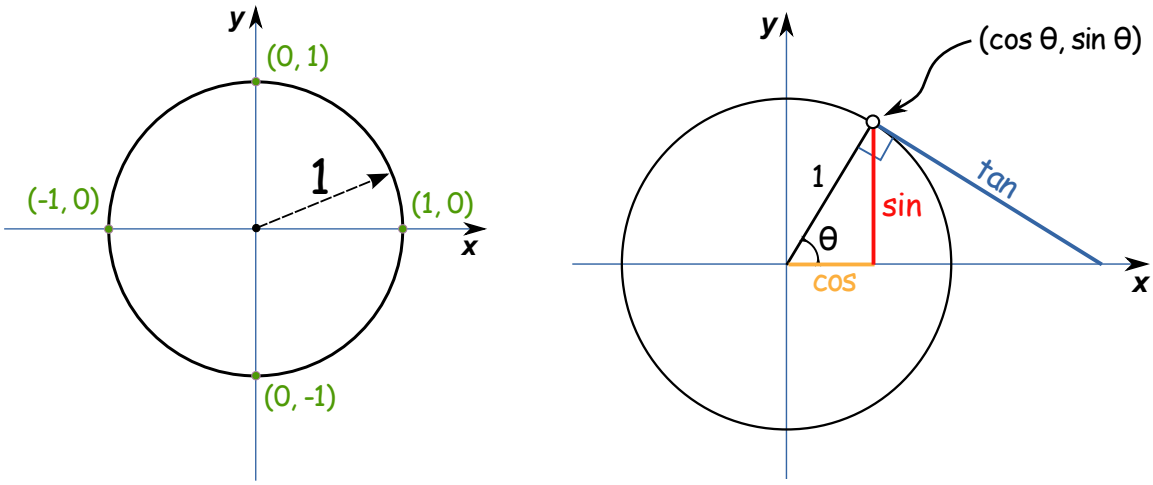

Figure 3.3 Unit circle

Since we know the θ and the hypotenuse which is 1, we can find the second point using the formula: (cos θ, sin θ)

Also, to find unit circle point from any other point (x, y) other than the origin,

the formula is (x + cos θ , y + sin θ)

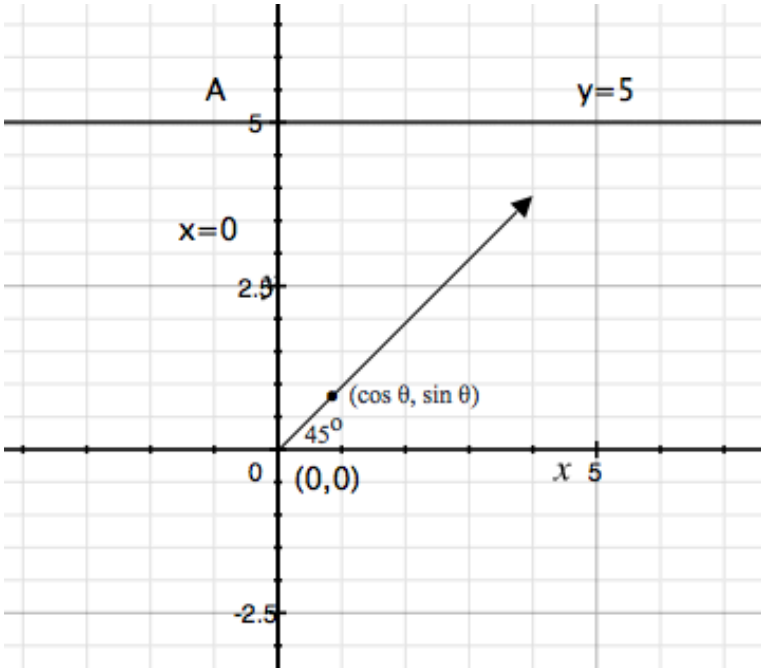

Figure 3.4 Ray

Now we have two points:

The origin of the ray, which I am going to refer as $(C_x , C_y) = (0 , 0)$ and,

The unit circle point, which I am going to refer as $(U_x , U_y) = (\cos \theta, \sin \theta)$

With two points now we can derive the line equation of the ray.

To represent our ray CU, I am going to use something called parametric function.

The ray will be represented by a new function R(t) which is a weighted average of the points C and U, where t is the weighted average of C and U.

$$R(t) = (1 - t)C + tU$$

Notice when t = 0, R(0) = C and when t = 1, R(1) = U and when t is between 0 and 1, R will be a point somewhere in the line between C and U.

Values of t greater than 1 will be points on the ray beyond U.

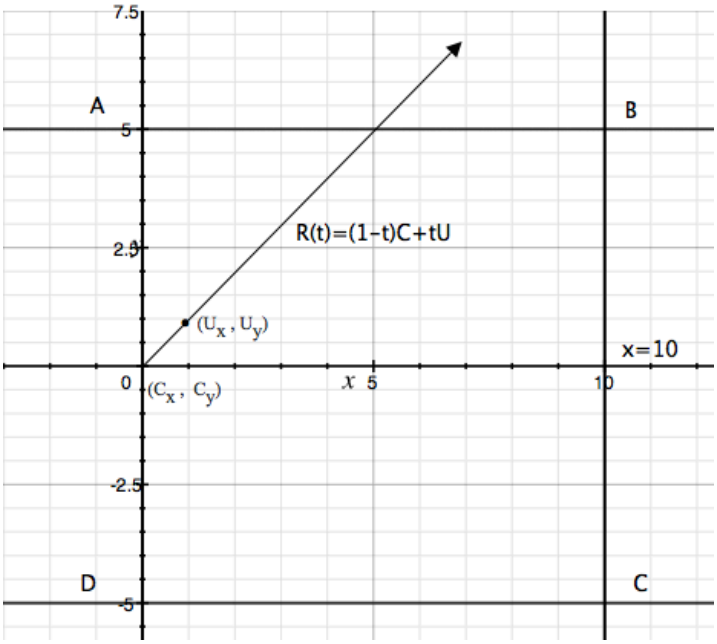

Figure 3.5 Parametric form of ray

Now we can calculate the intersection point (I_x , I_y) between our ray CU and the line AB.
We know that different values of t will locate different points on the ray and the intersection point we are looking for is one such point on the ray. So there must be some value of t such that the intersection point I = R(t)
This is two equations, one for the x coordinate of I and one for the y coordinate of I.

$$I_x = R_x(t) = (1-t)C_x + tU_x$$
$$I_y = R_y(t) = (1-t)C_y + tU_y$$

The point (I_x , I_y) we're looking for is on both of these lines. From the AB line equation we know that any point on the line AB will have y value 5.

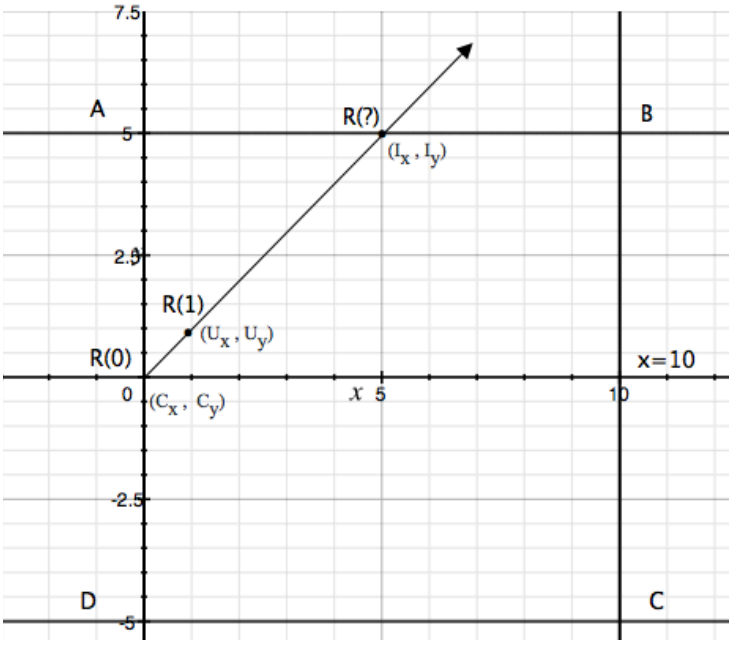

Figure 3.6 Intersection point

Since the point (I_x , I_y) is also on the line AB, I_y should also be equal to 5.

Know we know I_y, C_y and U_y, we can find t by substituting them in the equation,

$$I_y = R_y(t) = (1 - t)C_y + tU_y$$

where $$t = (I_y - C_y)/(U_y - C_y)$$

once we find t, we can substitute that in this equation to find I_x.

$$I_x = R_x(t) = (1 - t)C_x + tU_x$$

Thus we calculated the intersection point (I_x , I_y) of the ray and the line AB.

Once we have our intersection point, we know that the ray gets partially absorbed and partially reflected. We will look at the absorption factor later on and will focus on the reflected ray.

We know that for any specular reflection, the angle of incidence is equal to angle of reflection and also that alternate angles between two parallel lines are equal. From this we can conclude that the reflected ray will travel at an angle 360 - θ.

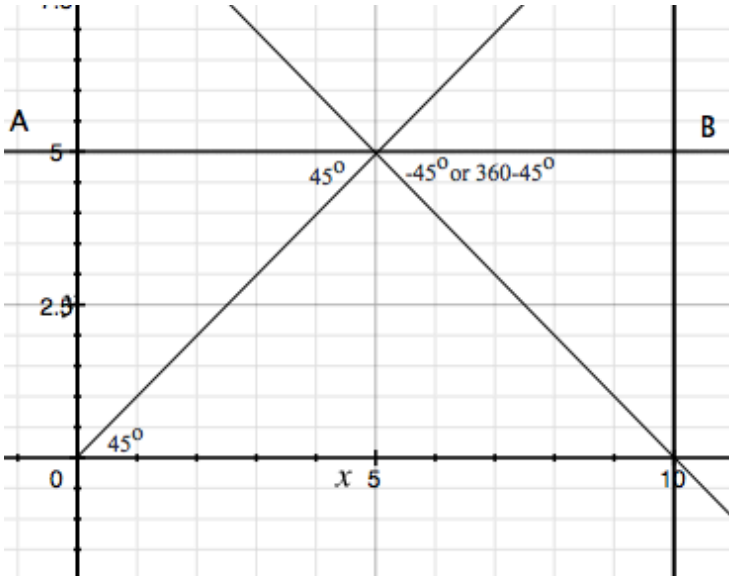

Figure 3.7 Angle of reflection

The origin of our reflected ray is going to be the intersection point (I_x , I_y). now we have a point and an angle. As we did earlier, we can find the unit circle point using ($x + \cos\theta$, $y + \sin\theta$)

We can find the intersection point of the ray with all the four lines using this method by substituting their respective line equation.

4. Implementation of Algorithm in Csound

4.1 Explanation of the Build

This 2D ray tracing-based reverberation application was built in Csound with the help of Csound frontend CsoundQt. CsoundQt helps in building the GUI for this application. To run this application, you need to install both Csound and CsoundQt. The link for downloading these softwares are provided in the References section. The link to the GitHub Repository for this project is also provided in the reference section. To run this application all you need to do is download and install both Csound and CsoundQt and run the .csd file.

4.2 Explanation of Csound code

We first start by creating an empty function table to write the impulse data. This function table (gicon) acts as our impulse response.

```
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 32
nchnls = 2
0dbfs = 1

gicon   ftgen     1, 0, -sr, 2, 0
giblank ftgen     2, 0, -sr, 2, 0

seed 0
```

Since we know the mathematical algorithm to trace a single ray, we could make Csound do the same calculations for any number of rays using recursion. To implement recursion, we have to create a User defined Opcode so that the UDO can call itself recursively.

```
sr = 44100
ksmps = 32
nchnls = 2
0dbfs = 1

gicon   ftgen     1, 0, -sr, 2, 0
giblank ftgen     2, 0, -sr, 2, 0

seed 0

opcode raytrace, 0, iiiiiiiiop

ixcod, iycod, ilength, ibreadth, iraydensity, ireflectionorder, idiffusion, iabsorption, icount, icount2  xin
```

We then find the line equation of the ray using origin (icx, icy), unit circle (iux, iuy) and the degree ideg.

```
ideg = (icount+0.1)/idivider

irad = ideg*($M_PI)/180

tableiw  1, 0, gicon

icx = ixcod
icy = iycod

iux = icx+cos(irad)
iuy = icy+sin(irad)

iraytotallength = 0
irefelections = 1
```

The line equation is not limited to any length. so the ray's line equation will intersect all the four boundaries at some point in the coordinate system, Except when the ray is parallel to a line.
To find just the boundary in which our ray hits, we could use few conditional statements with certain conditions.

First,

- Calculate intersection point using the boundary's line equation

```
icx = ixcod
icy = iycod

iux = icx+cos(irad)
iuy = icy+sin(irad)

iraytotallength = 0
irefelections = 1

iy = ibreadth/2

iiy = iy

it = (iiy-icy)/(iuy-icy)

iix = (1-it)*icx + it*iux
```

- Check whether the intersection point is on the boundary using its length and also check the angle of the ray to determine whether the ray is moving towards the boundary.

```
it = (iiy-icy)/(iuy-icy)

iix = (1-it)*icx + it*iux

if ((iix >= 0) && (iix <= ilength)) && ((ideg > 0) && (ideg < 180)) then
```

- If the condition is satisfied, then calculate the distance travelled by the ray, find the angle of the reflected ray, make intersection point as the source of the reflected ray. And start the process again.

```
if ((iix >= 0) && (iix <= ilength)) && ((ideg > 0) && (ideg < 180)) then

ia = iix-icx
ib = iiy-icy

iraylength = sqrt((ia*ia)+(ib*ib))

iraytotallength += iraylength

irefelections += 1

iran random 1, idiffusion

ideg = 360-ideg + iran

irad = ideg*($M_PI)/180

icx = iix
icy = iiy

iux = icx+cos(irad)
iuy = icy+sin(irad)
```

- If the condition is not satisfied, repeat the same process on other boundaries to find the intersection point.

The number of loops (reflections) can be controlled using the application's GUI.

To find whether the ray crosses the listener, we use another conditional statement. And if it satisfies the condition, then the total distance travelled by the ray is calculated. The time it took for the ray to reach the listener from the source can be calculated using the distance. This is the delay time of that echo. Using the delay time, we can determine the index at which the echo impulse has to be written.

```
if (iiy >= -0.1) && (iiy <= 0.1) then

ia = iix-icx
ib = iiy-icy

iraylength = sqrt((ia*ia)+(ib*ib))

iraytotallength += iraylength

itime = iraytotallength/340

iindex = 44100 * itime

iindex = int (iindex)
```

Two factors that has to be taken into consideration before writing a value into the function table are the sign and the intensity of the echo.

Every time the ray bounces of a boundary, its intensity reduces due to absorption. And the amount of absorption can be controlled using the GUI.

```
iindex = int (iindex)
ivalue = 1-(iabsorption*irefelections)
ivalue = ivalue >= 0 ? ivalue : 0
ifraction = frac (irefelections/2)
ivalue = ifraction = 0 ? ivalue : -ivalue
tableiw ivalue, iindex, gicon
```

This whole process is repeated recursively using different rays. And then the function table is convoluted with the incoming audio signal using ftconv opcode.

4.3 GUI of the Application

The application window is populated with the following widgets:

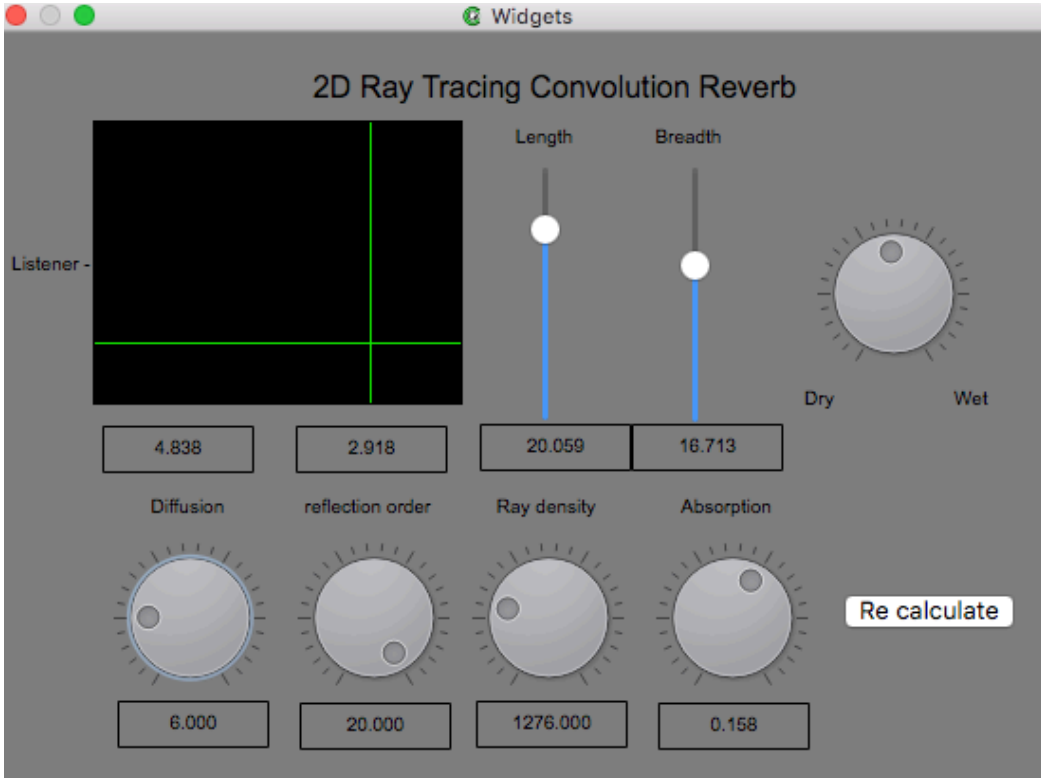

Figure 4.1 GUI

Length and breadth sliders:

The user can choose the length and breadth of the 2D room using these sliders. The length and breadth are in meters. The user can choose between 2m to 25m.

XY crosshair controller:

The position of the source inside the room can be chosen using this XY crosshair controller. The two displays below the controller displays the x and y position of the source in the coordinate system. The listener marker indicates the point of the listener which is at (0, 0) in the coordinate system.

Diffusion:

This knob lets the user control the amount of diffusion that happens when the ray hits the boundaries. It ranges from 0 degree to 30 degree. Diffusion is randomized, but this controls the amount of randomization.

Reflection Order:

This knob lets the user control the number of reflections a ray goes through before it dies. Higher the reflection order, more the probability of the ray reaching the listener. The user can choose between 1 to 20 reflections per ray.

Ray density:

This knob lets the user control the number of rays to be produced during the ray tracing. The user can choose between 100 to 5000 rays. Higher the ray density, more the resolution of the impulse response.

Absorption:

This knob lets the user choose the amount of absorption that happens during each reflection. The absorption value ranges from 0.025 to 0.25. It is more like choosing the material and surface of the wall. Different material will have different absorption coefficients.

Re calculate:

Every time the user changes any parameter in the GUI, It doesn't change the impulse response. After choosing all the desired parameters the user then have to press re calculate button to

create a new impulse response. Once this button is pressed, the ray tracing process starts again with the new parameters and creates an impulse response.

Dry Wet:

This knob lets the user control the mix between the dry signal and the reverberated signal.

5. Testing of the Application

This chapter will go through analysis of various impulse responses created using the application. Both the time domain and frequency domain of the impulse response will be examined. Audacity is used for viewing the time domain representation of the impulse response and Sonic Visualiser is used for viewing the frequency domain representation of the impulse response. The impulse files are created by sending an impulse into the function table and recording the output audio in Csound. Since there are numerous possible scenarios, we are going to look at only few. The testing is leaned towards finding the limitations of the application by using extreme values.

5.1 Length and breadth of the room

In this test, we will change the length and breadth of the room while keeping other parameters constant. For maximum resolution, let the ray density be 5000.

Length: 15.2m Breadth: 5.8m

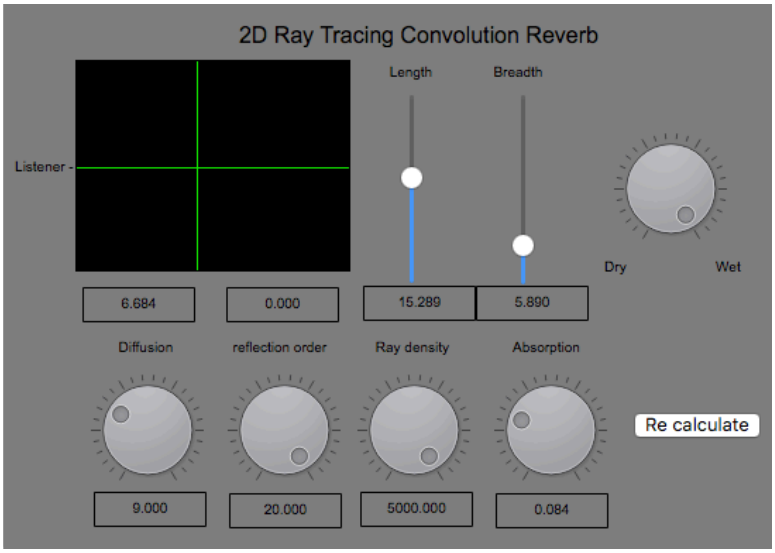

Figure 5.1 testing GUI 1

Impulse response waveform:

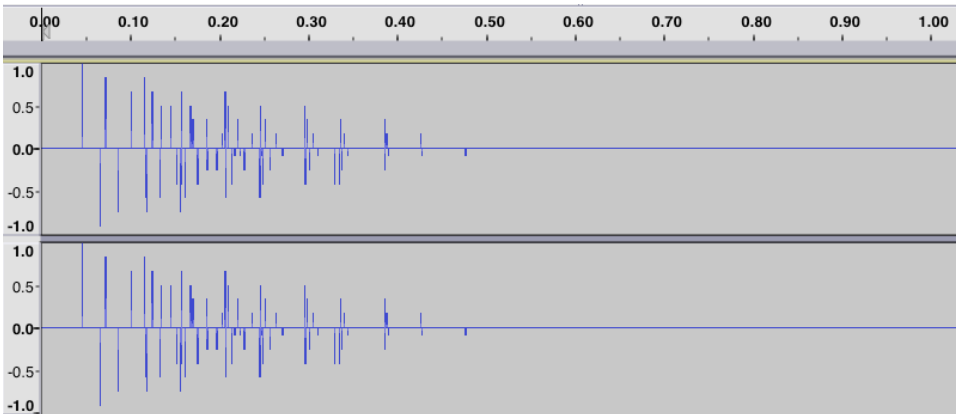

Figure 5.2 Waveform of IR 1

Spectrogram of Impulse response:

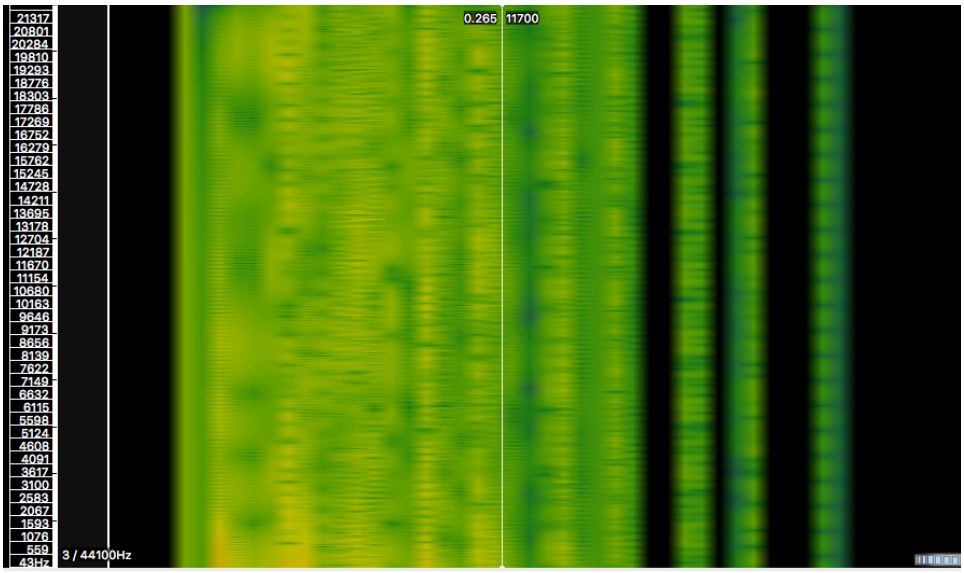

Figure 5.3 spectrogram of IR 1

When we look at the time domain representation, the reverberation time is around 0.45sec, which is close to estimated RT60 of a room with similar dimensions. The echoes are in quite good numbers. Decaying of the echoes are well represented. The frequency domain

representation shows that all the frequencies are covered. but unlike impulse response of real spaces, the higher frequencies are not attenuated faster than the lower frequencies. I think it is not able to represent standing modes and other effects which are observed at low frequencies. I also feel that the density of the late reflections can be improved largely if we could ray trace with even larger ray density.

Length: 25m Breadth: 25m

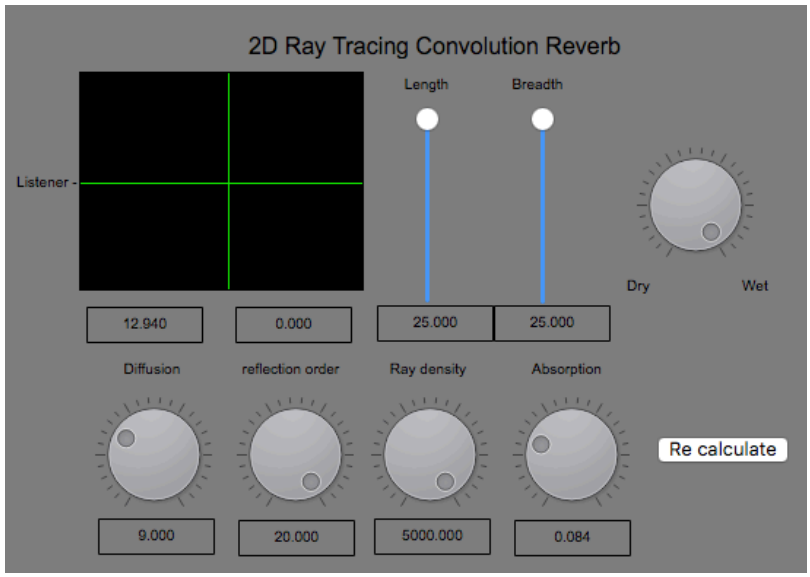

Figure 5.4 GUI 2

Impulse response waveform:

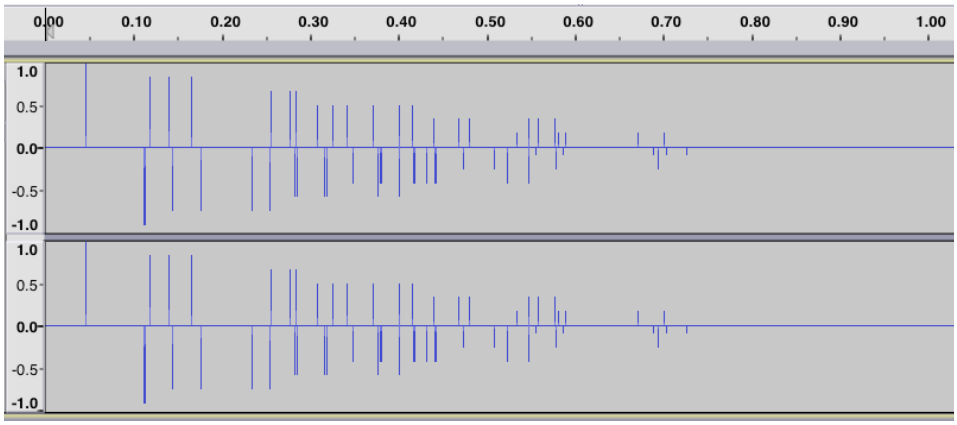

Figure 5.5 waveform of IR 2

Spectrogram of Impulse response:

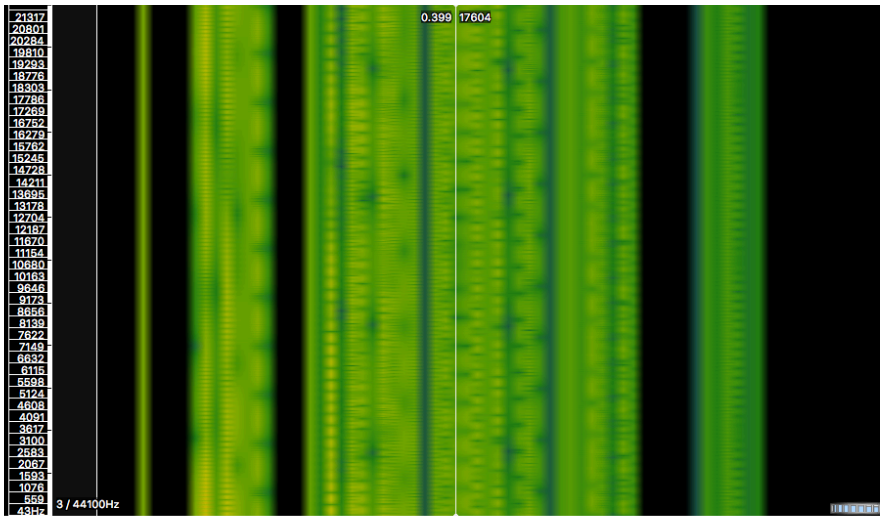

Figure 5.6 Spectrogram of IR 2

When we change the length and breadth to the maximum of 25m, we could see that the reverberation time has increased. But the distribution of echoes are not entirely uniform. Since the early reflections echoes are very few in number, there are gaps in the frequency representation. This also creates distinct echoes when convoluted with an audio signal rather than a reverberation. When I tried to find the reason for this, I found that even though there are so many early reflection rays reaching the listener, they all gets written in the same index.

```
instr 1:  iindex = 14787.000
instr 1:  iindex = 14784.000
instr 1:  iindex = 36477.000
instr 1:  iindex = 14796.000
instr 1:  iindex = 33947.000
instr 1:  iindex = 53935.000
instr 1:  iindex = 34080.000
instr 1:  iindex = 34080.000
instr 1:  iindex = 40567.000
instr 1:  iindex = 1433.000
instr 1:  iindex = 1433.000
instr 1:  iindex = 1433.000
instr 1:  iindex = 1433.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 1434.000
instr 1:  iindex = 33908.000
instr 1:  iindex = 7933.000
instr 1:  iindex = 47537.000
instr 1:  iindex = 54314.000
instr 1:  iindex = 53921.000
```

Figure 5.7 Csound console

The reason behind this is, since the time difference between these rays are very small, when calculating the index number using the delay time, it results in index numbers where the differences are in few decimal points. Since the index number has to be an integer, the index

number values are converted from float to integer, and due to this, all the index numbers ends up being the same. Using higher sample rate might be a solution to this issue.

5.2 Reflection Order

In this test we will try creating two impulse responses, one with maximum reflections (20 reflections) and one with less reflections (4 reflections). Will keep the source in the centre of the room, length at 15m and breadth at 6m. we will keep other parameters in optimal level.

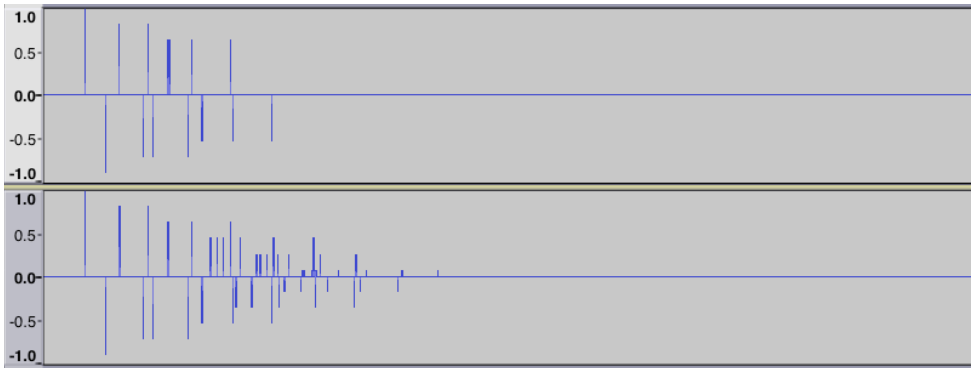

Figure 5.8 Waveform of IR 3

As we can see, the early reflection echoes are intact since reflect of the boundaries just once. The late reflections are not present since all the rays ceased to exist after 4 reflections. Nonuniformity is still present in the early reflections. That is a fundamental issue that has to be addressed in the upcoming versions.

5.3 Absorption and ray density

In this test we will create four impulse responses, one with absorption coefficient of 0.025 and the other one with absorption coefficient of 0.2. Just like the previous test, will keep the source in the centre of the room, length at 15m and breadth at 6m. will keep other parameters in optimal level. And the other two impulse responses with ray density 500 and 5000, while keeping other parameters in an optimal level.

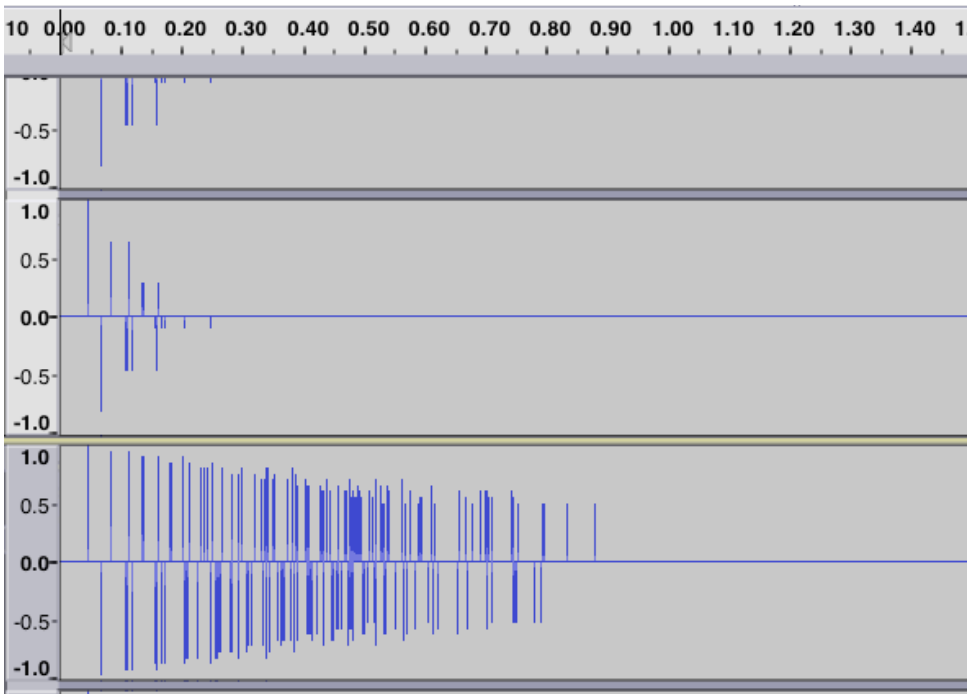

Figure 5.9 Waveform of IR 4 & 5

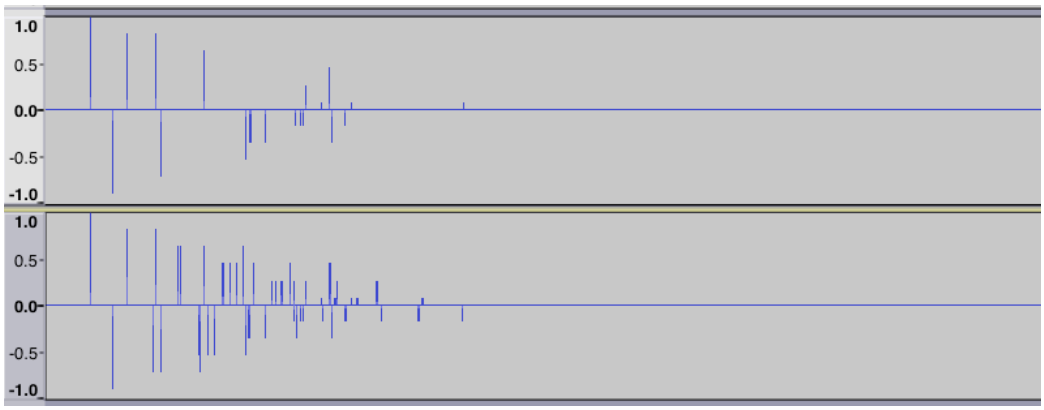

Figure 5.10 Waveform of IR 6 & 7

It is very evident from the impulse response that with high absorption coefficient, the echoes dies of quickly within few reflections. But in real world spaces, many materials like carpet, hardwood, etc has absorption coefficient around 0.2, but an impulse response taken in a room made up of hardwood would not be similar to this. That is due to the fact that we are trying to emulate sound waves which are infinite with finite number of rays. On the other hand, the impulse response created using 0.025 absorption coefficient doesn't has decaying reverb tail.

And that is due to the fact that the number of reflections are limited to 20. But in real space, the sound waves die of only when the intensity reaches zero. If the program is capable of performing higher number of reflections, then a smooth reverb tail is possible with low absorption coefficient.

And as we expected, with low ray density, the resolution of the impulse response is very low. With low ray density, only fewer rays reach the listener.

5.4 Ray tracing reverb vs Csound's reverb

This is a general comparison between our application and the Csound's reverb opcode. The Csound reverb opcode is based on the Schroeder's reverb algorithm. It is made up of four comb filters in parallel which then feeds into two all pass filters in series. The impulse response file is created by sending in one sample impulse through the reverb opcode with 1 sec delay time and recording the output audio. For comparison I am using the impulse file created with length 15m and breadth 6m.

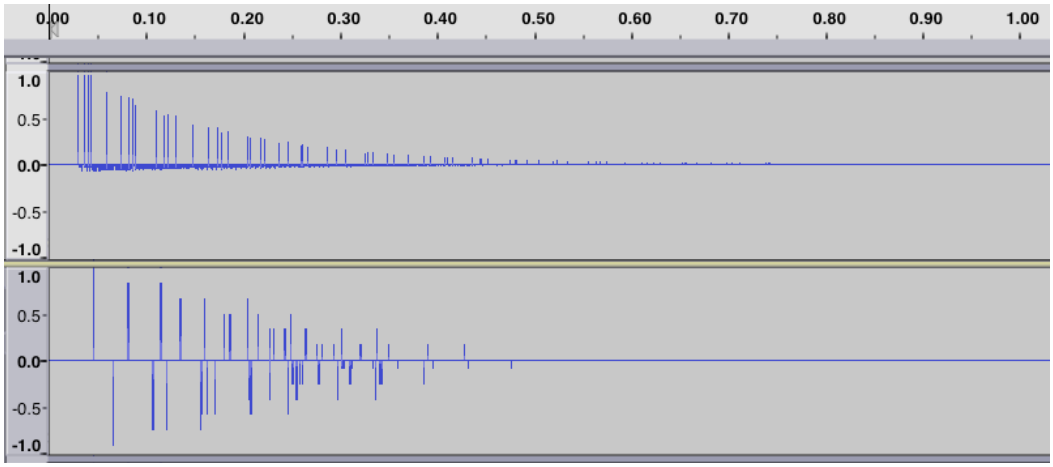

Figure 5.11 Waveform of IR 8 & 9

As we can see, the echo density of both the outputs are very similar. The Csound reverb exhibits Echoes only in the positive domain, this might be due to the fact that a single sample impulse was used to get the output. The key differences are that the Csound reverb has more number of early reflections and has a smooth reverb tail. Solutions to these two problems will result in a better reverberation unit.

6. Limitations and future works

This section talks about the limitations of 2D ray tracing and illustrates methods and ideas to improve the application in future.

6.1 Limitations

There are quite a few limitations in regards to ray tracing method. The frequency and wavelength of the sound waves are not factored into the algorithm. Sound waves are being considered as just plain rays and not a mixture of different waves with different frequency. The different natures of high frequency and low frequency were not considered. In a real space, higher frequencies attenuate much faster than lower frequencies. Effects like wave interference (causes standing waves between reflective parallel walls) and diffraction are not modelled. These things can be factored in only when we consider sound as waves and not as rays.

Nevertheless ray tracing is still efficient when compared to other methods for modelling a space. As I said earlier, This application is limited to 2D because the primary goal of this project is to establish the concept of ray tracing in Csound so that it can be a base for future developments. And the mathematics behind 2D ray tracing is not a complex one. Due to the fact that it is in 2 dimensions, there were many assumptions and limitations. The aim of this project was not to achieve a perfect reverberation unit, but rather to build an application that shows promising results. The 2D ray tracing method does shows promising results. Considering a 3 dimension room as a 2 dimensional plane has its own price to pay. Implementing 3D ray tracing would be computationally expensive but will yield excellent results. Another limitation was the shape of the room, for the sake of simplicity, the shape of the room was limited to rectangle. These limitations can be overcome in future versions.

Another issue was, for ray density greater than 5000, Csound crashes during initialisation. Even for a 2D model, ray density greater than 10000 will result in better impulse response. Implementing this algorithm in 3D might give us some solution for the fewer number of early reflections in our impulse responses.

6.2 Future works

Further improvements can be made on the existing 2D algorithm if we could compute large number of rays. But a better option is to implement the existing algorithm in 3D. The mathematics behind it is pretty much the same as 2D. There will be additional Z points. Instead of finding intersection of two lines, we have to find intersection of a line and a plane. Since height factor comes into play in 3D, the resolution of the impulse response increases greatly. 3D ray tracing can be implemented efficiently using vectors. There is a collection of opcodes in Csound to perform linear algebra operations, from scalar, vector, and matrix arithmetic up to and including QR based eigenvalue decompositions.

7. Conclusion

This research has been an extensive learning curve for me in exploring the concepts of Ray tracing, reverberation and acoustic room modelling. In the end, I am happy that I was successful in what I set out to do. I strongly believe that this project can be foundation for other creative minds to work and improve upon. In the future I hope to expand my horizon in these concepts in order to further develop around this idea.

Appendix A

Csound Source code

```
<CsoundSynthesizer>
<CsOptions>
-odac
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 32
nchnls = 2
0dbfs = 1

;function table for the IR

gicon   ftgen     1, 0, -sr, 2, 0
giblank ftgen     2, 0, -sr, 2, 0

seed 0

opcode raytrace, 0, iiiiiiiiop

ixcod, iycod, ilength, ibreadth, iraydensity, ireflectionorder, idiffusion, iabsorption, icount, icount2  xin

;idivider to divide 0 - 360 degree equally based on the desired ray density

idivider =  iraydensity/358

ideg = (icount+0.1)/idivider

irad = ideg*($M_PI)/180

tableiw  1, 0, gicon

;icx and icy are the coordinates for the origin of the ray

icx = ixcod
icy = iycod

;iux and iuy are the coordinates for the unit cirlce drawn from the point of origin of the ray

iux = icx+cos(irad)
iuy = icy+sin(irad)

iraytotallength = 0
irefelections = 1

;line equation of the north boundary

iy = ibreadth/2

;iix and iiy are the intersection point of the ray and the boundaries

iiy = iy

it = (iiy-icy)/(iuy-icy)

iix = (1-it)*icx + it*iux

;check whether the intersection point is whithin the boundary length

if ((iix >= 0) && (iix <= ilength)) && ((ideg > 0) && (ideg < 180)) then

ia = iix-icx
ib = iiy-icy

;pythagorean theorem to calculate the length of the ray

iraylength = sqrt((ia*ia)+(ib*ib))

iraytotallength += iraylength

irefelections += 1

; to randomize the reflected ray's angle

iran random 1, idiffusion

; angle of the reflected ray

ideg = 360-ideg + iran

irad = ideg*($M_PI)/180

;the point of intersection is the point of origin for the new reflected ray

icx = iix
icy = iiy

;coordinates of the unit circle drawn from the origin of the reflected ray

iux = icx+cos(irad)
iuy = icy+sin(irad)
```

```
else

;line equation of the east boundary,  repeat the whole process to find intersection point

ix = ilength

iix = ix

it = (iix-icx)/(iux-icx)

iiy = (1-it)*icy + it*iuy

if ((iiy >= -ibreadth/2) && (iiy <= ibreadth/2)) && (((ideg >= 0) && (ideg < 90)) || ((ideg > 270) && (ideg <= 360))) then

ia = iix-icx
ib = iiy-icy

iraylength = sqrt((ia*ia)+(ib*ib))

iraytotallength += iraylength

irefelections += 1

iran random 1, idiffusion

ideg = 360-ideg + iran

irad = ideg*($M_PI)/180

icx = iix
icy = iiy

iux = icx+cos(irad)
iuy = icy+sin(irad)

else

;line equation of the south boundary,  repeat the whole process to find intersection point

iy = -ibreadth/2

iiy = iy

it = (iiy-icy)/(iuy-icy)

iix = (1-it)*icx + it*iux

if ((iix >= 0) && (iix <= ilength)) && ((ideg > 180) && (ideg < 360)) then


ia = iix-icx
ib = iiy-icy

iraylength = sqrt((ia*ia)+(ib*ib))

iraytotallength += iraylength

irefelections += 1

iran random 1, idiffusion

ideg = 360-ideg + iran

irad = ideg*($M_PI)/180

icx = iix
icy = iiy

iux = icx+cos(irad)
iuy = icy+sin(irad)

else

;line equation of the west boundary,  repeat the whole process to find intersection point

ix = 0.0

iix = ix

it = (iix-icx)/(iux-icx)

iiy = (1-it)*icy + it*iuy

if ((iiy >= -ibreadth/2) && (iiy < -0.1) || (iiy > 0.1) && (iiy <= ibreadth/2)) && ((ideg > 90) && (ideg < 270)) then

ia = iix-icx
ib = iiy-icy

iraylength = sqrt((ia*ia)+(ib*ib))

iraytotallength += iraylength

irefelections += 1

iran random 1, idiffusion

ideg = 360-ideg + iran

irad = ideg*($M_PI)/180

icx = iix
icy = iiy

iux = icx+cos(irad)
iuy = icy+sin(irad)

else

;since the listener lies on the west boundary, check whether the ray hits the listener

if (iiy >= -0.1) && (iiy <= 0.1) then

ia = iix-icx
ib = iiy-icy
```

```
iraylength = sqrt((ia*ia)+(ib*ib))
;calculates the total distance the ray has travelled before reaching the listener
iraytotallength += iraylength
;calculates the time it took to reach the listener
itime = iraytotallength/340
;based on the time, calculates the index on which the echo has to be written
iindex = 44100 * itime
iindex = int (iindex)
;based on the no of reflections, the amount of sound energy absorption is calculated
ivalue = 1-(iabsorption*irefelections)
ivalue = ivalue >= 0 ? ivalue : 0
;based on the no of reflections, the phase of the reflected ray is calculated
ifraction = frac (irefelections/2)
ivalue = ifraction = 0 ? ivalue : -ivalue
;writes the value in the ftable
tableiw ivalue, iindex, gicon
igoto loopend
endif
endif
endif
endif
endif

;loop the whole process to calculate consecutive reflections
while icount2 < ireflectionorder do
iy = ibreadth/2
iiy = iy
it = (iiy-icy)/(iuy-icy)
iix = (1-it)*icx + it*iux
if ((iix >= 0) && (iix <= ilength)) && ((iix != icx) && (iiy !=icy)) then
ia = iix-icx
ib = iiy-icy
iraylength = sqrt((ia*ia)+(ib*ib))
iraytotallength += iraylength
irefelections += 1
iran random 1, idiffusion
ideg = 360-ideg + iran
irad = ideg*($M_PI)/180
icx = iix
icy = iiy
iux = icx+cos(irad)
iuy = icy+sin(irad)
else
ix = ilength
iix = ix
it = (iix-icx)/(iux-icx)
iiy = (1-it)*icy + it*iuy
if ((iiy >= -ibreadth/2) && (iiy <= ibreadth/2)) && ((iix != icx) && (iiy !=icy)) then
ia = iix-icx
ib = iiy-icy
iraylength = sqrt((ia*ia)+(ib*ib))
iraytotallength += iraylength
irefelections += 1
iran random 1, idiffusion
ideg = 360-ideg + iran
irad = ideg*($M_PI)/180
icx = iix
icy = iiy
iux = icx+cos(irad)
iuy = icy+sin(irad)
else
iy = -ibreadth/2
iiy = iy
```

```
it = (iiy-icy)/(iuy-icy)

iix = (1-it)*icx + it*iux

if ((iix >= 0) && (iix <= ilength)) && ((iix != icx) && (iiy !=icy)) then


ia = iix-icx
ib = iiy-icy

iraylength = sqrt((ia*ia)+(ib*ib))

iraytotallength += iraylength

irefelections += 1

iran random 1, idiffusion

ideg = 360-ideg + iran

irad = ideg*($M_PI)/180

icx = iix
icy = iiy

iux = icx+cos(irad)
iuy = icy+sin(irad)

else


ix = 0.0

iix = ix

it = (iix-icx)/(iux-icx)

iiy = (1-it)*icy + it*iuy

if ((iiy >= -ibreadth/2) && (iiy < -0.1) || (iiy > 0.1) && (iiy <= ibreadth/2)) && ((iix != icx) && (iiy !=icy)) then

ia = iix-icx
ib = iiy-icy

iraylength = sqrt((ia*ia)+(ib*ib))

iraytotallength += iraylength

irefelections += 1

iran random 1, idiffusion

ideg = 360-ideg + iran

irad = ideg*($M_PI)/180

icx = iix
icy = iiy

iux = icx+cos(irad)
iuy = icy+sin(irad)

else

if (iiy >= -0.1) && (iiy <= 0.1) then

ia = iix-icx
ib = iiy-icy

iraylength = sqrt((ia*ia)+(ib*ib))

iraytotallength += iraylength

itime = iraytotallength/340

iindex = 44100 * itime

iindex = int (iindex)

ivalue = 1-(iabsorption*irefelections)

ivalue = ivalue >= 0 ? ivalue : 0

ifraction = frac (irefelections/2)

ivalue = ifraction = 0 ? ivalue : -ivalue

tableiw ivalue, iindex, gicon

igoto loopend

endif
endif
endif
endif
endif

icount2 += 1

od


loopend:

;call the UDO recursively to trace all the rays

  if icount <iraydensity then
      raytrace ixcod, iycod, ilength, ibreadth, iraydensity, ireflectionorder, idiffusion, iabsorption, icount+1, 1
  endif


endop
```

```
instr 1

tableicopy gicon, giblank

ixcoordinate invalue "x"
iycoordinate invalue "y"
ilength  invalue "length"
ibreadth invalue "breadth"
iraydensity    invalue "raydensity"
idiffusion     invalue "diffusion"
iabsorption    invalue "absorption"
ireflectionorder invalue "reflection"
ireflectionorder = int (ireflectionorder)

ixcod = ixcoordinate*ilength
iycod = iycoordinate*ibreadth

raytrace ixcod, iycod, ilength, ibreadth, iraydensity, ireflectionorder, idiffusion, iabsorption


endin


instr 2

kxcoordinate invalue "x"
kycoordinate invalue "y"
klength  invalue "length"
kbreadth invalue "breadth"

kxcod = kxcoordinate*klength
kycod = kycoordinate*kbreadth

outvalue "newx", kxcod
outvalue "newy", kycod


kreinit invalue "reinit"
if trigger:k(kreinit,0.5,1) == 1 then
reinit IRtable
endif
IRtable:
event_i "i",1,0,0


ain,aignore soundin "tha.wav"

;convolute the input with the IR ftable

aout ftconv ain, gicon, 2048

ain delay ain, 2048/sr

kmix invalue "mix"

amix ntrpol ain, aout, kmix

outs amix, amix

endin


</CsInstruments>
<CsScore>
i 1 0 0
i 2 0 3
</CsScore>
</CsoundSynthesizer>
```

Reference

2D ray tracing in Csound GitHub
https://github.com/dharanipathirk/2D-ray-tracing-in-Csound/

The Canonical Csound Reference Manual
http://www.csounds.com/manual/html

Introduction to Computer Music: Volume One
http://www.indiana.edu/~emusic/etext/acoustics/chapter1_intro.shtml

Reverberation
http://johnlsayers.com/Recmanual/Pages/Reverb.htm

Understanding reverb
https://www.residentadvisor.net/features/1544

Math is fun
https://www.mathsisfun.com/geometry/index.html

Spring reverb
https://theproaudiofiles.com/spring-reverb/

Khan academy- Rendering
https://www.khanacademy.org/partner-content/pixar/rendering

Plate reverb
https://www2.ph.ed.ac.uk/~sbilbao/platerevpage.htm

RT60 Calculator
http://www.sengpielaudio.com/calculator-RT60.htm

Sound absorption
https://www.paroc.com/knowhow/sound/sound-absorption

Artificial reverberation – McGill university
https://www.music.mcgill.ca/~gary/618/week3/reverb.html

Perceived Differences between Natural and Convolution Reverberation in 5.0 Surround Sound - Alluri R. Shriram, University of Jyväskylä
https://jyx.jyu.fi/bitstream/handle/123456789/27082/1/URN%3ANBN%3Afi%3Ajyu-2011052610924.pdf

Ray tracing - wikipedia
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)