A Comparative Study of Analogue and Digital Mixing Techniques

Submitted in partial fulfilment
of the requirements of the degree of

Bachelor of Science (Honours)

of Rhodes University

Jessica L. Kent

Grahamstown, South Africa
October 31, 2014

Abstract

The difference between the summing quality of analogue and digital audio is widely debated. There are claims stating that audio summed by an analogue console sounds warmer than audio that has been summed by a digital audio workstation (DAW). It was hypothesized that a mechanism to provide the characteristics of an analogue mixer could be found and implemented in a digital mix. The project therefore aimed to prove whether an audible or visual difference between summing techniques existed. Possible causes for this difference, such as sampling rates, frequency response in human hearing, panning laws and analogue equipment were considered. A study of how three open source digital audio workstations (DAWs) define and implement gain, panning and summing of individual samples was then discussed. Previous testing of DAWs and alternate methods for summing samples were also reviewed.

An Audio Mix Analyzer was designed and implemented to facilitate the creation of audio tracks using various mixing algorithms, and to enable the comparison of two audio samples. Listening tests were performed using the Audio Mix Analyzer and summed sine waves were evaluated visually. Testing on the summing of an analogue mixing console was also performed. All participants could hear a difference between the analogue sum and the digitally-summed audio. This difference was dependent on the genre of the song and the number of tracks being summed. The digital algorithm that appeared to sound the most similar to analogue summing simply added audio samples together.

ACM Computing Classification System Classification

Thesis classification under the ACM Computing Classification System (1998 version, valid through 2014):
H.5.5 [Sound and Music Computing]: Methodologies and techniques
H.5.5 [Sound and Music Computing]: Signal analysis, synthesis, and processing
I.1.2 [Algorithms]: Analysis of Algorithms

General-Terms: Audio, Mixing Algorithms

Acknowledgements

First and foremost, I would like to thank Professor Richard Foss for his guidance, invaluable ideas and support throughout the entire year.

I would also like to thank Corinne Cooper for providing me with important sound technology concepts and the Rhodes University Music Department for allowing me to use their audio equipment.

A huge thanks to my parents, Andrew and Heather Kent, for their unconditional and financial support throughout my studies, including my four years at Rhodes.

I would also like to thank Brent Shaw and Jessica Hutchinson for their ideas and willingness to offer up their time whenever asked.

And finally, thank you to all of the listening test participants for their enthusiasm and for volunteering to assist with this project.

This work was undertaken in the Distributed Multimedia CoE at Rhodes University, with financial support from Telkom SA, Tellabs, Genband, Easttel, Bright Ideas 39, THRIP and NRF SA (TP13070820716). The authors acknowledge that opinions, findings and conclusions or recommendations expressed here are those of the author(s) and that none of the above mentioned sponsors accept liability whatsoever in this regard.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Statement of the Problem

In the sound engineering community there is debate about the sound quality of an audio mix that has been digitally summed instead of summed using an analogue mixing console. For the purpose of this paper, the terms "summing" and "mixing" will be used interchangeably and an "analogue" sample will refer to audio that has been summed in an analogue console, even if it played from a digital system. Some professionals say that a digital mix is lacking the "undeniable depth, width, punch and realism" (20, p. 1) of an analogue mix. However, as Leonard *et al.* (26, p. 1) states, there is not much "quantifiable evidence to support these claims" yet. And although others, like Cochrane (17), point out that summing with a digital audio workstation (DAW) or analogue console will not necessarily improve the quality of a song if it has not been correctly mixed (artistically speaking), it is generally accepted that the limitations of analogue equipment might account for the difference in the perceived character of the mix (18). For sound engineers who have been recording and mixing for the past 20 years, this unexplained variation of sound is a source of frustration, and summing boxes, like the Crane Song Egret or Rupert Neve Design 5059 (37), are being sold for the sole purpose of creating an analogue mix.

1.2 Goals of Research

The hypothesis for this thesis is that a mechanism to provide the characteristics of an analogue mixer can be found and implemented in a digital mix. The first goal of this project is to definitively prove whether a visual or audible difference in sound quality exists through visual and aural testing. The second objective is to design and implement an interface that facilitates user comparison of two audio samples. The user should also be able to control the mixing algorithm and other variables affecting the sound of a digitally summed mix using the interface. Lastly, it should be determined whether a digital mixing algorithm can be created to satisfactorily emulate the sound of a mix summed in an analogue console.

1.3 Thesis Overview

Audio concepts and relevant definitions are introduced in Chapter 2. The effect of sample rates, Fletcher-Munson curves, RMS levels and panning laws on the summing process are investigated. This is followed by an analysis of the technique used by analogue equipment to sum audio signals. Chapter 3 looks at the current mixing techniques employed by open source DAWs. An introduction to mixing algorithms is given before the code of Ardour, Audacity and Rosegarden is investigated. Their approach to gain representation, panning laws and sample summing is thoroughly explored. Alternative summing methods are then considered for possible implementation. These include an algorithm developed by Toth and two Dynamic Range Compression algorithms. Chapter 4 then goes on to review current literature relating to audio testing. Both subjective and objective testing methods are discussed, as well as previous published research papers. The importance of timing for real-time audio mixing is then deliberated.

Chapters 5 and 6 describe the design and implementation of the Audio Mix Analyzer. This is the program that was produced to allow a user to create a digital mix and then compare two audio samples. The requirements and object-oriented design are provided in Chapter 5. These include the

use case and class diagrams of the system. Chapter 6 elaborates on the language and development environment choices and the implementation of the system. The implementation is broken into four sections: the user interface, the descriptions of the mixing algorithms, the adjustment of the RMS level and the timing test calculations.

Chapter 7 depicts the entire testing process. This incorporates the recording of sine waves and audio tracks, the visual testing of the sine waves, the listening tests, analogue console experiments and timing tests. The aural testing elaborates on the configuration of the listening tests, the completion of the testing by participants, the results of the testing and an analysis of the findings. Chapter 8 concludes with a restatement of the project hypothesis, a description of the overall findings and proposed work that should be examined in the future.

Chapter 2

Mixing and the Analogue/Digital Debate

2.1 Definitions and Concepts in Audio Recording

To begin comparing digital and analogue mixing methods, some background information about audio recording is needed. Audio was first recorded using analogue equipment, usually onto magnetized tape. A recording device, typically a microphone, would record and convert fluctuations in air pressure to a measurable electronic signal. These varying electrical signals would then be recorded onto tape (40). As technology has advanced, digital equipment has been created to record, edit and mix audio signals. However, microphones still capture analogue signals, so some kind of conversion needs to take place for audio to be processed using a DAW. This has triggered many discussions about the quality variations between analogue and digital equipment and even among the "perceivable difference in sound quality between different DAWs" themselves, as stated by Leonard *et al.* (26, p. 1).

To convert the recorded voltages to a digital audio signal, measurements of the voltages, called samples, are taken at fixed time intervals. This

pulse code modulation (PCM) method is generally accepted as the standard system used in the conversion process (27). To make sure none of the original signal is lost, samples need to be taken at very high rates of 44.1 kHz, 48 kHz, 96 kHz and 192 kHz. An example of a wave being sampled at two different rates, the first one much lower than the other, is shown in Fig. 2.1. The rate of 44.1 kHz was chosen to be used on digital compact discs (CDs) for numerous reasons - one of which is that it is more than twice the rate of the highest frequency humans can hear (40, pp. 207–209). This is a result of the Nyquist Theorem, which states that at least two sample points are needed to recreate a wave of a specific frequency (27). These samples are then stored numerically and can be used to recreate and modify the recorded sound wave (40).

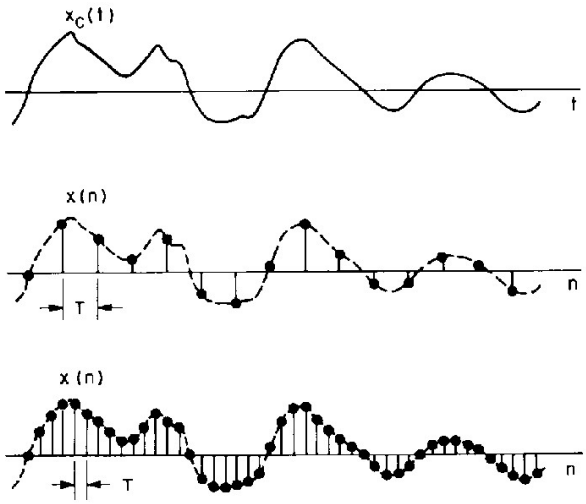


Figure 2.1: Sound wave sampled at two different rates (32, p. 80)

The amplitude or level of an audio wave is measured in deciBels - a logarithmic scale named after Alexander Graham Bell (40, p. 66). The standard calculation used to convert a ratio value (generally between 0 and 1) to its deciBel equivalent is given in Eq. (2.1) (40, p. 66).

$$deciBels = 20 \cdot log_{10}(amplitude\_ratio) \tag{2.1}$$

Just as the maximum frequency that can be stored is determined by the choice of sample rate, the bit depth limits the amount of signal that can

be stored by a digital system. When an input signal is too loud and the amplitude of the sound wave is too big for a digital system to store, the wave 'clips' causing the sound to distort (19). The industry standard bit depth for CDs is 16-bit although audio is usually recorded and processed at 24-bit or 32-bit and converted to 16-bit after being mixed and mastered (22, p. 51).

2.2 Sound Waves and Sample Rates

A sound wave being produced by a musical instrument is typically a complex wave, made up of the fundamental frequency and higher harmonics (or overtones) that are a positive multiple of the fundamental frequency. In the 1800s, Joseph Fourier discovered that any existing sound wave can be represented mathematically as a sum of "simple harmonic terms" (24, p. 15), which are usually sine waves. This means that the sound wave in Fig. 2.2 can be expressed as the sum of two sine waves.

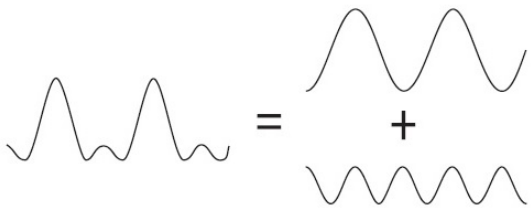


Figure 2.2: Sound wave comprised of two sine waves (12, p. 7)

In conventional DAWs, audio is recorded and summed at a sampling rate of either 44.1 kHz or 48 kHz. According to the Nyquist Theorem, frequencies above half the sampling rate (i.e. 22.05 kHz or 24 kHz respectively) are not recorded because there are not enough sample points to recreate them. However, as Izhaki (22, pp. 456–457) points out, if a complex wave or any distortion is recorded at 48 kHz, the harmonics of the wave above 24 kHz will not be recorded. This is not a problem on an analogue console. Although humans cannot hear sound waves higher than 20 kHz (40, pp. 45-46), they are still recorded by the analogue console. When multiple audio tracks, one or more of which contains sound waves above 24 kHz, are

digitized and then summed together, it is theoretically possible that this high frequency wave will affect sounds waves below 20 kHz. This would affect what humans can hear and could possibly be the difference heard when summing audio digitally.

However, Watkinson (40, pp. 729-730) emphatically says that although demonstrations may have shown that higher sampling rates sound better, the experiments were not designed or carried out correctly. He also states that while converters sampling at 96 kHz have been proven to sound better, "this does not prove that 96 kHz is necessary" (40, p. 730) because a better designed converter sampling at a lower rate could produce the same results.

2.3 Fletcher-Munson curves and RMS Levels

Not only are human ears limited to the frequencies they can hear, but they perceive some frequencies to be louder or softer than others. A set of experiments performed by Harvey Fletcher and W.A. Munson in 1933 (22, p. 12) resulted in the Feltcher-Munson curves (as shown in Fig. 2.3) or equal-loudness curves. These curves show that in order to hear a bass

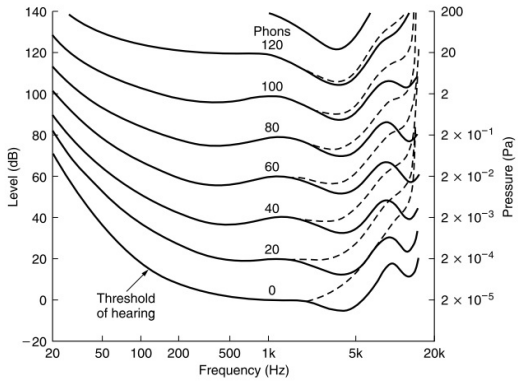


Figure 2.3: Fletcher-Munson curves (40, p. 46)

frequency under 100 Hz as loud as a mid-range frequency of, say a 1kHz

wave, the level of the audio needs to be higher. In other words, bass frequencies are not heard clearly in softer audio but our ears are very receptive to frequencies between about 2 kHz and 5 kHz (40, p. 45). The fact that the "ear's frequency response changes with signal level" (34, p. 30) means that the average loudness of two audio tracks need to be the same to compare them (22, p. 96). The Root Mean Square (RMS) level can be used to measure this average loudness of an audio signal. It can be worked out mathematically using Eq. (2.2) (30).

$$rms = \sqrt{x^2} \tag{2.2}$$

2.4 Panning Laws

When creating a stereo mix of a song, individual tracks are usually panned to change the horizontal location that they appear to originate from. The most common way to compute this position in a DAW is to use a "sine-cosine" calculation or panning law (21). A single track is panned in a DAW using some sort of digital panning control. The specified panning is then converted to an angle between 0 and 90 degrees (as shown in Fig. 2.4) representing a panning anywhere from hard left to hard right respectively (21). The output of the stereo track is calculated using Equations 2.3 and 2.4, where a is the angle. Although these equations provide constant loudness, where $left^2 + right^2 = input^2$ (21), when tracks are center-panned there is a 3dB level increase heard to be coming from the center. To compensate for this, a panning law can be used.

$$left = cos(a) \cdot input \tag{2.3}$$

$$right = sin(a) \cdot input \tag{2.4}$$

The four most frequently used panning laws are the 0dB, -3dB, -4.5dB and -6dB panning laws (22, pp. 189-193) where the values refer to the total reduction of deciBels on center-panned tracks. It is well-established that the "-3dB pan law is generally the best option when stereo mixing" (22, p. 193) but not all DAWs use this as the default panning law. For example, Rosegarden uses the 0dB panning law as a default setting (see AudioLevel.cpp in Appendix A.3).

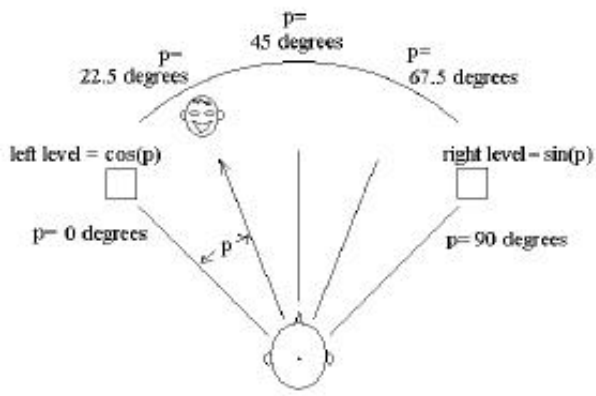


Figure 2.4: Angle of panning (21)

2.5 Analogue Equipment

When examining various digital summing algorithms it is useful to investigate how an analogue mixer combines signals to produce a summed output. An example of a summing amplifier circuit is shown in Fig. 2.5 where three input signals (V_1, V_2 and V_3) are added together to produce an output voltage ($-V_{out}$). When the input impedances ($-R_{IN}$) are the same, the output can be calculated using Equation 2.5. This means that the output voltage is proportional to the sum of the input voltages (38).

$$-V_{out} = \frac{R_F}{R_{In}}(V_1 + V_2 + V_3) \tag{2.5}$$

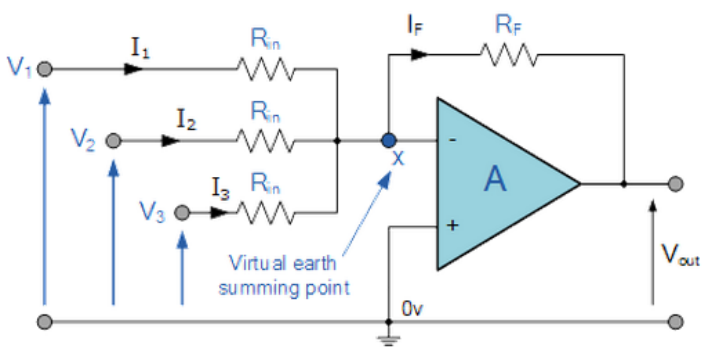


Figure 2.5: Summing Amplifier Circuit (38)

2.6 Summary

This chapter introduced various audio concepts that could affect the summing process. An overview of sample rates, Fletcher-Munson curves, RMS levels, panning laws and analogue equipment was given.

Chapter 3

Current Techniques for Digital Mixing

3.1 Mixing Algorithms

Since a song typically consists of multiple tracks, a DAW needs to be able to sum these separate tracks to create a master stereo track. This process is not always as simple as adding the two values together, as this new summed value could clip or "over- or underflow the range available" as Vogler (39, p. 1) puts it. Therefore various mixing algorithms have been developed which aim to add the tracks together in such a way that the level of each individual track is not perceived to be louder or softer than it was originally. There is a diverse selection of DAWs available and industry recording engineers argue about which DAW is technically the best for summing audio. For the purposes of this investigation, the three DAWs that were selected are all free, open source applications. These applications are Audacity (15), Ardour (13) and Rosegarden (33). They were chosen with the aim of examining the source code to determine the specific summing algorithms, gain representation and panning laws used, as these are the three properties that could be responsible for the quality difference.

3.2 Representation of Gain

Audacity's interface allows the user to choose between a linear or logarithmic scale to represent the gain on the meter display. The linear scale has a minimum of 0 and maximum of 1 while the default logarithmic scale has a maximum level of 0 dB (14). Any audio that exceeds these maximum values is shown to be clipping. The methods *ToDB* and *ToLinearIfDB* (see Meter.cpp in Appendix A.1), based on Equation (2.1), are used to convert the user's chosen gain between representations. However, all sample calculations use the linear scale and gain is represented as a double floating point value between 0 and 1.

Ardour makes use of floating point representation to store the gain value, which can be between $-\infty$ and 6dB using the logarithmic scale and a floating point value between 0 and 2 using the linear scale (13). The conversion calculations used are also based on Equation (2.1) and can be seen in dB.h in Appendix A.2.

Rosegarden provides five distinct fader types (see AudioLevel.cpp in Appendix A.3), each with specific minimum, maximum and step values. Depending on the settings, the user can choose a gain value between -70 dB and 10 dB. This value is converted to a floating point value between 0 and 10 using the *dB_to_multiplier* method in Appendix A.3. Interestingly, Rosegarden uses a base factor of 10 instead of using 20 as Audacity and Ardour do.

3.3 Panning Laws

Audacity's panning control allows the user to specify a value between -1 and 1 in increments of 0.1 (see ASlider.cpp in Appendix A.1), which is displayed as being a percentage left or right as depicted in Fig. 3.1. The method *GetChannelGain* (see WaveTrack.cpp in Appendix A.1) calculates the gain for either a left or right output by using Equation (3.1) to get a positive pan value and multiplying that by the user-specified gain to find

the gain for the channel. This is not the sin-cos panning law discussed earlier and Audacity therefore, does not account for an increase in level when tracks are center-panned.

$$\begin{cases} right = pan + 1.0 & pan < 0 \\ left = 1.0 - pan & pan > 0 \end{cases} \tag{3.1}$$

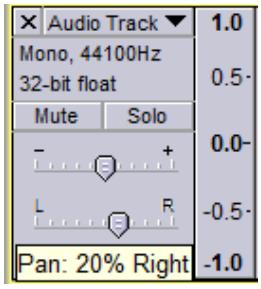


Figure 3.1: Panning slider set at "20% Right" in Audacity (15)

Ardour defines their panning in terms of Azimuth angles, a coordinate system using North and East as reference points (31). The user can set the panning anywhere between 0 and 180 degrees (13) and this is converted to a double floating point value between 0 and 1 using the method *azimuth_to_lr_fract* (see panner.h in Appendix A.2). In the method *distribute_one* (from panner_2in2out.cc in Appendix A.2), Ardour applies the panning to the audio samples.

In the Rosegarden code AudioLevel.cpp (see Appendix A.3), the various panning laws are given in the panGainRight method, where a pan value between -100 and 100 is converted to a value between 0 and 2 (33). The -0db and -6db laws add 100 to the panning value and then divide this sum by either 100 or 200 while both -3dB panning laws return the square root of this calculation. The overall gain and panning for the left and right channel is applied to the samples in the *setBussLevels* method in the Rosegarden code AudioProcess.cpp.

It is interesting to note that each DAW defined and handled panning uniquely. This may be one of the reasons why digitally summed audio is difficult to test and compare, as remarked by Leonard *et al.* (26) in their investigations of DAW summing.

3.4 Summing of Samples

In Audacity the samples, after being multiplied by the selected gain, are simply added together and it is up to the user to avoid clipping. As can be seen in Mix.cpp in Appendix A.1 the summing process takes place in the *MixBuffers method*, which loops through all the audio samples and multiplies each one by the gain of the channel (a floating value between 0 and 1) before adding it to the destination buffer.

Ardour implements the mixing process in the appropriately named mix.cc (see Appendix A.2) where two methods for mixing the samples are given - one including gain and one without. These methods resemble the implementation by Audacity: the samples are added together and stored in the destination buffer. The *default_mix_buffers_with_gain* method multiplies the samples by a floating point gain value (between 0 and 2, as previously mentioned) before adding them.

Rosegarden first adds the samples from each channel together and then adds them to a destination buffer (see PlayableAudioFile.cpp in Appendix A.3) after having applied panning and gain in the *setBussLevels* method.

Although different gain and pan algorithms are used, each DAW sums the samples in very similar ways. This means there is an opportunity to modify and experiment with this process when implementing new mixing algorithms to emulate the analogue mixing process.

3.5 Alternative Summing Methods

3.5.1 Viktor T. Toth

Since the effect of audio summing methods has been so widely debated, alternative methods for adding the samples together have been developed. For example, Toth devised Equation (3.2) to calculate the summed value of two samples a and b that themselves have values between 0 and 1. This

formula can be extended for three samples as in Equation (3.3). He refined the formula to account for samples that could be added together without clipping to get Equation (3.4). However, as Voegler (39) points out, this method is not symmetrical (see Figure 3.2) and favours extreme values in the subsequent mix.

$$sum = a + b - ab \tag{3.2}$$

$$sum = a + b + c - ab - ac - bc + abc \tag{3.3}$$

$$sum = \begin{cases} 2ab & a < 0.5 \cap b < 0.5 \\ 2(a+b) - 2ab - 1 & a \geq 0.5 \cup b \geq 0.5 \end{cases} \tag{3.4}$$

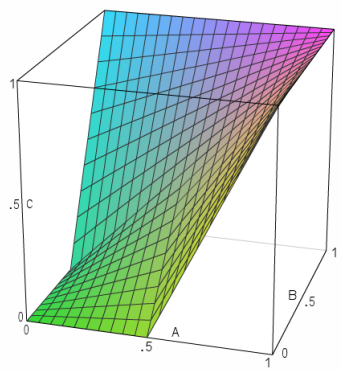


Figure 3.2: Viktor T. Toth method (39)

3.5.2 Linear and Logarithmic Dynamic Range Compression

Voegler (39) himself developed a method for adding two samples together which he calls "Loudness Normalization by Logarithmic Dynamic Range Compression". He begins by stating that the easiest method for making sure two samples stay within the output range is to divide by 2 as in Equation (3.5). However, if one sample was very quiet the other sample would

effectively be half as loud as it was originally. He decided to then dynamically compress the output level only if it was above a specific threshold. This would ensure that most samples could be added together without further calculations. This Linear Dynamic Range Compression method is presented in Equation (3.6) where $x = a + b$ and is a floating point value between 0 and 2 and t is the chosen threshold between 0 and 1. As can be seen in Fig. 3.3, where a threshold of 0.6 is used, the compression takes effect pretty dramatically, which results in an audible reduction in level. To lessen the impact of this, Voegler (39) used mathematics beyond the scope of this paper to produce a Logarithmic Dynamic Range Compression method given in Equation (3.7). This results in a smoother transition as compression is applied as shown in Fig. 3.4. It would be interesting to see what effect implementing these alternative summing methods would have on the sound quality of a digital mix.

$$sum = \frac{a+b}{2} \tag{3.5}$$

$$sum = \begin{cases} x & -t \leq x \leq t \\ \frac{x}{|x|} \cdot (t + \frac{1-t}{2-t} \cdot (|x| - t) & |x| > t \end{cases} \tag{3.6}$$

$$sum = \begin{cases} x & -t \leq x \leq t \\ \frac{x}{|x|} \cdot (t + (1-t) \cdot \frac{ln(1+f_\alpha(t) \cdot \frac{|x|-t}{2-t})}{ln(1+f_\alpha(t))} \cdot & |x| > t \end{cases} \tag{3.7}$$

3.6 Summary

This chapter investigated current mixing algorithms used by open source DAWs. The representation of gain, panning and the summing of samples by Ardour, Audacity and Rosegarden were studied. Alternative summing methods, like those designed by Toth and Voegler, were explored.

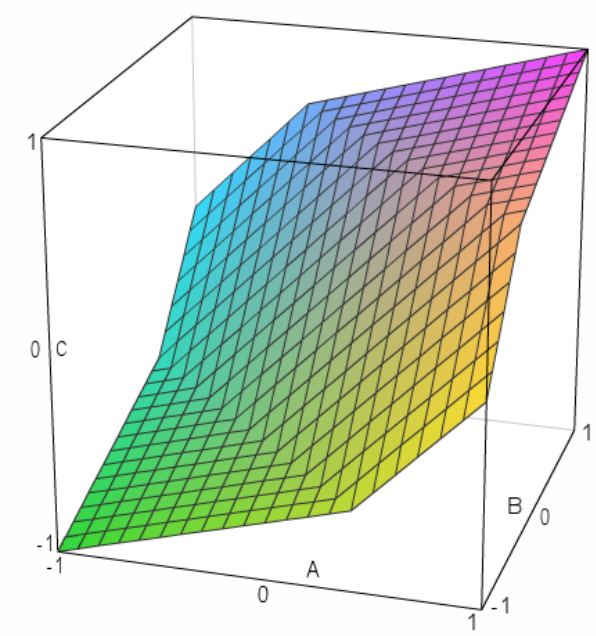


Figure 3.3: Linear Dynamic
Range Compression (t = 0.6) (39)

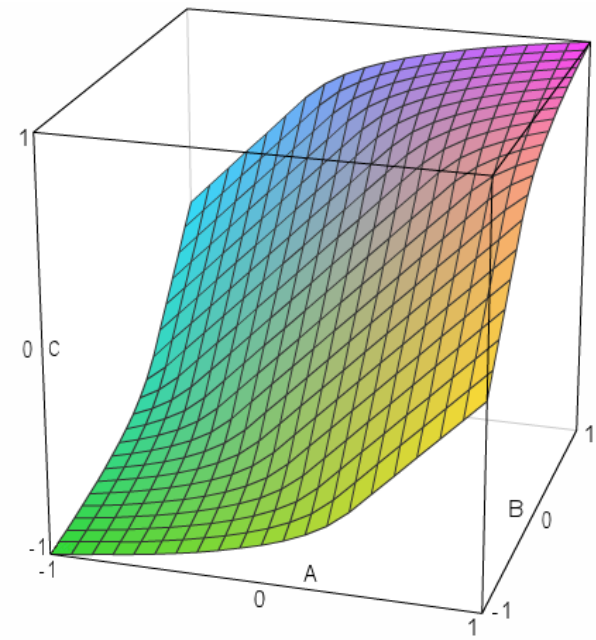


Figure 3.4: Logarithmic Dynamic
Range Compression (t = 0.6) (39)

Chapter 4

Overview of Audio Testing

4.1 Testing

After implementing these panning and mixing algorithms, some sort of testing should be conducted in order to make any conclusions about their effect on the overall sound quality of the mix. As far as audio testing is concerned, both objective and subjective testing is normally performed as music has both technical and artistic properties. As Watkinson (40, p. 708) explains, combining both approaches is "the only way to achieve outstanding results".

4.2 Objective Testing Methods

In experiments with investigations similar to this one, objective testing has usually been performed using visual testing and difference methods (26). Objective testing includes measuring the difference between a sound wave and the same sound wave after it has been processed in some way. Results should be reproducible in order for the test to be seen as a valid method for analyzing a particular system (40, p. 710). However, conclusions drawn from objective tests do not necessarily indicate how well a system will perform in the real world. This is why subjective testing is usually employed

in combination with objective testing. Visual or difference testing as used in the experiments by Leonard *et al.* (26) will be used to examine if any differences exist between the summing of analogue and digital sound waves in the subsequent thesis.

4.3 Subjective Testing Methods

For subjective testing to be accurate, the loudspeakers being used need to be as accurate as possible (40, p. 720). Listening tests are the most common method used to subjectively compare audio samples but it is difficult to remove bias from the results (40, p. 709). In the experiments conducted by Leonard *et al.* (26) listening tests were carried out on three specifically chosen DAWs out of the five that were originally analyzed. When conducting digital listening tests the interface used to switch between and compare samples is very important and this needs to eliminate any possible bias.

4.4 Previous Analog/Digital Tests

There have been very few official tests done as far as analogue or digital summing is concerned. The most recent testing was conducted by Leonard *et al.* (26) where the internal summing of five different DAWs was investigated. They decided that they needed to test three specific aspects of each DAW to find out which of the factors, if any, were the source of the differences. They chose to examine how the input was read in and any error correction done, the internal summing of the sources and how the gain was changed in each DAW. After some initial testing (25) they discovered an error in the panning of one of the tracks. This resulted in an examination of the effect that panning had on the summing of the tracks. They found that the different panning laws of the DAWs caused significant variations in output levels and sound quality. They then investigated the panning laws of each DAW and found the panning technique to be inconsistent across DAWs, especially when center-panning the audio tracks instead of

hard panning left or right (26). After retesting and normalizing the audio, they discovered that there was no difference in the addition of the samples between any of the DAWs. They did, however, find an objective and subjective difference in level on the summing of tracks that were center-panned even when the same panning law was used across all DAWs. They suggest that the most problematic component appears to be the individual panning laws utilized by each DAW and recommend further testing of DAW panning laws to explore the effect on sound quality.

Another experiment done by Aarts (11) compared two DAWs with different sampling rates to an analogue system in order to find the most similar sounding DAW. Their tests showed that the DAW with the higher sampling rate was the system that sounded most like the analogue system. They do not, however, say if there were any audible differences between the analogue and digital systems.

4.5 Timing

If audio is to be mixed and played back in real time, the summing algorithm needs to be extremely efficient to calculate the sum as fast as possible. When a sample rate of 48kHz is chosen the instructions used to add the samples need to be executed in under $\frac{1}{48000}$ of a second or 20.83 μs. If a higher sample rate is used then the summing needs to be performed even faster. Table 4.1 shows the maximum amount of time available to calculate the sum corresponding to various sample rates.

Table 4.1: Maximum Times for Calculations in microseconds

Sample Rate (kHz)	**Time (μs)**
44.1	22.68
48	20.83
72	13.89
96	10.42

4.6 Summary

An overview of subjective and objective testing mechanisms was considered in this chapter. Previous papers dealing with digital summing were explored before the timing requirements of real-time mixing was discussed.

Chapter 5

Requirements and Design for Mixing Technique Testing System

5.1 Requirements for Testing System

In the early discussions about the difference between analogue and digital mixing it became clear that there was no established method for testing two audio samples. There was also no interface available that enabled a user to instantly and seamlessly switch between an analogue sample and a digital one to hear if differences between them existed. Therefore, it was decided that a program with this functionality should be created to assist with listening tests. This would accomplish the first goal of the project. The second goal was to determine if a digital mixing algorithm could emulate the summing of an analogue console. Mixing and saving multiple samples using various mixing algorithms would be easier to achieve with a user-friendly interface. The initial mock-up of the Audio Mix Analyzer interface is shown in Fig. 5.1.

These two goals motivated the development of a program that could both create a digital mix of up to eight audio tracks with different mixing algo-

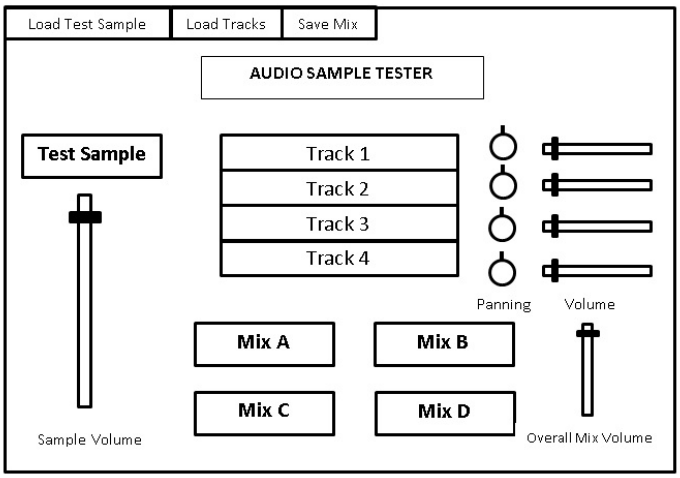


Figure 5.1: Interface Mock-up

rithms and allow a user to evaluate the audio samples that were created. When creating a mix the user would be required to:

- Start the application
- Load tracks to create a mix
- Create a mix
- Save a mix

When evaluating a mix the user should be able to:

- Load two tracks for comparison
- Play the tracks out of an audio device
- Switch between the tracks

These requirements are shown in the Use Case Diagram in Fig. 5.2.

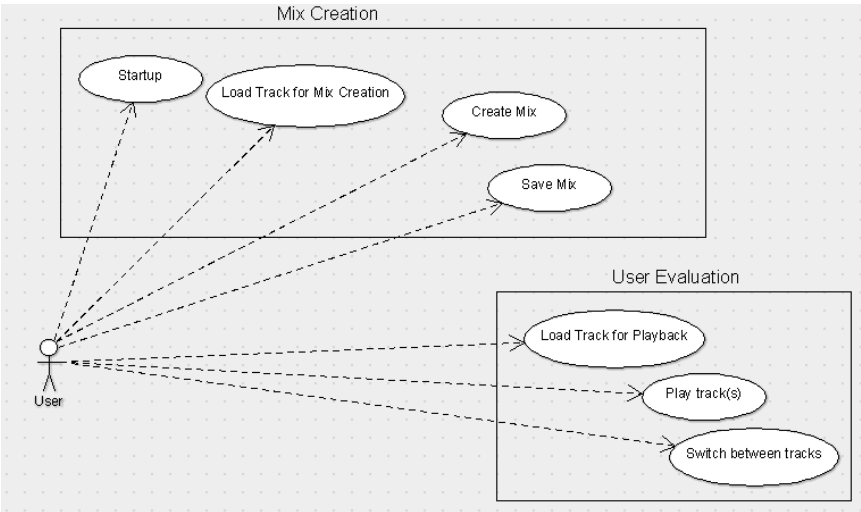


Figure 5.2: Use Cases for Audio Mix Analyzer

5.2 Object-Oriented Design

Since the Audio Mix Analyzer can be separated into two different components, mixing tracks and playing tracks, it was only logical to create a class for each component. As can be seen in the class diagram (Fig. 5.3), these two classes are the Track Mixer and Track Player. Each class contains human interaction component functionality and aspects of the problem domain.

The Audio Mix Analyzer was implemented using functionality inherited from the JUCE library, which is further elaborated on in Section 6.1. This impacted the configuration of the classes, especially in the case of the main window components. The *AudioTester* class implements the main initialization and shutdown of the program. It also creates a *MainWindow* object, which is used in all JUCE programs, and this initializes the *MainComponent* object in the program. The tab functionality, also inherited from JUCE, is then used to create a tabbed component with two separate tabs - one for the Track Mixer and the other for the Track Player. This initialization process can be seen in the Startup sequence diagram in Fig. B.1.

Other classes inherited from JUCE that are used in the implementation of the Audio Mix Analyzer are listed below:

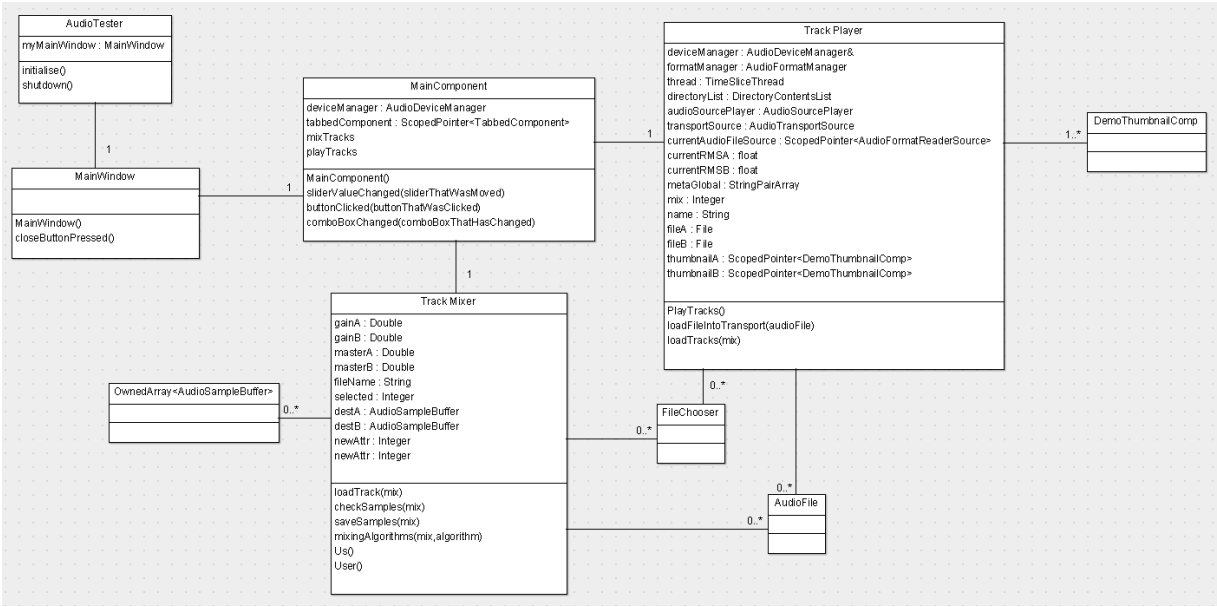


Figure 5.3: Class Diagram for Audio Mix Analyzer

- OwnedArray
- FileChooser
- AudioFile
- DemoThumbnailComp

The next stage is to design the behavioral aspects of the program. The complete behavioral design and state transitions are given in Appendix B. The nature of the design is clarified in Chapter 6 where elements of the design will be used to illustrate specific aspects of the implementation.

5.3 Summary

This chapter listed the requirements and design of an Audio Mix Analyzer. Use case and class diagrams were used to explain the main objectives of the program. These were the creation of digital mixes using various mixing algorithms and an interface for users to compare audio tracks.

Chapter 6

Implementation and Timing Issues

6.1 Language Selection and Development Environment

Based on the design of the Audio Mix Analyzer and the requirements of audio programming, it was decided that C++ (2) should be used as the primary coding language. C++ has many libraries specifically designed to implement audio-based programs which made it a natural choice for this project. One of these libraries is the Jules' Utility Class Extensions (JUCE) library (6).

JUCE also has a "project-management tool" (5) designed specifically to assist with the implementation of the user interface, known as the Introjucer. It has a drag-and-drop facility to aid in the creation of GUI components. The Introjucer also manages all the JUCE modules being used in the project and adds the necessary code to the project.

Once it was decided that the project would be programmed using JUCE and C++, it automatically followed that Microsoft Visual Studio 2010 (8) would be used for the main implementation of the program as this was the only viable development environment.

6.2 Implementation

After the requirements and design of the program had been finalized, and the programming environment had been chosen, the Audio Mix Analyzer could be implemented. There were five main components to this process: the interface, the mixing algorithms, the adjustment of the RMS level, the playback of tracks and the timing of the algorithms. Each of these topics are discussed in detail below.

6.2.1 Interface

In Section 5.1 it was established that the two different elements of the program should be implemented in two different classes and it seemed logical to separate these into two distinct interface components. The easiest way to do this was to use the tab component from the JUCE library.

Initially the Mix Creation tab of the Audio Mix Analyzer interface, based on the mock-up design (Fig. 5.1), was created using Introjucer. The code was then modified manually using Microsoft Visual Studio 2010. The Mix Creation tab can be seen in Fig. 6.1.

Mix Creation
The Mix Creation tab allows the user to create two different mixes at once and the interface is split into two identical halves. Each half has the potential for eight 'Load Track' buttons, pan and gain sliders, a combo box used to select a mix, a slider showing the RMS level and 'Mix Sample' and 'Save Sample' buttons.

Each row of track controls corresponds to one track. From the beginning of the project, the decision was made to only allow a maximum of eight tracks per mix for each half. When a user begins the program only one row of track controls is visible for each mix. Once the user loads the first track, the second row of controls becomes visible. After the second track is loaded, controls for the third are shown, and so on. As a user loads the audio files they want to sum, the samples from that file need to be stored so that they

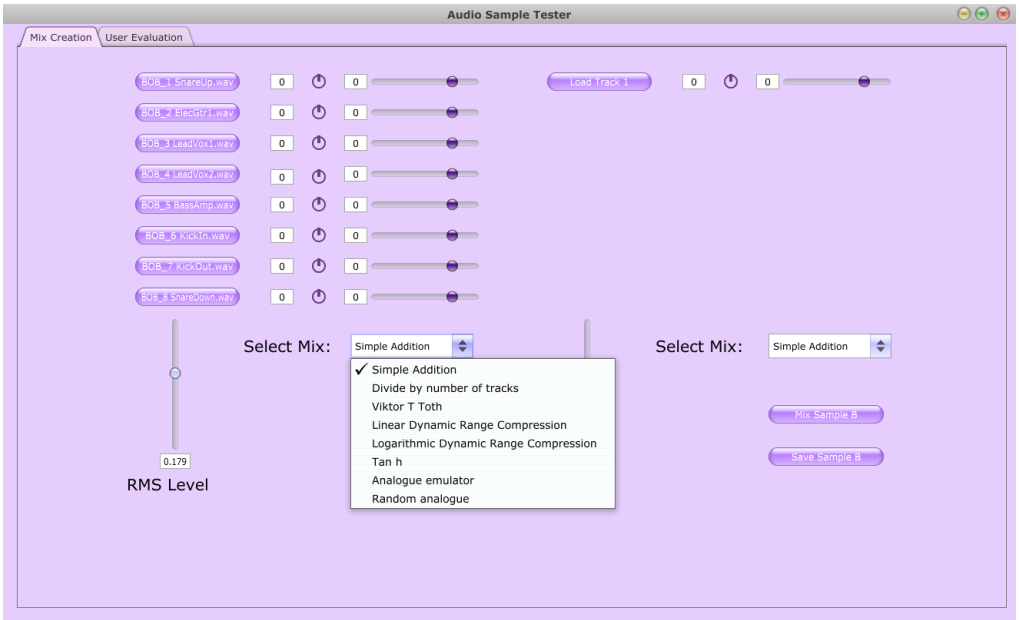


Figure 6.1: Mix Creation tab of Audio Mix Analyzer

can later be read and mixed together. (Many different approaches were taken before finally using two *OwnedArrays* of *AudioSampleBuffers* for the two mixes.) When the 'Load Track' button is pressed the series of events shown in Fig. B.2 is executed:

- A File Chooser is created by the Track Mixer
- The user selects the desired WAV file
- The Track Mixer reads and adds the samples from the file to an array
- The name of the audio file is displayed on the interface

Once the user has loaded all the tracks they want, they can select the mixing algorithm they want to use using the 'Select Mix' combo box JUCE component. When the 'Mix Sample' button is pressed the samples will be summed using the specified algorithm. During the summing process, the program also ensures that the RMS level is within a certain range. The implementation of the mixing algorithms and adjustment of the RMS level is discussed further in Section 6.2.2. When the 'Save Sample' button is

pressed a file browser is opened for the user to save the newly created audio file (see Fig. B.4).

User Evaluation

After all the audio files required have been created, the user can use the User Evaluation tab (see Fig. 6.2) to compare two tracks. This tab allows the user to load two tracks using the 'Load Track buttons'. It also allows the user to control the playback of audio files using 'Play', 'Pause' and 'Stop' buttons and to switch between audios files by pressing the 'Play Track A' and 'Play Track B' buttons. The two thumbnails used to display the waveforms and adjust the playback position of the file were inherited from the *DemoThumbnailComp* class in the Demo code provided by JUCE. The RMS level of the audio file is also displayed after a file has been loaded.

Figure 6.2: User Evaluation tab of Audio Mix Analyzer

The 'Load Track' buttons can be used to load the desired two files. As the user presses 'Load Track' the events shown in Fig. B.5 occur:

- A File Chooser is created by the Track Player
- The user selects the desired WAV file

- The Track Player reads the audio file and sets the corresponding thumbnail
- The waveform, name and RMS level of the audio file is displayed on the interface

Now that the user has loaded the files for comparison, they can be played back to the user, as in Fig. B.6. When the user presses 'Play', the Track Player inherits from the *AudioTransportSource* class to play the file. However, the user can also press the 'Pause' and 'Stop' buttons during playback. The state transition diagram for these cases are given in Fig. B.8. As can be seen from the diagram, the current play position is stored when the user pauses playback but is set back to the beginning when 'Stop' is pressed.

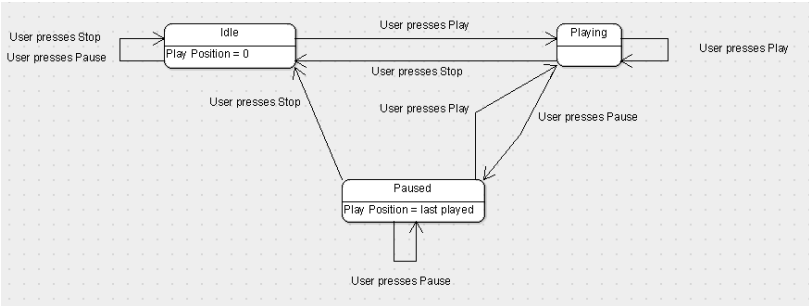


Figure 6.3: State Diagram for Playback

Once playback has begun, the user can switch between audio files by pressing one of the 'Play Track' buttons . The method used to achieve this is as follows (see Fig. B.7):

- The Track Player gets the current play position of the file from the *AudioTransportSource* class
- The corresponding audio file is loaded into the transport source
- The play position for the (new) file is conveyed to the *AudioTransportSource* class
- The Track Player calls the *AudioTransportSource* class to play the (new) audio file

6.2.2 Mixing Algorithms

When 'Mix Sample' is pressed, a check is performed to ensure every track has the same number of samples, if more than one track has been loaded. Once this is completed, the process exhibited by Fig. B.3 occurs:

- The Track Mixer creates a destination buffer
- The corresponding samples in each track are summed together using the specified mixing algorithm (see Fig. A.4)
- The summed samples are stored in the destination buffer

The implementation of each mixing algorithm is described in detail below.

Simple Addition
The Simple Addition algorithm was the simplest to implement (Appendix A.4.1). The sample for each track was retrieved and multiplied by the value of the gain control for that track. It should be noted that a method to convert the decibel value from the slider to a gain value was used. These samples were then simply added together. This is the same algorithm implemented by Ardour, Audacity and Rosegarden (see Section A.4).

Divide by Tracks
The Divide by Tracks (Appendix A.4.2) algorithm sums the samples similarly to the Simple Addition algorithm but divides the final sum by the number of tracks the user has loaded. Eq. 3.5 shows the calculation for two samples.

Viktor T Toth
Toth's method (Eq. (3.2)) was implemented next. The value of 0.5 being used in Toth's equation needed to be adjusted as the range of possible sample values is between -1 and 1 in the context of the program, and not between 0 and 1. This value was adjusted depending on the number of tracks needing to be summed (see Appendix A.4.3). If every sample was less than this value then Eq. (6.1) was used. If one of the samples was larger than this value then Eq. (6.2) was used to calculate the sum. However, as

mentioned in Section 3.5, this algorithm is not symmetrical and it was decided that it should not be used in the listening tests due to the limited number of algorithms that could be tested.

$$sum = 2ab \tag{6.1}$$

$$sum = 2(a + b) - 2ab - 1 \tag{6.2}$$

Linear and Logarithmic Dynamic Range Compression

The Linear Dynamic Range Compression Algorithm (Appendix A.4.4) and Logarithmic Dynamic Range Compression (Appendix A.4.5) algorithms were implemented by adding the samples together and then, if the absolute value of the sample was above a threshold of 0.6, a new value was determined using Eq. (3.3) and Eq. (3.4) respectively. A list of $f_\alpha(t)$ values (see Eq. (3.4)) corresponding to various thresholds was given in Voegler's publication (39). Since a threshold of 0.6 was chosen a value of 7.48 was used for $f_\alpha(t)$. If time had permitted, algorithms with different thresholds would also have been implemented. Both the Linear and Logarithmic algorithms are very similar in concept. Only one of these algorithms could be tested and the Logarithmic algorithm was chosen to represent the Dynamic Range Compression method.

Tanh

The last three algorithms implemented were experimental and the product of speculation about different summing methods. The hyperbolic function tanh appeared in several forums and online articles (see (1), (10) and (7)) concerning audio summing. In an attempt to create and experiment with new algorithms, the Tanh algorithm (Appendix A.4.6), which applies the tanh function to the summed samples, was implemented.

Analogue Emulator

The Analogue Emulator (Appendix A.4.7) applies a 3 dB gain reduction to the sum. This idea stemmed from the '-3dB pan law' used by the Mackie desk. Since the RMS level is adjusted after summing, this did not seem

to have an overall affect on the mix. The Tanh algorithm was therefore selected for listening tests.

Random Addition

The final algorithm came from the idea that analogue mixing consoles are inherently random. They are made up of electronic components that deteriorate over time and the time taken for signals to travel through the desk can cause phase problems which would affect the sound of the summing. It was therefore suggested that a 'random' algorithm be tested (Appendix A.4.8). This algorithm generates two random values. An integer between 0 and 9 was divided by 1000 to determine a small 'random' number to add to the summed samples. An integer of either 0 or 1 is then generated to evaluate the sign of the final value. Although the value added was only in the range -0.0009 and 0.0009, it still created an audible 'white noise' sound when this algorithm was used. Thus, the algorithm was ruled out as a possibility for the listening tests.

6.2.3 RMS Level

As expressed in Section 2.3, the RMS levels of each audio sum needed to be within a certain range in order to be suitable for the listening tests. In an explanation about mastering, Bartlett (16, p. 336) agrees with the recommendation from Bob Katz (23) that RMS levels for pop music should be about -14 dB. Since three different genres of music were being considered, an RMS level of -15 dB was selected. This translated to a gain voltage of roughly 0.177828. An acceptable RMS range was hence determined to be between 0.179 and 0.180, which corresponds to levels between -14.94 dB and -15.04 dB, respectively.

After mixing the samples the RMS value of the final destination buffer was retrieved. If this value was not within the specified range, the gain of the buffer was either increased or decreased using the code in Fig. 6.4 until it was within this range (Appendix A.5). Once it reached a satisfactory value, the RMS Level slider in the Mix Creation tab was set to reflect this.

```
float r = destA.getRMSLevel(0,0,length);
while(r < 0.177 || r > 0.179){
    if(r < 0.177)
        destA.applyGain(0,0,length,1.01);
    if(r > 0.179)
        destA.applyGain(0,0,length,0.99);
    r = destA.getRMSLevel(0,0,length);
}
```

Figure 6.4: Code to Increase or Decrease RMS Level

6.2.4 Timing Issues

The Audio Mix Analyzer was designed for static mix creation and playback, but it was important to check if it could be implemented in a real-time scenario. The code in Appendix A.6 was used to time the different mixing algorithms of the program. A time object was instantiated after the 'Mix Sample' button was pressed and the current time was obtained. The mixing process for a selected algorithm, which - as previously established - consisted of two for loops and a switch statement, was then executed. The time taken to complete the summing was calculated by subtracting the previously-obtained time from the current time. This information was then printed out using the debugger.

6.3 Summary

This chapter described the selection of the language and development environment used to implement the Audio Mix Analyzer. The implementation of various components of the program, namely the interface, mixing algorithms, RMS levels and timing calculations, was detailed.

Chapter 7

Aural and Visual Feedback

7.1 Audio Sample Recordings

Before the Audio Mix Analyzer could be used to facilitate any listening tests, audio sample tracks needed to be recorded. It was decided that songs of various genres would be used for the listening tests while visual testing could be conducted using summed sine waves of different sample rates and bit depth. All audio samples were summed using both analogue and digital devices to allow for a direct comparison. A Mackie 24•8 (29) analogue mixing console was used for the analogue summing and the DAW chosen for digital summing was Cubase Artist 6.5 (3) created by Steinberg. The converter used was the RME Fireface UFX (4). The "-3dB @ center" (28, p. 2) panning law is used by the Mackie desk therefore the Cubase and RME panning laws were set to the same panning law manually to guarantee that this did not affect the recordings. No additional panning or gain adjustments were used in this process.

7.1.1 Sine Waves

In Chapter 2 it was debated whether sound waves higher than 20 kHz, that is, inaudible waves, could affect an audible sound wave if they were

summed by an analogue console. To test this theory, various sine waves were generated using Wavelab 6.11 (9) and summed using both the Mackie desk and Cubase. The tests were designed to determine the effect of higher sampling rates and bit depths. For this reason, sine waves were generated with the following four sample rate and bit depth configurations:

1. 48 kHz at 24-bit
2. 96 kHz at 24-bit
3. 48 kHz at 32-bit
4. 96 kHz at 32-bit

A 30kHz wave was chosen to be summed with 1kHz and 300Hz waves. It was also decided that a wave with complex frequencies should be investigated, and a 20kHz square wave was selected for this. The list of summed waves is shown below. Section 7.2 documents the analysis of these sine waves.

- 30 kHz only
- 300Hz + 30kHz
- 1kHz + 30kHz
- 300Hz + 1kHz + 30kHz
- 1kHz + 20kHz square wave

7.1.2 Multitrack recordings

The audio sample tracks were recorded with the listening tests in mind. It was determined that contemporary songs of different genres should be used. This would establish whether the genre of a song influenced the sound of the summing. Three of the more popular broad genre classifications

were picked to represent the diverse nature of contemporary music. These classifications were 'Country', 'Rock' and 'RnB/Electronic'. Now eight WAV tracks for each song were needed to be summed and recorded. The *'Mixing Secrets' Free Multitrack Download Library* contains multiple 'raw multitrack projects' (35) of uncompressed and unedited WAV tracks. From this library three songs that could be classified by the indicated genres were chosen. These songs were

- 'Black Out Betty' by Banned From The Zoo [1] (Pop/Rock)
- 'Catching Up' by Lyndsey Ollard (Singer-Songwriter)
- 'Rockshow' by ANiMAL [2] (Hip-Hop/Electronic)

All the tracks were sampled at a rate of 44.1 kHz and had a 24-bit depth. It was not necessary to have a higher sample rate or bit depth as these are the standard settings used in the sound engineering community and the tests were designed to emulate a realistic scenario. The project files for each song consisted of more than eight separate WAV files. However, the Audio Mix Analyzer was built to sum a maximum of eight tracks. Consequently, eight of the most significant tracks were chosen for each song. The instruments and tracks chosen for each song are shown in Table 7.1. To save time, and bearing in mind that the samples were going to be used in aural tests, only 30 seconds of each song was recorded. An effort was made to select a section of the song where all eight of the tracks produced a signal.

Another aspect that could impact the difference between analogue and digital summing was the number of tracks summed. For this reason each song was summed four times - each time with the number of tracks shown below:

1. Two tracks
2. Three tracks

[1] http://www.bannedfromthezoo.com/
[2] https://soundcloud.com/animalmusicuk

Table 7.1: Eight tracks selected to be summed for each song

Song	**Black Out Betty**	**Catching Up**	**Rockshow**
1	Snare Up	Acoustic Guitar 2	Synth
2	Electric Guitar 1	Snare Up 1	Bass Amp
3	Lead Vocals 1	Electric Guitar 1	Decks 5
4	Lead Vocals 2	Lead Vocals 1	Snare Up 1
5	Bass Amp	Bass DI	Lead Vocals 1
6	Kick In	Kick In 1	Kick In 1
7	Kick Out	Kick Out 2	Kick Out 1
8	Snare Down	Bass Amp	Hihat

3. Five tracks

4. Eight tracks

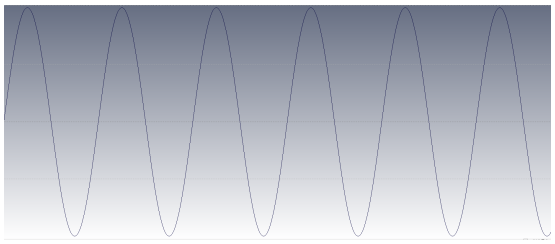
The order of the tracks in Table 7.1 reflects the order in which they were summed. For example, the three tracks summed for Catching Up were the Acoustic Guitar, Snare Up and Electric Guitar. After all of these tracks had been recorded, they were loaded into the Audio Mix Analyzer and the gain was then adjusted to ensure that the RMS of each track was between -14.94 dB and -15.04 dB.

Once both digital and analogue sums of the three songs were recorded, the Audio Mix Analyzer was used to sum the tracks using the digital mixing algorithms implemented in Chapter 6. The panning of the tracks was not altered in any way but the gain was modified until the RMS was in the designated range. The songs were also summed four times with an increasing number of tracks, as previously described, so that any comparison could be made between the summing methods. The execution of the aural tests is described in Section 7.3.

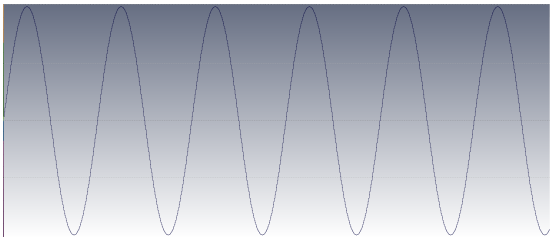
7.2 Visual Testing

7.2.1 Sine Waves

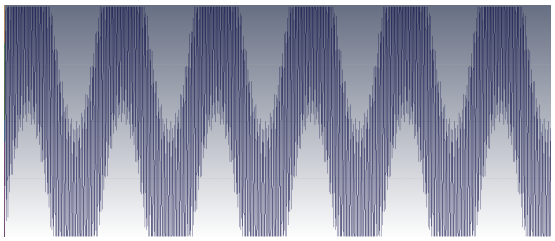
The previously summed and recorded sine waves were opened in Wavelab 6.11 for analysis. Although many different waves were recorded, time only permitted for the in-depth study of...



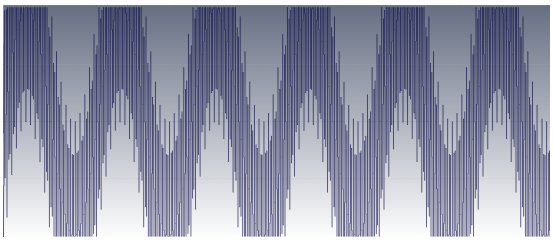
(a) Digital 300Hz wave sampled at 96 kHz



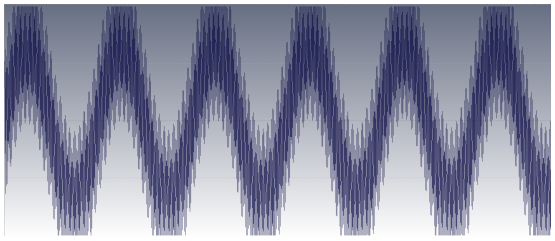
(b) Digital 300Hz wave sampled at 48 kHz



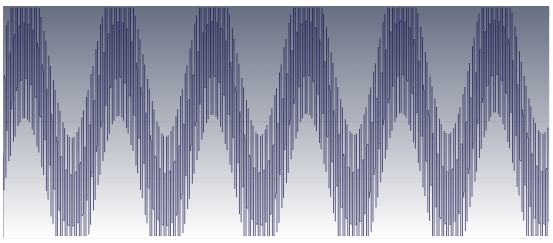
(c) Digital 30kHz + 300Hz sum sampled at 96 kHz



(d) Digital 30kHz + 300Hz sum sampled at 48 kHz



(e) Analogue 30kHz + 300Hz sum sampled at 96 kHz



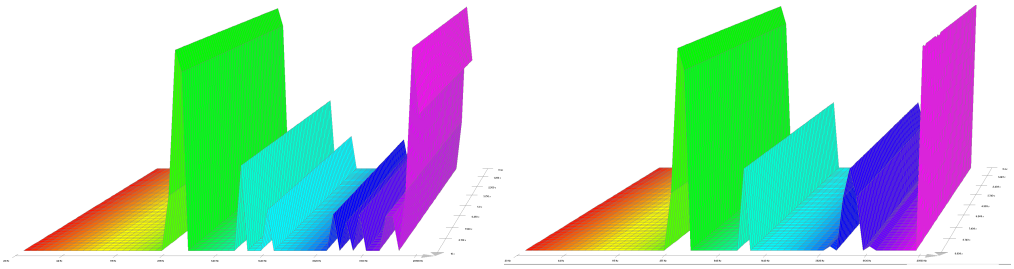
(f) Analogue 30kHz + 300Hz sum sampled at 48 kHz

Figure 7.1: Waveforms of Various Sine Waves

In Fig. 7.1a and Fig. 7.1b a 300Hz sine wave is presented at both sample rates. The digital sum of the 300Hz wave and a 30kHz sine wave is then shown in Fig. 7.1c and Fig. 7.1d, also at both sample rates. Lastly, the analogue sum of the same waves is displayed in Fig. 7.1e and Fig. 7.1f.

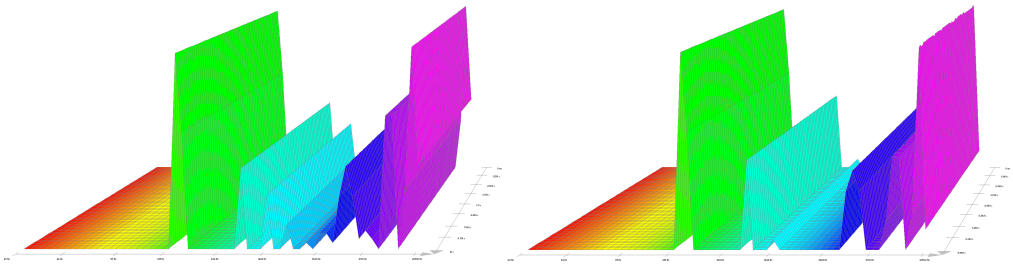
By comparing these images it immediately becomes apparent that a 30kHz sine wave affects a 300Hz wave when they are added together, regardless of summing method or sample rate. However, the waves samples at 96 kHz are definitely more affected by the addition of the 30kHz wave. And the analogue sums are even more affected...

Unfortunately these images can not give us enough information to draw a conclusion about the audible differences between them. Therefore a frequency analysis was conducted on the summed waves in Wavelab. Only an analysis between 20 Hz and 20 kHz was available. However this is the range of audible sound, which is what is of interest. The frequency spectrum for the 300Hz wave added to the 30kHz wave using digital and analogue methods at 32-bit and sample rates of 48 kHz and 96 kHz are shown below.



(a) Digital Sum (b) Analogue Sum

Figure 7.2: Frequencies of a 30kHz + 300Hz wave sampled at 96 kHz



(a) Digital Sum (b) Analogue Sum

Figure 7.3: Frequencies of a 30kHz + 300Hz wave sampled at 48 kHz

Looking at Fig. 7.2 and Fig. 7.3 it is clear that all four waves consist of slightly different frequencies. It is apparent that both digital sums contain more mid-range frequencies than the analogue sums. However, the digital

sum at 48 kHz contains additional frequencies between 1000 Hz and 3000 Hz that do not appear in the 96 kHz sum. These are frequencies that are humans are very receptive to (see Section 2.3) and could definitely cause the 48 kHz wave to sound less clear.

Conventional songs are not made up of pure sine waves, therefore it would be interesting to compare the waveforms of the audio tracks being used in the listening tests.

7.2.2 Audio Tracks

The User Evaluation tab of the Audio Mix Analyzer displays the waveform of loaded files. This provides an easy way to visually compare the summed waves of the different algorithms. Fig. 7.4 contains the waveforms of the eight 'Rockshow' tracks summed using all ten methods (i.e. the eight digital algorithms and the Mackie and Cubase sums).

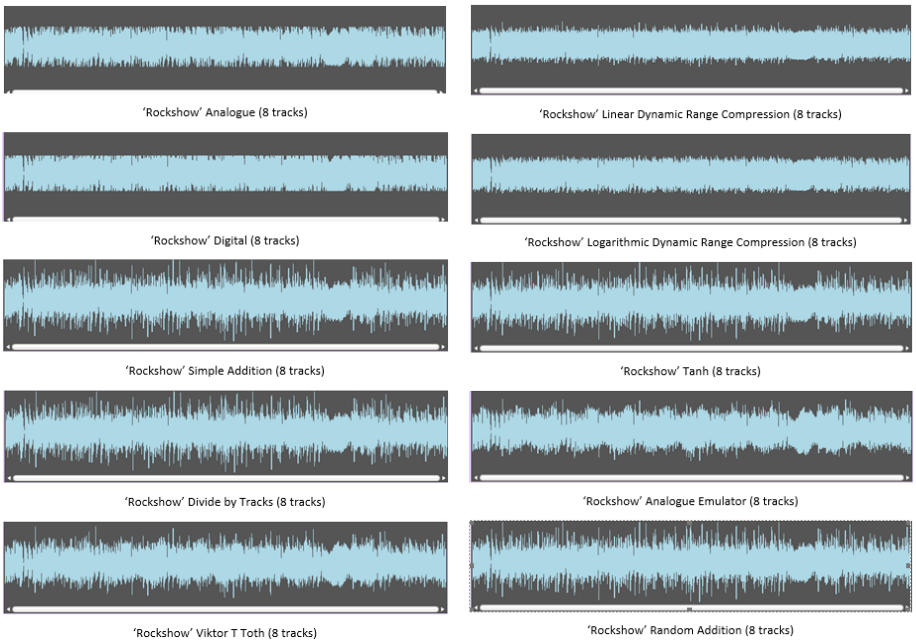


Figure 7.4: Waveforms of 'Black Out Betty' Sum for Different Algorithms

Although some of the waveforms look extremely different, upon closer inspection it can be seen that some waves are simply more compressed than

others. For example, the Analogue and Simple Addition waveforms both contain peaks and troughs in the same positions. The Analogue wave is just more compressed, which could be from the adjustment of the RMS after the recording of the wave. The Digital waveform is even more compressed than this. Now, aural testing should be done to determine whether the visual differences in the waveforms affect the overall sound of the waves.

7.3 Aural Testing

In Section 7.1.2 the process of recording and summing a large number of audio tracks was described. From these tracks a few were selected to be used in the listening tests. Participants were then chosen to listen and compare the tracks, as well as provide feedback. The listening tests took place with appropriate audio equipment and the resulting data was analyzed.

7.3.1 Test configuration

Many factors influenced the selection of the samples used in the listening tests. As each test would have to be completed individually there was a limit to the number of samples that could be used. Hence only the most important comparisons could be chosen. In an attempt to restrict the time needed for each test session to under 30 minutes, only six tests per song were selected, resulting in a total of 18 tests. As the objective of the thesis is to conclude whether there is a difference in analogue and digital summing, it was decided that an analogue sample should be compared to a digital sample for every test. For each song a control test was used, where the analogue sum of all eight tracks was used for both samples. This was to establish whether a participant could tell when there was no difference between two samples.

Another essential test was the comparison of the digital and analogue sums of all eight tracks for the three songs. Two more tests compared the analogue and digital sums of a smaller number of tracks. Test 4, for example,

compared the digital sum of five tracks from 'Black Out Betty' to the analogue sum of these five tracks. These tests could indicate if increasing the number of tracks being summed makes the difference in the summing more obvious.

This only left two tests per song to compare the analogue sum to any of the digital mixing algorithms. The most logical decision was to compare the Simple Addition algorithm for every song and then select three other algorithms to be used for the last three tests. For clarity, the six comparisons per song are shown below:

1. Analogue vs Analogue (8 tracks)
2. Analogue vs. Digital (8 tracks)
3. Analogue vs Digital (5 tracks)
4. Analogue vs Digital (2 or 3 tracks)
5. Analogue vs Simple Addition (8 tracks)
6. Analogue vs Digital Mixing Algorithm (8 tracks)

All digital algorithm audio samples were the sum of eight tracks to ensure the maximum difference, if any. Once the tests themselves were selected, the order in which the tests would be asked per song and the classification of each sample as 'A' or 'B' was randomized. However, the exact same order and classification of samples was used for every session. The list of tests can be seen in Appendix C.1.

As can be seen in Appendix C.1 there are 19 tests but only 18 tests have been mentioned. The 19th test was added after the first listening test had taken place. This was due to a suspicion that comparing the digital sum to a sum using the Simple Addition algorithm would produce an interesting result. This is discussed further in Section 7.3.4.

7.3.2 Participants and Completion of Testing

An expert panel of 13 participants, between the ages of 18 and 55, was selected to take part in the listening tests. The sample size of participants needed to be large enough to analyze the results, but small enough to be manageable, taking into consideration the time constraints. An ideal number of participants was thought to be between 10 and 15 people. Participants were selected based on their music expertise and capabilities, from self taught musicians to a sound engineer with over 20 years of experience. Approval was obtained from the Rhodes University Computer Science Ethics Board (tracking number CS14-14).

Before the listening tests began, participants were told that they would be listening to two audio samples and would be attempting to hear differences between the two. They were also told that sometimes the two samples would be identical and therefore there would be no difference for those tests. However, they were not told how many tests would have identical samples.

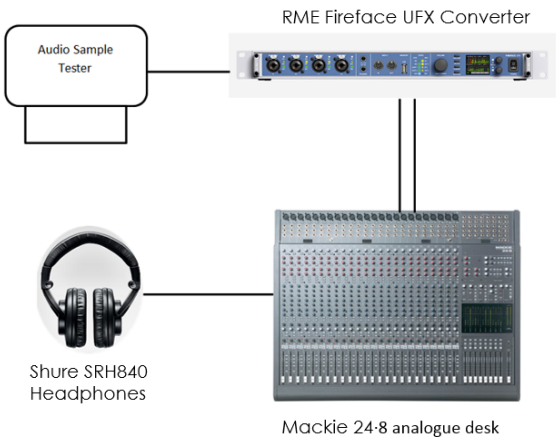


Figure 7.5: Listening Test Equipment Setup

An effort was made to guarantee that the setup was identical for each participant. Audio was played out from the Audio Mix Analyzer to the Mackie desk via the RME Converter. Participants listened to the tracks through Shure SRH840 headphones (36), which were connected to outputs on the Mackie desk. An illustration of the setup is shown in Fig. 7.5. The

same audio level was used throughout all of the listening tests to prevent differences in perception of frequencies (see Section 2.3).

Preventative measures were also taken to ensure participants did not know which summing method had been used for each sample. The names and RMS values of the samples were covered (with masking tape), as can be seen in Fig. 7.6. Participants were also asked not to allow the visual representation of the samples to influence their responses.

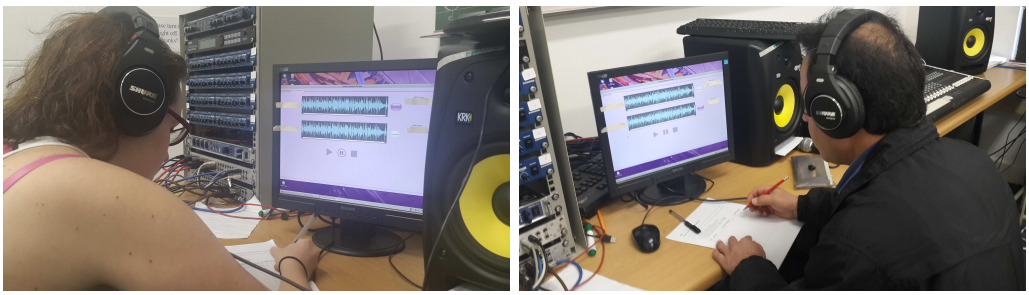


Figure 7.6: Two participants completing the Listening Tests

During the testing, each participant was asked to answer four questions on the response sheet (see Appendix C.2) for each of the 19 tests. The main objectives of the test were to determine whether differences between tracks could be heard and, if so, how clear that difference was. An opportunity to provide any other feedback about the difference was also needed. For this reason, the following questions were asked:

1. Is there a difference between the samples?
2. How you would describe this difference?
3. How would you rate the clarity of Sample A, from 1 (clear) to 5 (muffled).
4. How would you rate the clarity of Sample B, from 1 (clear) to 5 (muffled).

After all 19 tests, each participant was allowed to see the Audio Test Configuration (Appendix C.1) and compare this with their own response sheet for their personal interest.

7.3.3 Results

Table 7.2: Summary of Responses per Test

					Difference		
Test	**Song**	**Tracks**	**Sample A**	**Sample B**	**Yes**	**No**	**Clarity**
1	BOB	8	Analogue	Digital	13	0	-2.692
2	BOB	8	Analogue	Simple Addition	6	7	0
3	BOB	8	Logarithmic	Analogue	8	5	0.1538
4	BOB	5	Digital	Analogue	12	1	-1.077
5	BOB	8	Analogue	Analogue	5	8	0.1667
6	BOB	3	Analogue	Digital	13	0	1.2692
7	CU	8	Simple Addition	Analogue	4	9	0.1538
8	CU	8	Analogue	Digital	13	0	-1.538
9	CU	8	Analogue	Analogue	4	9	0.0385
10	CU	3	Digital	Analogue	11	2	-0.923
11	CU	8	Analogue	Tanh	5	8	-0.231
12	CU	5	Analogue	Digital	12	1	0.7308
13	RS	5	Digital	Analogue	5	8	-0.333
14	RS	8	Analogue	Analogue	1	12	-0.091
15	RS	8	Analogue	Digital	13	0	1.5385
16	RS	8	Analogue	Simple Addition	13	0	1.6154
17	RS	8	Analogue	Divide by Tracks	13	0	2.1923
18	RS	2	Analogue	Digital	6	7	-0.038
19	RS	8	Simple Addition	Digital	5	7	-0.364

After all participants had completed the listening tests the results of questions 1, 3 and 4 were recorded. The raw data are presented in Appendix C.3.2. Table C.3 contains the participants' answer to whether they hear a difference between samples, ordered by age. Table C.4 shows the participants' rating of the clarity of each track in response to questions 3 and 4, also ordered by age. As stated in the previous section, a lower value indicates a clear sound while a high value means that the track sounded more muffled. The absolute difference in clarity for each test was was found by subtracting the analogue rating from the digital rating, as shown in Eq. (7.1)[3]. The average of these values was then calculated to get a 'Clarity' value. These results are given in Table C.5, as well as with the average value for each test. A summary of this data, next to the configuration of each test

[3]In the case of test 19 the digital rating was subtracted from the Simple Addition rating.

to provide context, can be found in Table 7.2.

7.3.4 Analysis

Observation of Sample Differences

The first result worth noting is the number of differences heard between samples. Fig. 7.7 shows the number of 'Yes' and 'No' responses to question 1 for every test. The percentage of participants that heard a difference between samples for each of the six comparisons were calculated and can be seen in Table 7.3.

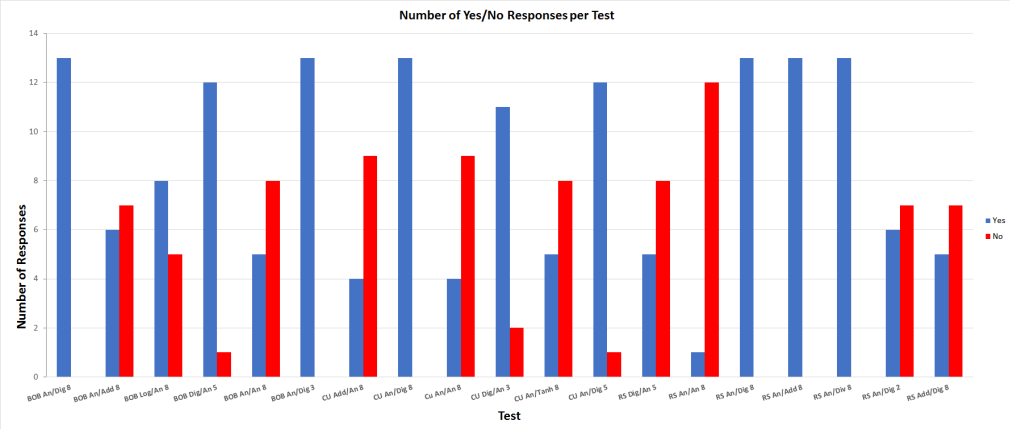


Figure 7.7: Graph showing Question 1 Responses[4]

These figures indicate some interesting findings. First, every participant heard a difference between the analogue and digital eight-track samples for all three songs. In fact, over 70% of participants could hear a difference between all of the analogue and digital samples, for any number of tracks. Only two-thirds of participants heard a difference between the analogue sample and one of the digital mixing algorithms, which would imply that these are slightly better summing algorithms. However, although the same track was used in the Analogue vs. Analogue comparisons 25.64% of participants still heard a difference. The 'Best vs Simple Addition' comparison will be discussed after an analysis of the responses regarding clarity.

[4]Tests are labeled using the format [Song Name] [Sample A/Sample B] [Number of Tracks]

Table 7.3: Percentage of Participants Hearing a Difference for each Algorithm

Tracks	Test	Yes
8	Analogue vs Analogue	25.64%
8	Analogue vs Digital	100.00%
5	Analgue vs Digital	74.36%
2 or 3	Analgue vs Digital	76.92%
8	Analogue vs Other	66.67%
8	Best vs Simple Addition	39.47%

Description of Difference

The participants were also given the option of describing the differences they heard. Many commented on noticeable difference in frequencies between the samples. For example, one participant noted that "the higher frequencies on track A are more prominent" for the first test and another said that "A sounds more brittle and (has) too much high frequency content" for test eight. Some participants also noticed a difference in the panning between samples, even though none had been applied. For example, test eight was said to have "better spatial content" by one participant while another thought track B had "more highs and stereo image" for test 15. All of these examples are from comparisons between analogue and digitally summed tracks. This is very interesting, as the summing process obviously affects the perceived frequency content and stereo sound of a song.

A summary of the general observations made for significant tests is given in Table 7.4.

Table 7.4: General descriptions of differences made by participants

Test	Comments
1	A has more prominent high frequencies, while B has more bass
6	B has higher frequencies
7	A has more bass
8	B sounds warmer, although A is clearer
10	B sounds warmer, although A is clearer
12	B has more high frequencies and stereo image
15	B has more high frequencies and is clearer.

Difference in Clarity

Clearly, participants could hear differences between many of the samples but these results can not reveal which of the two samples sounds clearer. Using the summary of responses (Table 7.2), a graph showing the average difference in clarity rating between the two samples for each algorithm was generated. This is shown in Fig. 7.8. This graph shows a very interesting phenomena when observing the values for each song. As previously stated, the clarity values were evaluated by subtracting the digital rating from the analogue rating. A positive value, therefore, means that the participants thought that the analogue sample sounded clearer, while a negative value implies the digital sample was clearer.

$$Clarity = (DigitalRating - AnalogueRating)/Responses \tag{7.1}$$

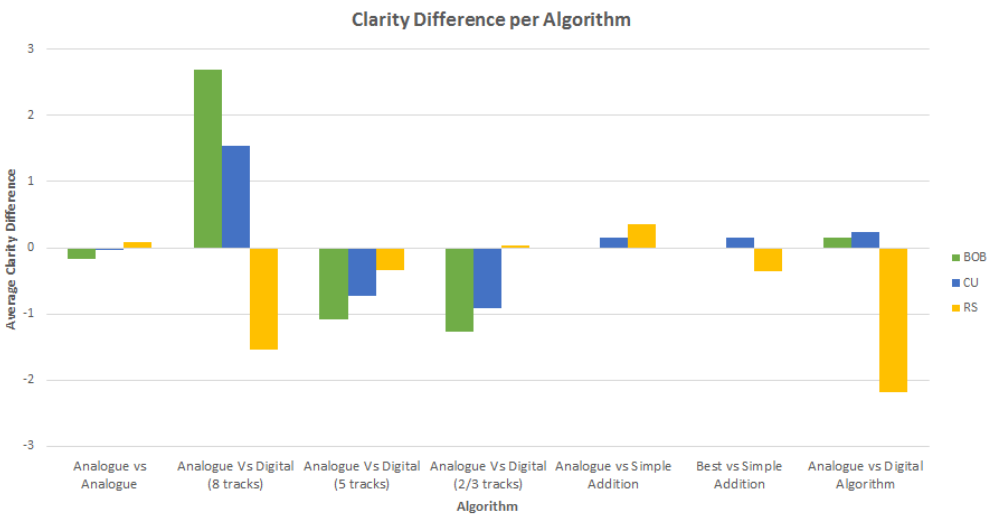


Figure 7.8: Graph showing Clarity Difference per Algorithm

Figure 7.8 consequently seems to suggest that the analogue summed tracks sounded much clearer for 'Black Out Betty' and 'Catching Up', the Pop/Rock and Singer-Songwriter songs, while the digital sum was clearer for 'Rock-show', the Hip-Hop/Electronic song. This would imply that the best summing method is highly dependent on the genre of the song.

As stated in Section 7.3.1, this occurrence was spotted after the first listening test, resulting in the addition of a 19th test. It was suspected that

the Simple Addition algorithm would be the most similar to the 'clearer' sample which is why the 19th test is a comparison between the Digital and Simple Addition algorithms. In Table 7.3 and Fig. 7.8 the algorithm called 'Best' corresponds to the Digital sum for 'Rockshow' and to the Analogue sum for 'Black Out Betty' and 'Catching Up'.

The graph also illustrates that, although about $\frac{1}{4}$ of participants heard a difference between the identical analogue samples, they could not hear a big difference in clarity. This helps the establishment of validity for this set of listening tests.

Another interesting observation is the impact that the number of tracks has on the clarity. The digital sum was perceived to be more clear when only five tracks were summed, irrespective of the song. This was also the case for when three tracks were summed. The comparison of two summed tracks from 'Rockshow' seems to be an exception to this pattern, as participants noticed the analogue sum to be slightly clearer than the digital sum. Overall, the number of tracks definitely appears to affect the clarity of the summed tracks.

The results for the other digital algorithms used can only be analyzed for the specific song they were used with. Both the Logarithmic Compression and Tanh algorithms, used for 'Black Out Betty' and 'Catching Up' respectively, have slightly higher clarity differences when compared to the Simple Addition algorithm. This means that the Simple Addition algorithm sounds slightly clearer and closer to the sound of analogue summing.

Now the Analogue and Divide by Tracks comparison for 'Rockshow' shows that the Divide by Tracks algorithm sounded much clearer. This is to be expected, given the previous genre-based results. However, the clarity difference seems to be slightly larger than the difference between the analogue and digital comparison of eight tracks. This seems to indicate that the Divide by Tracks algorithm could sound marginally clearer than the Cubase sum or, given the margin of error, Cubase could be using the Divide by Tracks. This would explain the similarity in clarity in this case.

One last relationship to investigate is that between the number of people who heard a difference between samples and the clarity rating. Fig. 7.9

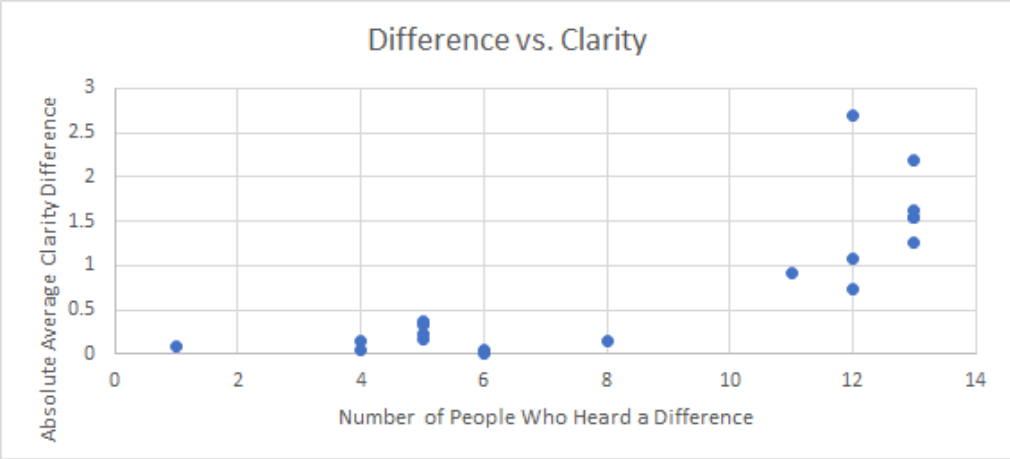


Figure 7.9: Graph showing Correlation between Difference and Clarity

graphs the correlation between the people who responded 'Yes' to question 1 and the absolute average of difference in clarity. The relationship appears to be approximately linear. This means that most participants associated a difference in samples with a difference in clarity.

7.3.5 Testing Limitations

In hindsight the listening test process definitely had some limitations. Although every effort was made to ensure that all tests was identical, the nature of individual testing and makes it impossible to guarantee that external factors did not affect the listening tests.

Many participants seemed to get frustrated at having to repeatedly listen to the same song and noted that they were not sure if the difference they were hearing actually existed or if it was psychological. Changing the configuration of the tests every time could have eliminated the influence the of order in which comparisons were asked on the results. It is also important to note that listening tests are inherently subjective and the difference between audio samples can be difficult to quantify. Therefore the interpretation of the term 'clarity' was left up to each participant.

Due to time constraints a bigger sample size of participants was not possible. Human error also plays a significant role in the outcome of the tests,

both with the participant's ability to distinguish sounds and the recording of the results.

7.4 Analogue Mixing Console Experiments

During the discussion of creating digital algorithms to emulate analogue summing it was suggested that the summing of the Mackie desk be tested. The desk is over 20 years old and it also seemed necessary to confirm that it was summing as stated in the Mackie manual. Mono sine waves of were generated at 300 Hz and 1 000 Hz in Wavelab 6.11. They were played out through Cubase Artist 6.5 into the Mackie desk via the RME Fireface UFX converter. As shown in Fig. 7.10 and Fig. 7.11 a Tektronix TDS1001B Portable Digital Oscilloscope was used to measure the signals.

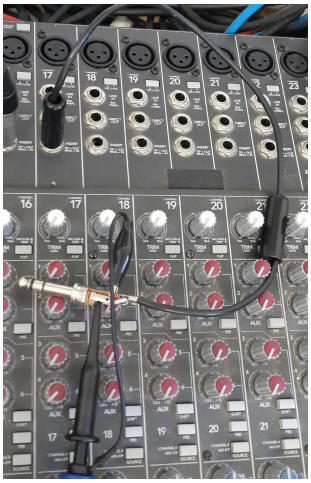


Figure 7.10: Input voltage being measured on Channel 17

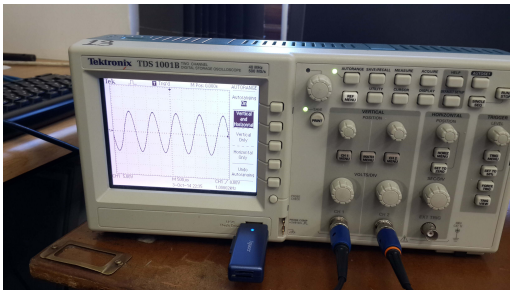


Figure 7.11: Oscilloscope output

The configuration of the RME only allowed access to channels 17 to 24 on the desk. Unfortunately, these were not the channels that the tracks for the visual and aural testing had been recorded through. However, it was assumed that the eight channels available would be representative of the entire 24-channel desk. As specified by Mackie, the desk has "an active buffered constant power pan control (-3dB @ center)" (28, p. 2). When

mono waves are summed, "the center image remains at the same level while the extremes drop by 3 dB" Izhaki (22, p. 190).

To start with, two 1000Hz waves were played out of the DAW into channels 17 and 18. The voltages of the waves were measured at the channel inserts. The output volume of the DAW was adjusted until the peak-to-peak voltage for channel 17 was 9 V. This was used as the reference voltage. Each individual track was assigned to the main mix and the signal was measured at the main mix output with both center-panning and hard-panning. Both channels were then assigned to the main mix and the summed value was recorded for both panning configurations. This set of experiments was repeated with a reference value of 5 V. The results in volts are summarized in Table 7.5.

Table 7.5: Output voltage of Two 1000Hz Waves

Channels	**Inserts**	**Center-panned output**	**Hard-panned output**
17	9	6.64	9
18	9	6.64	9
17 + 18	N/A	13.2	17
17	5	3.76	5
18	5	3.76	5
17 + 18	N/A	7.6	10.2

These values are all in volts so a conversion to deciBels is needed for analysis. The equation to convert voltage gain to decibels is given in Eq. (7.2) (from Appendix A.2). The converted values can be found in Table 7.6.

$$db = 20 * log_{10}(gain) \tag{7.2}$$

As a -3dB panning law is used by Mackie, we would expect the center-panned output to be 3dB less than the input values and but would predict that the hard-panned output would remain the same. The individual channel output measurements were very similar to what was expected with a -3 dB panning law. Although the hard-panned results were exactly as expected, the center-panned values were slightly larger than it was presumed they would be. An interesting result occurred with summed signal measurements.

Table 7.6: Actual and Expected Output of Two 1000Hz Waves (dB)

		Center-panned		**Hard-panned**	
Channels	**Inserts**	**Expected**	**Actual**	**Expected**	**Actual**
17	19.08	16.08	16.44	19.08	19.08
18	19.08	16.08	16.44	19.08	19.08
17 + 18	N/A	32.17	22.41	38.17	24.61
17	13.98	10.98	11.50	13.98	13.98
18	13.98	10.98	11.50	13.98	13.98
17 + 18	N/A	21.96	17.62	27.96	20.17

The voltage outputs in Table 7.5 are the exact sum of the individual channel outputs. However, when converted to deciBels, this does not remain true. The summed values are much lower than they were expected to be. It is clear that the -3 dB pan law only applies to individual channels when center-panned on the Mackie desk and signals are summed by adding the voltage values together.

After this, it was decided that the consistency of all eight channels should be tested. A 300Hz wave was played to channels 17 to 24 at the same output volume. The voltage was measured at the insert of every channel and recorded in Table 7.7. It becomes immediately apparent that the values vary greatly, from 5.8 V to 9.4 V for the exact same signal. This appears to be very problematic, as the overall volume of an individual track could be affected by the channel of the desk being used.

Table 7.7: Voltage output of a 300Hz wave for Channels 17 - 24

Channel	**Insert (V)**
17	8.6
18	8.6
19	9.4
20	9.4
21	5.8
22	9.4
23	9.3
24	9.4

7.5 Timing Results

The digital mixing process implemented by the Audio Mix Analyzer is all executed statically. If the mixing was to take place in real-time, the program would need to sum the samples extremely quickly, as explained in Section 4.5. For a samples rate of 44.1 kHz samples would need to be summed in under 22.68 μs (see Table 4.1). To see if the Audio Mix Analyzer could achieve this, timing tests were run on every mixing algorithm.

```
Time to mix using Linear Dynamic Range Compression algorithm: 1853 ms
Time to mix using Linear Dynamic Range Compression algorithm: 1867 ms
Time to mix using Linear Dynamic Range Compression algorithm: 1866 ms
Time to mix using Logarithmic Dynamic Range Compression algorithm: 1913 ms
Time to mix using Logarithmic Dynamic Range Compression algorithm: 1902 ms
Time to mix using Logarithmic Dynamic Range Compression algorithm: 1892 ms
Time to mix using Tan h algorithm: 3330 ms
Time to mix using Tan h algorithm: 3341 ms
Time to mix using Tan h algorithm: 3337 ms
Time to mix using Analogue emulator algorithm: 2962 ms
Time to mix using Analogue emulator algorithm: 2963 ms
Time to mix using Analogue emulator algorithm: 2974 ms
Time to mix using Random analogue algorithm: 2582 ms
Time to mix using Random analogue algorithm: 2595 ms
Time to mix using Random analogue algorithm: 2605 ms
```

Figure 7.12: Screenshot of Timing Tests for 'Rockshow'

Using the code shown in Appendix A.6, the eight tracks from each song used in the listening tests were summed with each algorithm. The time taken to sum the tracks was documented. An example of this can be seen in Fig. 7.12. Each test was repeated three times and the average time taken for each method per song is shown in Table 7.8. The average times per method were also calculated and, from these values, we can conclude that the Divide by Tracks algorithm was the quickest while Viktor T. Toth's algorithm took the longest to sum the samples.

Although the values in Table 7.8 are interesting when comparing algorithms, they do not indicate how long it takes to calculate a single sample of the summed audio file. To compute this value, the number of samples in each track was needed. Since each track was 30 seconds long and sampled at 44.1 kHz it would be assumed that there were 1 323 000 samples per track. A simple check during the execution of the program confirmed that this was indeed the case. Therefore the values in Table 7.8 were divided by 1 323 000 to calculate the time it took to sum eight samples - or the time taken to determine a single sample of the summed audio track. These times can be found in Table 7.9 (given in microseconds).

Table 7.8: Timing Test Results in milliseconds

Algorithm	Addition	Divide by Tracks	Viktor T Toth	Linear Compression
Black Out Betty	846.3	853.0	2220.7	857.3
Catching Up	1209.0	886.7	2234.7	884.0
Rockshow	2601.3	1858.7	4791.3	1862.0
Average	1552.2	1199.4	3082.2	1201.1
Algorithm	Random	Tan h	Emulator	Log Compression
Black Out Betty	1194.7	1527.0	1371.3	880.3
Catching Up	1228.3	1549.3	1392.7	894.3
Rockshow	2594.0	3336.0	2966.3	1902.3
Average	1672.3	2137.4	1910.1	1225.7

Table 7.9: Timing per Sample Test Results in microseconds

Algorithm	Addition	Divide by Tracks	Viktor T Toth	Linear Compression
Black Out Betty	0.640	0.645	1.679	0.648
Catching Up	0.914	0.670	1.689	0.668
RockShow	1.966	1.405	3.622	1.407
Average	1.173	0.907	2.330	0.908
Algorithm	Random	Tan h	Emulator	Log Compression
Black Out Betty	0.903	1.154	1.037	0.665
Catching Up	0.928	1.171	1.053	0.676
RockShow	1.961	2.522	2.242	1.438
Average	1.264	1.616	1.444	0.926

The longest it took to sum eight samples was 2.33 μs using the Viktor T Toth algorithm. From Table 4.1 we know that we have 22.68 μs to sum tracks sampled at 44.1 kHz in real-time. Evidently any of the digital algorithms could be used to sample eight tracks in real-time, even if they were sampled at 96 kHz. However, sound engineers typically sum more than eight tracks when mixing down conventional recordings. Then again, the times taken to sum the eight samples are low enough that the Simple Addition algorithm could be used to sum over 144 tracks sampled at 44.1 kHz in real-time. From these calculations it can be concluded that it is realistically feasible to use of the described digital mixing algorithms to sum a large number of tracks in real-time.

7.6 Summary

This chapter gave an overview of the listening and visual feedback. The recording of sine waves and audio tracks was described. This was followed by the visual testing of summed sine waves and the audio tracks summed using various mixing algorithms. Many differences could clearly be seen, especially when different sample rates were used to sum sine waves. The aural testing was detailed, from the configuration of the testing to the completion of listening tests by the participants. The results of the testing were provided and these were then analyzed. It was found that the clearer summing method depended on the genre of the audio tracks and the number of tracks being summed. Analogue experiments and timing tests were also performed and it was found that the digital algorithms could be used in real-time scenarios.

Chapter 8

Conclusion

8.1 Reiteration of Hypothesis

As indicated in Chapter 1, this study began with an indication from audio engineers that the sound produced by the analogue summing process differs from the sound created by digital summing. The hypothesis of the project was that a mechanism to provide the characteristics of an analogue mixer could be found and implemented in a digital mix. This was explored in three steps:

1. Firstly, it was definitively proven that a visual and audible sound difference caused by the summing method exists.
2. Secondly, an Audio Mix Analyzer was created to facilitate the comparison of audio samples, as well as allowing users to control the selection of the digital mixing algorithm to be used.
3. Lastly, several mixing algorithms were programmed in an attempt to emulate the sound of an analogue mix. The results of the listening tests indicated that the Simple Addition algorithm achieved this.

8.2 Observations

During the listening tests all participants noticed a difference between the analogue and digital summing of eight audio tracks, with the clearer summing method being genre dependent. About $\frac{3}{4}$ of participants noticed a difference between analogue and digital summing of two, three and five tracks. Depending on the genre, the method that sounded clearer for eight tracks was often reversed when fewer tracks were summed. This proves that the clearer summing method depends on the number of tracks being summed.

Various mixing algorithms were implemented in the Audio Mix Analyzer and, after testing, it was found that about 60% of the participants could not hear a difference between the Simple Addition algorithm and analogue summing. These results, which corroborated the study of the open source DAWs and the experiments with an analogue mixing console, all indicate that simply adding the samples together is the summing method most similar to the analogue summing. However, from the results of the listening tests, it appears that Cubase, the closed-source DAW that was used to digitally sum audio tracks, is not using this approach.

The Audio Mix Analyzer interface enabled both the efficient creation of digitally-summed audio tracks and allowed users to easily compare two audio files during the listening tests. Timing tests were performed on all digital mixing algorithms and it was found that they could be implemented in a real-time scenario.

Overall it appears that there are many possible causes for the difference heard when different summing techniques are used. The investigation into sampling rates by means of visual tests proved that additional frequencies are present in digital audio when samples rates of 48 kHz and lower are used. It is therefore suggested that sample rates above the industry standard rate of 44.1 kHz could be used in digital summing to attain a summing quality more analogous to analogue summing.

8.3 Future Work

It is clear that much more work in the field of audio summing could be performed. The investigation into the effect of panning was beyond the scope of this project. This topic could be explored further to fully explain the variation in panning that can be heard when different summing algorithms are used.

The summing methods of Cubase and other closed-source DAWs could also be investigated by testing the output values when summing multiple tracks.

From the visual testing it can be seen that further investigation into the audible differences of various sampling rates is required. Additional listening tests could also be performed with some of the digital mixing algorithms that were not tested.

References

[1] Algorithm(s) to mix audio signals without clipping. Online. Accessed on: 30 October 2014. Available from: http://dsp.stackexchange.com/questions/3581/algorithms-to-mix-audio-signals-without-clipping.

[2] C Plus Plus. Online. Accessed on: 20 October 2014. Available from: http://www.cplusplus.com/.

[3] Cubase. Online. Accessed on: 17 October 2014. Available from: http://www.steinberg.net/en/products/cubase/.

[4] Fireface UFX. Online. Accessed on: 21 October 2014. Available from: http://www.rme-audio.de/en_products_fireface_ufx.php.

[5] The Intojucer. Online. Accessed on: 30 October 2014. Available from: http://www.juce.com/documentation/introjucer.

[6] JUCE Features. Online. Accessed on: 20 October 2014. Available from: http://www.juce.com/features.

[7] The tanh() function. Online. Accessed on: 30 October 2014. Available from: http://folk.ntnu.no/oyvinbra/gdsp/Lesson4tanh.html.

[8] Visual Studio. Online. Accessed on: 30 October 2014. Available from: http://www.visualstudio.com/products/visual-studio-with-msdn-overview-vs.

[9] Wavelab. Online. Accessed on: 20 October 2014. Available from: https://www.steinberg.net/en/products/wavelab/.

[10] Why is Tanh commonly used in digital audio distortion implementations? Online. Accessed on: 30 October 2014. Available from: http://www.quora.com/Why-is-Tanh-commonly-used-in-digital-audio-distortion-implementations.

[11] Aarts, R., Engel, J., Reefman, D., Usher, J., and Woszczyk, W. Which of the two digital audio systems best matches the quality of the analog system? In *Audio Engineering Society Conference: 31st International Conference: New Directions in High Resolution Audio* (Jun 2007).

[12] Agilent Technologies. The Fundamentals of Signal Analysis: Application Note 243. Online, 06 2000. Accessed on: 20 May 2014. Available from: http://cp.literature.agilent.com/litweb/pdf/5952-8898E.pdf.

[13] Ardour. Ardour Website. Online, 2014. Accessed on: 25 May 2014. Available from: http://ardour.org/.

[14] Audacity. Audacity Meter Toolbar. Online, 2014. Accessed on: 26 May 2014. Available from: http://manual.audacityteam.org/o/man/meter_toolbar.html.

[15] Audacity. Audacity Website. Online, 2014. Accessed on: 25 May 2014. Available from: http://audacity.sourceforge.net/.

[16] Bartlett, B. *Practical Recording Techniques: The Step- by- Step Approach to Professional Audio Recording*. Taylor & Francis, 2013.

[17] Cochrane, G. Analog Summing And Why You Shouldnt Care. Online, 2012. Accessed on: 20 May 2014. Available from: http://therecordingrevolution.com/2012/05/04/analog-summing-and-why-you-shouldnt-care/.

[18] Cooper, P. Q. Is analogue mixing superior to digital summing? Sound on Sound, 06 2004. Accessed on: 28 February 2014. Available from: http://www.soundonsound.com/sos/jun04/articles/qa0604-5.htm.

[19] Cornell. Bit Depth. Online, 2009. Accessed on: 29 May 2014. Available from: `http://digital.music.cornell.edu/cemc/book/export/html/1594`.

[20] Farmelo, A. Analog vs. Digital Summing. Self-published, 2011. Accessed on: 28 February 2014. Available from: `http://www.farmelorecording.com/in-the-press/analog-vs-digital-summing/`.

[21] Griesinger, D. Stereo and surround panning in practice. In *Audio Engineering Society Convention 112* (Apr 2002).

[22] Izhaki, R. *Mixing Audio: Concepts, Practices and Tools*. Electronics & Electrical. Focal Press, 2008.

[23] Katz, B., and Katz, R. *Mastering Audio: The Art and the Science*. Focal Press, 2002.

[24] Kinsler, L. *Fundamentals of Acoustics*, 2nd ed. Wiley, 1962.

[25] Leonard, B., and Buttner-Schnirer, P. Subjective differences in digital audio workstation math. In *Audio Engineering Society Convention 132* (Apr 2012).

[26] Leonard, B., Levine, S., and Buttner-Schnirer, P. Objective and subjective evaluations of digital audio workstation summing. In *Audio Engineering Society Convention 133* (Oct 2012), p. 1.

[27] Lipshitz, S. P., and Vanderkooy, J. Pulse-code modulation–an overview. *J. Audio Eng. Soc 52*, 3 (2004), 200–215.

[28] Mackie. 8 Bus Architect's Specifications. Online. Accessed on: 21 October 2014. Available from: `http://www.mackie.com/pdf/8Bus_Arch.pdf`.

[29] Mackie. 8 Bus Specifications. Online. Accessed on: 16 October 2014. Available from: `http://www.mackie.com/products/8bus/8Busspecs.html`.

[30] MathWorld, W. Root-Mean-Square. Online, 2014. Accessed on: 29 October 2014. Available from: http://mathworld.wolfram.com/Root-Mean-Square.html.

[31] Peat, C. Online. Accessed on: 30 May 2014. Available from: http://www.heavens-above.com/glossary.aspx?term=azimuth.

[32] Rabiner, L. R. Digital techniques for changing the sampling rate of a signal. In *Audio Engineering Society Conference: 1st International Conference: Digital Audio* (Jun 1982), p. 80.

[33] Rosegarden. Rosegarden Website. Online, 2014. Accesed on: 25 May 2014. Available from: http://www.rosegardenmusic.com/.

[34] Rumsey, F., and McCormick, T. *Sound And Recording, An Introduction*. Focal Press, Jordan Hill, Oxford, 2004.

[35] Senior, M. The 'Mixing Secrets' Free Multitrack Download Library. Online. Accessed on: 20 October 2014. Available from: http://www.cambridge-mt.com/ms-mtk.htm.

[36] Shure. SRH840 Professional Monitoring Headphones. Online. Accessed on: 24 October 2014. Available from: http://www.shure.com/americas/products/earphones-headphones/headphones/srh840-professional-monitoring-headphones.

[37] Sound on Sound. Analogue Summing Mixers. Sound on Sound, 06 2012. Accessed on: 20 May 2014. Available from: http://www.soundonsound.com/sos/jun12/articles/spotlight-0612.htm.

[38] Storr, W. The Summing Amplifier. Online, 2014. Accessed on: 28 May 2014. Available from: http://www.electronics-tutorials.ws/opamp/opamp_4.html.

[39] Voegler, P. Mixing two digital audio streams with on the fly Loudness Normalization by Logarithmic Dynamic Range Compression. Self-published, 04 2012. Accessed on: 15 February 2014. Available from: http://voegler.eu/pub/audio/digital-audio-mixing-and-normalization.html#idx_3.

[40] Watkinson, J. *The art of digital audio*, third ed. Focal Press, 2001.

Appendix A

Code Listings

A.1 Audacity Code

Gain Representation from Meter.cpp

```
...
static float ToDB(float v, float range)
{
   double db;
   if (v > 0)
      db = 20 * log10(fabs(v));
   else
      db = -999;
   return ClipZeroToOne((db + range) / range);
}

double Meter::ToLinearIfDB(double value)
{
   if (mDB)
      value = pow(10.0, (-(1.0-value)*mDBRange)/20.0);
   return value;
}
```

Pan definition from ASlider.cpp

```
...
case PAN_SLIDER:
        minValue = -1.0f;
        maxValue = +1.0f;
        stepValue = 0.1f;
        orientation = wxHORIZONTAL;
        break;
...
```

Pan Algorithm from WaveTrack.cpp

```
float WaveTrack::GetChannelGain(int channel)
{
    float left = 1.0;
    float right = 1.0;
          ...
    if (mPan < 0)
        right = (mPan + 1.0);
    else if (mPan > 0)
        left = 1.0 - mPan;
    if ((channel%2) == 0)
        return left*mGain;
    else
        return right*mGain;
```

Summing Algorithm from Mix.cpp

```
void MixBuffers(int numChannels, int *channelFlags, float *gains,
                samplePtr src, samplePtr *dests,
                int len, bool interleaved)
{
          ...
        float gain = gains[c];
        float *dest = (float *)destPtr;
        float *temp = (float *)src;
```

```
        for (int j = 0; j < len; j++) {
            *dest += temp[j] * gain;    // the actual mixing process
            dest += skip;
        }
    }
```

A.2 Ardour Code

Gain Representation from dB.h

```
static inline float dB_to_coefficient (float dB) {
        return dB > -318.8f ? pow(10.0f, dB * 0.05f) : 0.0f;
}

static inline float accurate_coefficient_to_dB (float coeff) {
        return 20.0f * log10f (coeff);
}
```

Panner conversion from panner.h

```
static double azimuth_to_lr_fract (double azi) {
        /* 180.0 degrees=> left => 0.0 */
        /* 0.0 degrees => right => 1.0 */
        return 1.0 - (rint(azi)/180.0);
}
static double lr_fract_to_azimuth (double fract) {
        /* fract = 0.0 => degrees = 180.0 => left */
        /* fract = 1.0 => degrees = 0.0 => right */
        return rint (180.0 - (fract * 180.0));
}
```

Panning Algorithm from panner_2in2out.cc

Note: A very similar method can be found in panner_1in2out.cc

```
void
```

```
Panner2in2out::distribute_one (AudioBuffer& srcbuf,
BufferSet& obufs, gain_t gain_coeff,
pframes_t nframes, uint32_t which)
{
...
/* LEFT OUTPUT */
...
for (n = 0; n < limit; n++) {
        left_interp[which] = left_interp[which] + delta;
        left[which] = left_interp[which] +
        0.9 * (left[which] - left_interp[which]);
        dst[n] += src[n] * left[which] * gain_coeff;
}
pan = left[which] * gain_coeff;

mix_buffers_with_gain (dst+n,src+n,nframes-n,pan);
...
/* RIGHT OUTPUT */
...
for (n = 0; n < limit; n++) {
        right_interp[which] = right_interp[which] + delta;
        right[which] = right_interp[which] +
        0.9 * (right[which] - right_interp[which]);
        dst[n] += src[n] * right[which] * gain_coeff;
}
pan = right[which] * gain_coeff;
mix_buffers_with_gain(dst+n,src+n,nframes-n,pan);
}
```

Mixing Algorithm from mix.cc

```
...

void default_mix_buffers_with_gain (ARDOUR::Sample * dst,
                    const ARDOUR::Sample * src,
                    pframes_t nframes, float gain)
```

```
{
        for (pframes_t i = 0; i < nframes; i++) {
                dst[i] += src[i] * gain;
        }
}
void default_mix_buffers_no_gain (ARDOUR::Sample * dst,
                const ARDOUR::Sample * src, pframes_t nframes)
{
        for (pframes_t i=0; i < nframes; i++) {
                dst[i] += src[i];
        }
}
```

A.3 Rosegarden Code

Gain Representation from AudioLevel.cpp

```
...
const float AudioLevel::DB_FLOOR = -1000.0;
...
static const FaderDescription faderTypes[] = {
    FaderDescription(-40.0,  +6.0, 0.75), // short
    FaderDescription(-70.0, +10.0, 0.80), // long
    FaderDescription(-70.0,   0.0, 1.00), // IEC268
    FaderDescription(-70.0, +10.0, 0.80), // IEC268 long
    FaderDescription(-40.0,   0.0, 1.00), // preview
};
...
float
AudioLevel::multiplier_to_dB(float multiplier)
{
    if (multiplier == 0.0) return DB_FLOOR;
    float dB = 10 * log10f(multiplier);
    return dB;
```

```
}

float
AudioLevel::dB_to_multiplier(float dB)
{
    if (dB == DB_FLOOR) return 0.0;
    float m = powf(10.0, dB / 10.0);
    return m;
}
```

Panning Laws and Conversion from AudioLevel.cpp

```
int AudioLevel::m_panLaw = 0;
...
float
AudioLevel::panGainRight(float pan)
// Apply panning law to right channel
{
    if (m_panLaw == 3) {
                // -3dB pannig law (variant)
        return sqrtf(fabsf((100.0 + pan) / 100.0));

    } else if (m_panLaw == 2) {
                // -6dB pan law
        return (100.0 + pan) / 200.0;

    } else if (m_panLaw == 1) {
                // -3dB panning law
        return sqrtf(fabsf((100.0 + pan) / 200.0));

    } else {
                // 0dB panning law (default)
        return (pan < 0.0) ? (100.0 + pan) / 100.0 : 1.0;

    }
}
```

Applying Gain and Panning to Samples from AudioProcess.cpp

```
float gain[2];
gain[0] = rec.gainLeft;
gain[1] = rec.gainRight;
void AudioBussMixer::setBussLevels(int bussId,
                    float dB, float pan)
{
        ...
    float volume = AudioLevel::dB_to_multiplier(dB);
    rec.gainLeft = volume *
        ((pan > 0.0) ? (1.0 - (pan / 100.0)) : 1.0);
    rec.gainRight = volume *
        ((pan < 0.0) ? ((pan + 100.0) / 100.0) : 1.0);
        ...
}
```

Mixing Process from PlayableAudioFile.cpp

```
for (size_t i = 0; i < n; ++i) {
        sample_t v = cached[0][scanFrame + i]
+ cached[1][scanFrame + i];
        destination[0][i + offset] += v;
}
```

A.4 Audio Mix Analyzer Mixing Algorithms

A.4.1 Simple Addition

```
for(int track = 0; track<size; track++)
    {
        left += mixA[track]->getSample(0,index)
            *Decibels::decibelsToGain(gainA[track]);

    }
```

A.4.2 Divide by Number of Tracks

```
for(int track = 0; track<size; track++)
    {
        left +=mixA[track]->getSample(0,index)
            *Decibels::decibelsToGain(gainA[track]);
    }
    left = left/size;
```

A.4.3 Viktor T. Toth

```
//set toth value - depends on number of tracks
float toth = 1;
if(size==2) toth = 0;
if(size>2) toth /= -size;
...
less = true;
for(int track = 0; track<size; track++)
{
    if(mixA[track]->getSample(0,index) >= toth)
        less = false;
}
if(less)//2ab
{
    for(int z = 0; z<size; z++)
        left *=mixA[z]->getSample(0,index)
                *Decibels::decibelsToGain(gainA[z]);
    left *=size;
}
else//2(a+b)-2ab-1
{
    float sum = 0;
    for(int z = 0; z<size; z++){
        sum += mixA[z]->getSample(0,index)
            *Decibels::decibelsToGain(gainA[z]);
```

```
        left *=mixA[z]->getSample(0,index)
                *Decibels::decibelsToGain(gainA[z]);
    }
    left = size*sum - size*left - 1;
}
```

A.4.4 Linear Dynamic Range Compression

```
for(int track = 0; track<size; track++)
    {
        left +=mixA[track]->getSample(0,index)
            *Decibels::decibelsToGain(gainA[track]);

    }
    abs = left;
    if(left <0)
        abs = -left;

    if(abs>t){
        left = (left/abs)*(t + (1-t)/(2-t)*(abs - t));
    }
```

A.4.5 Logarithmic Dynamic Range Compression

```
float abs;        //absolute value
float f = 7.48; //f(alpha) value for t = 0.6
...
for(int track = 0; track<size; track++)
    {
        left +=mixA[track]->getSample(0,index)
            *Decibels::decibelsToGain(gainA[track]);
    }
    abs = left;
    if(left <0)
        abs = -left;
```

```
        //if absolute value is bigger than the threshold
    if(abs>t){
        left = (left/abs)*(t + (1-t)
                *(log(1+f*(abs-t)/(2-t))/log(1+f)));
    }
```

A.4.6 Tan h

```
for(int track = 0; track<size; track++)
    {
        left +=tanh(mixA[track]->getSample(0,index))
            *Decibels::decibelsToGain(gainA[track]);
    }
```

A.4.7 Analogue Emulator

```
for(int track = 0; track<size; track++)
    {
        left += mixA[track]->getSample(0,index)
            *Decibels::decibelsToGain(gainA[track])
            *Decibels::decibelsToGain(-3.00);

    }
```

A.4.8 Random Analogue

```
for(int track = 0; track<size; track++)
    {
        sign = rand.nextInt(2);
        int val = rand.nextInt(10);
        value = val*0.0001;
        if(sign==0)
            value = value*-1;
        left += (mixA[track]->getSample(0,index)+value)
            *Decibels::decibelsToGain(gainA[track]);
    }
```

A.5 RMS Level Adjustment

```
destA = mixingAlgorithms(0,selected);
float r = destA.getRMSLevel(0,0,length);
while(r < 0.177 || r > 0.179){
        if(r < 0.177)
                destA.applyGain(0,0,length,1.01);
        if(r > 0.179)
                destA.applyGain(0,0,length,0.99);
        r = destA.getRMSLevel(0,0,length);
}
master_volumeA->setValue(r);
```

A.6 Timing Tests

```
Time *time = new Time();
int currentTime = time->currentTimeMillis();
...
        for(int track = 0; track<len; track++){
                switch(algorithm){
                        ...
                }
                destination.setSample(0,track,left);
        }
        int timeTaken = time->currentTimeMillis() - currentTime;
        DBG("Time to mix using " + chooseMixA->getItemText(selected-1)
             + " algorithm: " + String(timeTaken)+ " ms");
```

Appendix B

Object-Oriented Design

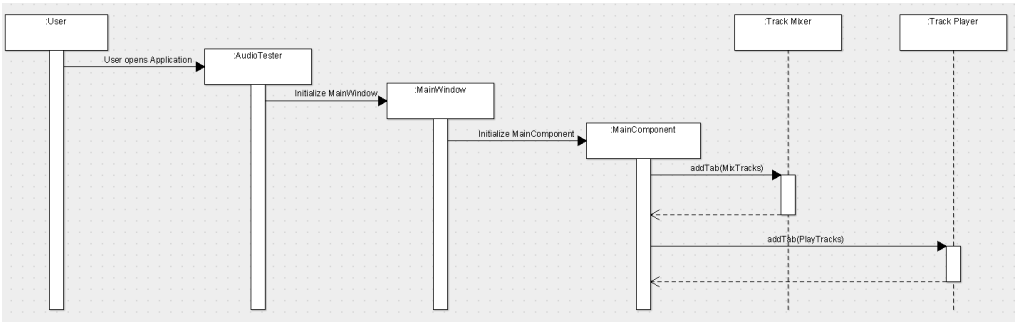


Figure B.1: Sequence Diagram for Startup

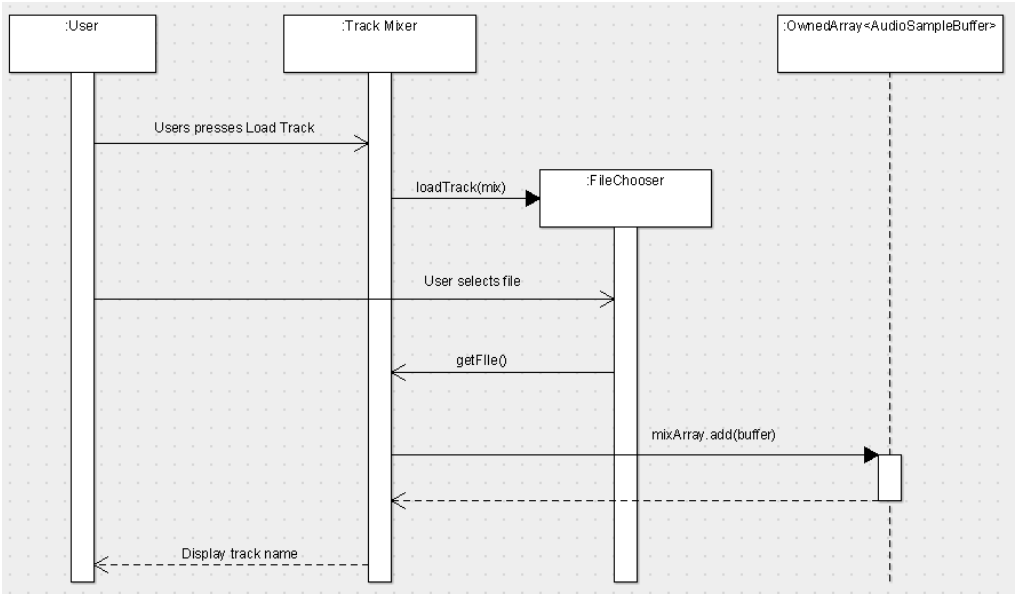


Figure B.2: Sequence Diagram for Load Track to Mix

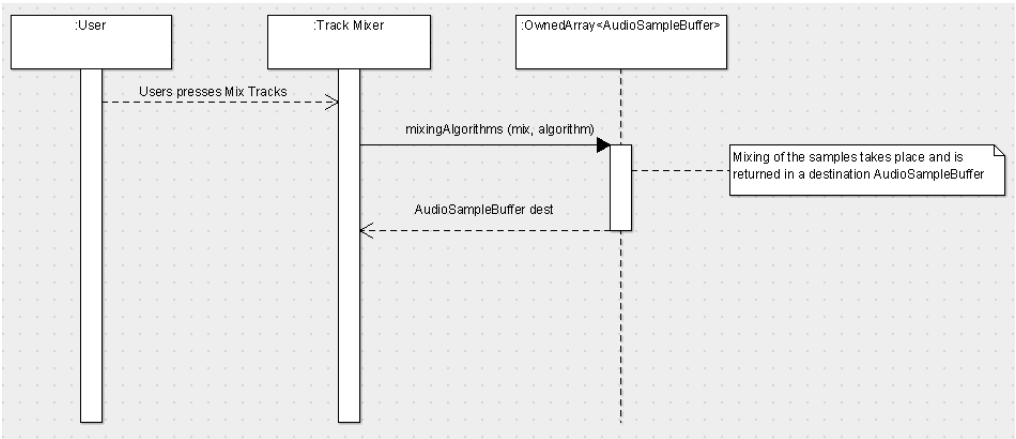


Figure B.3: Sequence Diagram for Create Mix

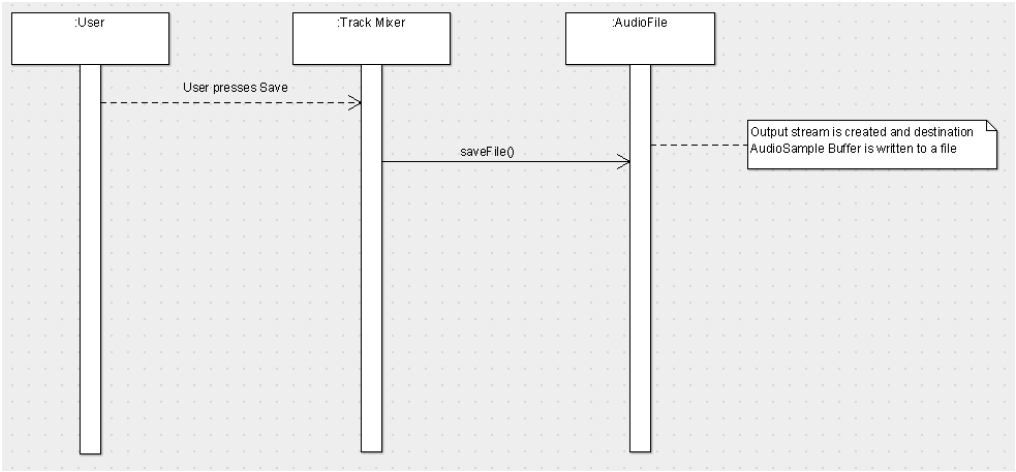


Figure B.4: Sequence Diagram for Save Mix

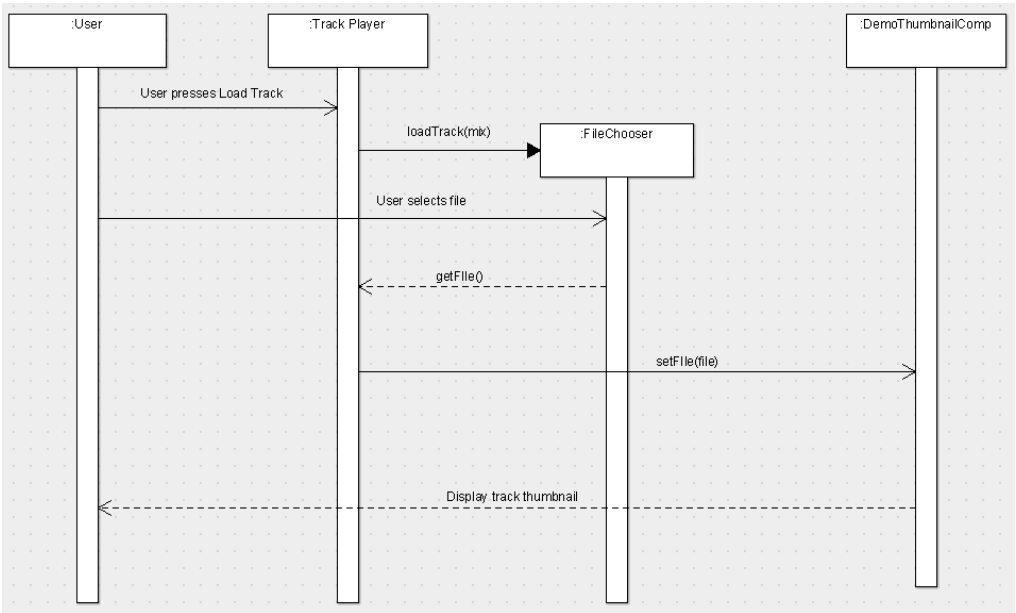


Figure B.5: Sequence Diagram for Load Track to Play

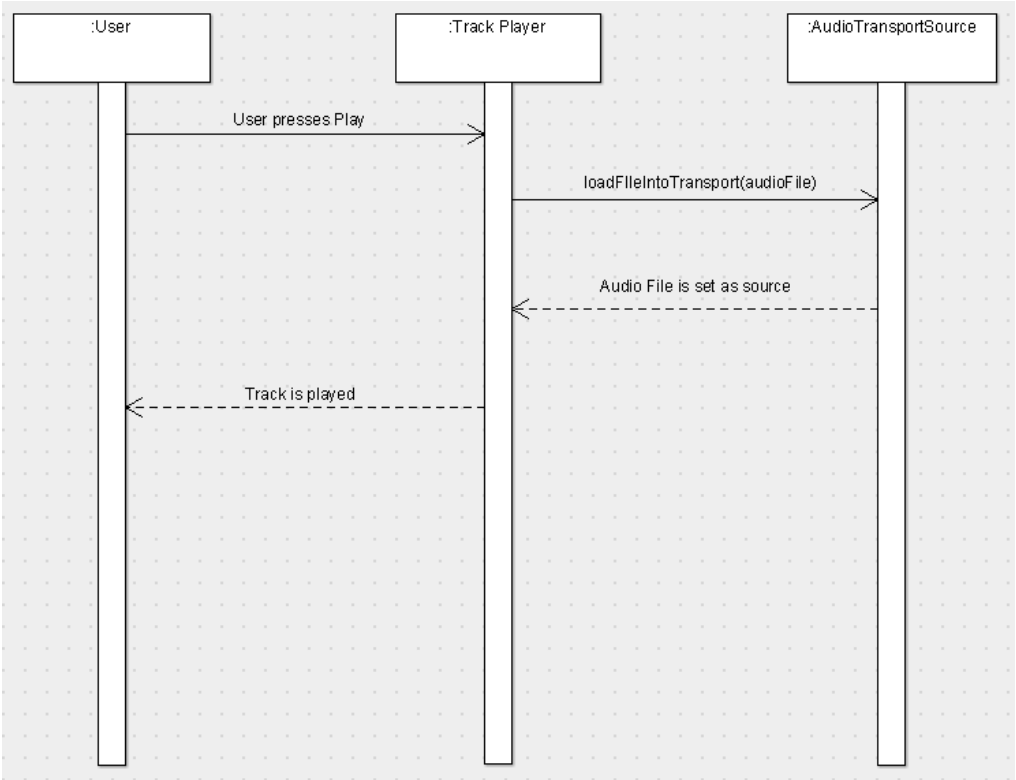


Figure B.6: Sequence Diagram for Play Tracks

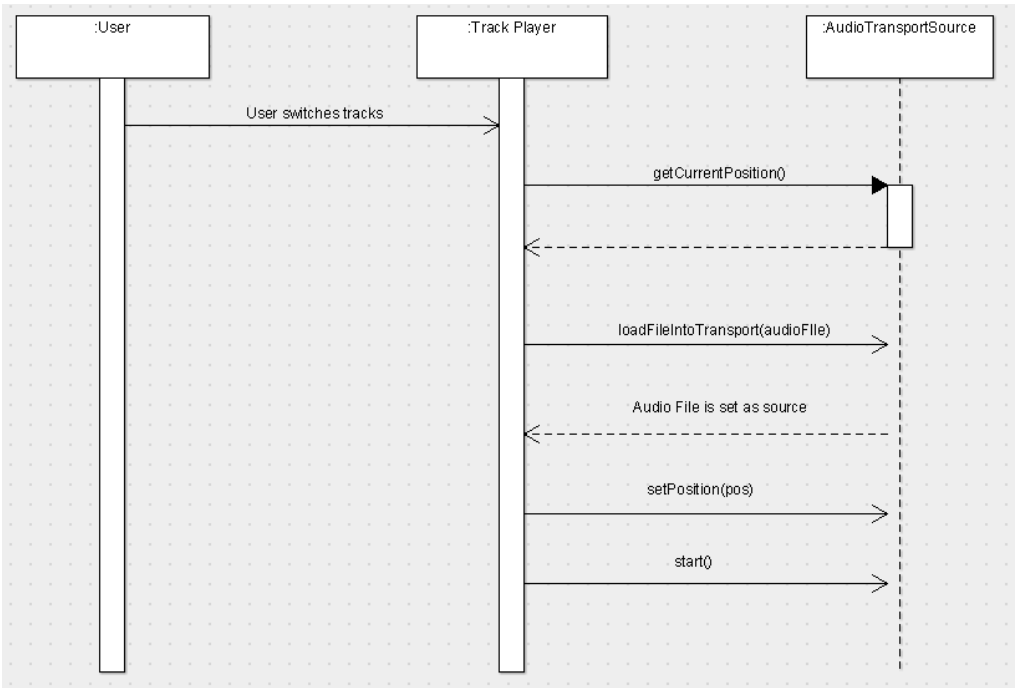


Figure B.7: Sequence Diagram for Switching between Tracks

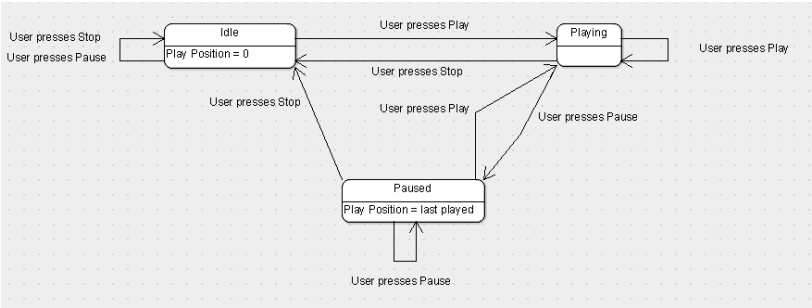


Figure B.8: State Diagram for Playback

Appendix C

Listening Tests

C.1 Test Configuration

Table C.1: Audio Test Samples

Test	Song	Number of Tracks	Sample A	Sample B
1	Black Out Betty	8	Analogue	Digital
2	Black Out Betty	8	Analogue	Simple Addition
3	Black Out Betty	8	Logarithmic	Analogue
4	Black Out Betty	5	Digital	Analogue
5	Black Out Betty	8	Analogue	Analogue
6	Black Out Betty	3	Analogue	Digital
7	Catching Up	8	Simple Addition	Analogue
8	Catching Up	8	Analogue	Digital
9	Catching Up	8	Analogue	Analogue
10	Catching Up	3	Digital	Analogue
11	Catching Up	8	Analogue	Tanh
12	Catching Up	5	Analogue	Digital
13	Rockshow	5	Digital	Analogue
14	Rockshow	8	Analogue	Analogue
15	Rockshow	8	Analogue	Digital
16	Rockshow	8	Analogue	Simple Addition
17	Rockshow	8	Analogue	Divide by Tracks
18	Rockshow	2	Analogue	Digital
19	Rockshow	8	Simple Addition	Digital

C.2 Response sheet

Age: _____

Previous music (listening) experience:

_____

_____

For each test please write down:

1. If you can hear a difference between the samples?
2. How you would describe this difference?
3. The clarity of Sample A rated from 1 (clear) to 5 (muffled).
4. The clarity of Sample B rated from 1 (clear) to 5 (muffled).

Test	Difference	Description of Difference	A clarity	B clarity
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				

C.3 Response Data

C.3.1 Summary Tables

Table C.2: Responses per Test ordered by Average Absolute Clarity Difference

					Difference		
Test	**Song**	**Tracks**	**Sample A**	**Sample B**	**Yes**	**No**	**Clarity**
1	BOB	8	Analogue	Digital	13	0	2.6923
17	RS	8	Analogue	Divide by Tracks	13	0	2.1923
16	RS	8	Analogue	Simple Addition	13	0	1.6154
8	CU	8	Analogue	Digital	13	0	1.5385
15	RS	8	Analogue	Digital	13	0	1.5385
6	BOB	3	Analogue	Digital	13	0	1.2692
4	BOB	5	Digital	Analogue	12	1	1.0769
10	CU	3	Digital	Analogue	11	2	0.9231
12	CU	5	Analogue	Digital	12	1	0.7308
19	RS	8	Simple Addition	Digital	5	7	0.3636
13	RS	5	Digital	Analogue	5	8	0.3333
11	CU	8	Analogue	Tanh	5	8	0.2308
5	BOB	8	Analogue	Analogue	5	8	0.1667
3	BOB	8	Logarithmic	Analogue	8	5	0.1538
7	CU	8	Simple Addition	Analogue	4	9	0.1538
14	RS	8	Analogue	Analogue	1	12	0.0909
9	CU	8	Analogue	Analogue	4	9	0.0385
18	RS	2	Analogue	Digital	6	7	0.0385
2	BOB	8	Analogue	Simple Addition	6	7	0

C.3.2 Raw Data

Table C.3: Question 1 Response by Age

	Age												
Test	**18**	**20**	**21**	**22**	**23**	**23**	**23**	**23**	**24**	**24**	**28**	**51**	**55**
1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	No	Yes	Yes	No	No	Yes	Yes	Yes	No	No	No	Yes	No
3	Yes	Yes	Yes	Yes	No	Yes	No	No	No	Yes	Yes	Yes	No
4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
5	No	Yes	No	Yes	No	No	No	Yes	Yes	No	No	Yes	No
6	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
7	No	Yes	Yes	Yes	No	No	No	No	No	No	No	Yes	No
8	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
9	No	Yes	No	Yes	No	No	No	Yes	No	No	No	No	Yes
10	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No
11	No	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	No
12	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
13	No	Yes	Yes	No	No	Yes	No	Yes	No	No	No	Yes	No
14	No	No	No	No	No	Yes	No	No	No	No	No	No	No
15	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
16	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
17	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
18	Yes	Yes	Yes	No	No	No	No	No	Yes	No	No	Yes	Yes
19	Yes	No	Yes	Yes	No	No	No	/	Yes	No	No	Yes	No

Table C.4: Question 3 and 4 Response by Age

Age	**18**		**20**		**21**		**22**		**23**		**23**		**23**		**23**		**24**		**24**		**28**		**51**		**55**	
Test	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
1	1	3	1	5	1	5	1	5	1	4	1	3	1	2	1	5	1	4	1	4	3	4	1	4	2	3
2	1	1	1	1	3	2	2	2	1	1	1	2	1	2	2	1	2	2	3	3	2	2	1	1	2	2
3	1	1	1	1	3	1	2	3	1	1	1	1	2	2	1	1	2	2	3	3	3	2	1	1	2	2
4	1	3	1	2	3	2	2	4	1	3	2	1	2	3	1	5	3	3	2	4	2	3	1	2	1.5	1.5
5	1	1	2	1	2	2	2	2	2	2	1	1	2	2	2	1	2	2	3	3	2	2	1	1	/	/
6	3	2	3	2	3	2	4	2	1	1	3	1	3	2	5	1	2	1	2.5	2	4	2	3	2	1	1
7	2	2	1	1	3	2	3	2	2	2	2	2	2	2	1	1	3	3	3	3	4	4	2	2	1	1
8	2	3	1	3	2	4	2	3	1	2	1	4	1	3	1	5	2	3	2.5	3	2	3	1	2	1	1.5
9	1	1	2	3	1	1	2	2	2	2	1	1	2	2	2	1	2	2	2.5	2.5	2	2	1	1	1.5	1
10	1	2	1	2	1	1	2	2	1	2	1	3	1	2	1	5	1	2	2.5	2.5	2	3	1	1	1	1
11	1	1	1	1	1	2	2	2	2	2	1	2	2	2	1	2	2	2	3.5	3.5	3	3	1	1	1	1
12	2	1	3	2	4	4	3	2	2	1	5	3	1	2	5	1	2	2	2.5	2.5	4	4	1	1	1.5	1
13	2	2	1	2	4	5	2	2	3	3	3	4	2	2	1	2	2	2	3.5	3.5	4	4	1	1	/	/
14	1	1	2	2	3	3	2	2	2	2	2	3	/	/	1	1	1	1	2	2	4	4	2	2	/	/
15	2	2	4	1	4	3	3	2	3	1	4	1	3	2	5	1	2	1	2	3	5	3	2	1	3	1
16	2	1	4	1	4	2	3	2	3	1	4	2	3	2	5	1	3	2	1.5	3.5	5	3	3	1	3	1
17	2	1	4	1	4	2	3	2	3	1	5	1	3	1	5	1	3	1	3.5	2	5	3	3	1	3.5	1.5
18	2	1	5	5	3	4	2	2	1	1	3	3	4	4	1	1	2	2	4	4	4	4	1	1	3	3.5
19	1	1	/	/	1	2	2	2	1	1	4	4	2	2	/	/	1	3	2.5	2.5	3	3	1	2	2.5	2.5

Table C.5: Absolute Difference in Clarity per Test

Difference in clairty														
	18	**20**	**21**	**22**	**23**	**23**	**23**	**23**	**24**	**24**	**28**	**51**	**55**	**Average**
1	2	4	4	1	3	2	1	4	3	3	1	3	1	2.462
2	0	0	1	2	0	1	1	1	0	0	0	0	0	0.462
3	0	0	2	2	0	0	0	0	0	0	1	0	0	0.385
4	2	1	1	2	2	1	1	4	0	2	1	1	0	1.385
5	0	1	0	2	0	0	0	1	0	0	0	0	/	0.333
6	1	1	1	4	0	2	1	4	1	0.5	2	1	0	1.423
7	0	0	1	3	0	0	0	0	0	0	0	0	0	0.308
8	1	2	2	2	1	3	2	4	1	0.5	1	1	0.5	1.615
9	0	1	0	2	0	0	0	1	0	0	0	0	0.5	0.346
10	1	1	0	2	1	2	1	4	1	0	1	0	0	1.077
11	0	0	1	2	0	1	0	1	0	0	0	0	0	0.385
12	1	1	0	3	1	2	1	4	0	0	0	0	0.5	1.038
13	0	1	1	2	0	1	0	1	0	0	0	0	/	0.500
14	0	0	0	2	0	1	/	0	0	0	0	0	/	0.273
15	0	3	1	3	2	3	1	4	1	1	2	1	2	1.846
16	1	3	2	3	2	2	1	4	1	2	2	2	2	2.077
17	1	3	2	3	2	4	2	4	2	1.5	2	2	2	2.346
18	1	0	1	2	0	0	0	0	0	0	0	0	0.5	0.346
19	0	/	1	2	0	0	0	/	2	0	0	1	0	0.545