Knobs and Nodes: A Study of UI Design in Audio Plugins

An exegesis presented in partial fulfilment of the requirements for the degree of

Master's

in

Creative Enterprise

At Massey University, Wellington, New Zealand

Jonathan Peter McGregor

2019

Abstract

Knobs and Nodes explores how alternate user interfaces influence the use of audio plugins in music production. This idea was investigated through the development and user testing of audio plugins with node-based user interfaces. The *Nodal Plugin Duo*, comprises two unique audio plugins which exhibit delay and reverb digital signal processing. Once developed, these plugins were tested by music producers, sound designers, and composers in their native creative environments to provide insight on the usability and interactivity of the *Nodal Plugin Duo*.

Acknowledgements

Firstly, I would like to thank my supervisors Bridget and Jon, for not only your guidance throughout this degree, but for being genuinely awesome people. I would also like to thank Chris, for your wealth of knowledge, great yarns, and putting up with the time difference! Dave, for whipping this exegesis into shape, without your time and effort, I would still be writing my introduction. Thank you Catherine, for helping me formulate my ideas, and the endless supply of cat memes - they were an absolute treat.

Thank you Kaysha and Callum; The Jalas Foundation. We started this together, and now look at us go. May we continue our patronage to Jalas for years to come.

Thanks Mum and Dad, for supporting me not only through this degree, but every moment of my life leading up to it, and every moment that follows it.

Finally, I would like to thank Sophia, you have kept me sane throughout this whole process. I love you, and I can't wait to see what our next adventure is.

Contents

List of Figures

1. Introduction

1.1 Motivation

As a music producer my interests lie in the tools which I use to create and produce new music. This interest led me towards software development during my undergraduate degree. As I explored this field in the context of audio application development I began to understand the processes involved in developing the tools I and other music producers use for creative expression. Once I began developing audio software, I became increasingly aware of both the challenges and importance of user interface design within these applications. When considering the design of my own user interfaces, I'd look at the software used for my personal music-making for inspiration. I'd ask questions such as "why are these interfaces designed this way?", and "how does this design help me as a composer?".

Guerrero [1] identifies that traditional user interface paradigms were not developed for the purpose of musical composition, but rather for the emulation of hardware devices. This is because the original target audience of the software was professional sound engineers during the time which digital recording was becoming a mainstream method of music production [1]. Due to the target audience's unfamiliarity with this new medium, software developers would model their products off of analogue devices, thus affording familiarity toward the professionals transitioning into the software. Products designed in this way are often called skeuomorphs [2].

1.2 Key Terminology

Skeuomorphism refers to any digital user interface which emulates real-world objects in appearance and/or interactivity [3]. Within the realm of music software, skeuomorphism primarily manifests in the form of knobs, sliders, and buttons, which are the key user interaction devices on music hardware.

The user interface model which I propose in my study is a *node-based* interface where audio parameter values are determined by the relationships between circular interface elements called nodes. Similar interfaces have been developed prior to this study, i.e. Timothy Baraclough's *Pyxis Minor* [4] and The Cargo Cult's *Slapper* [5] (Figure 1.2.1), as well as many parametric EQ plugins on the market.

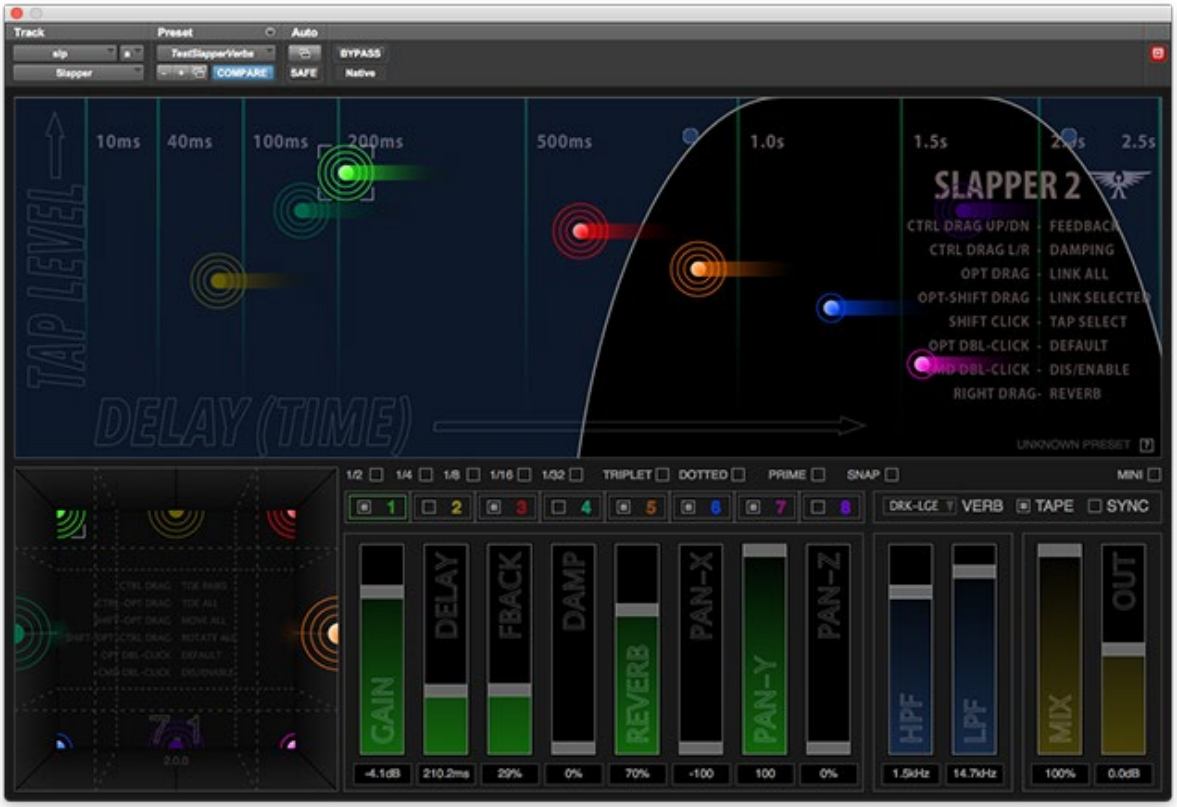

Figure 1.2.1 – Cargo Cult's Slapper – Node based multi-tap delay

1.3 Research Goals

The following research goals were developed based on the motivation outlined in section 1.1 of the *Introduction*:

1. Investigate the paradigms of audio software user interface design.
2. Investigate the works of those who have questioned these paradigms.
3. Develop a suite of audio plugins with alternative user interface design elements to explore intuitive user interaction with audio software.

In addressing these research goals, this exegesis hopes to provide the field with a methodology for designing new software user interfaces which prioritise user-centred design principles to encourage creative expression of audio.

1.4 Exegesis Structure

The following chapters of this exegesis will document the process of developing the *Nodal Plugin Duo* to achieve the above research goals. In Chapter 2 a review of past works in the field of audio software user interface design both from the academic and commercial spheres will be provided. This chapter will discuss the role of skeuomorphism in modern audio software user interface paradigms as well as alternate user interface design models and metaphors.

Chapter 3 discusses the methodologies behind the development of the *Nodal Plugin Duo* in terms of the software development life cycle (SDLC) as well as the programming language and application programming interface (API) selected for development. Although a range of methodologies were considered during initial development, the SDLC selected was an Agile lifecycle, and the chosen API was JUCE in C++.

Chapter 4 contains the implementation of the methodology outlined in Chapter 3 to develop both plugins in the *Nodal Plugin Duo*. After explaining the development process of the plugins *Lapse* and *Space*, the outcome of a user test is then discussed which informs the final iterations of both plugins.

In Chapter 5, the outcome of the *Nodal Plugin Duo* is reflected on in relation to the research goals of the project and the potential for future works is outlined.

2. Contextualisation

2.1 Introduction

This chapter reviews literature both from the commercial and academic spheres to provide a context and direction for the Nodal plugins.

Throughout the course of this review a variety of literature will be referred to in order to address the following questions:

1. What is the value of skeuomorphic user interface design in audio production software?
2. How has the functionality of audio plugins been represented by alternative user interface designs?

These questions address the first two research goals and the resultant findings from these goals were applied to the design and development of the *Nodal Plugin Duo*.

2.2 Paradigms of Audio Software User Interface Design

In audio software development, user interfaces are considered equally as important as the audio processing behind it [6]. Therefore, when designing new audio software, it is important to consider current user interface paradigms in relation to the target audience that will be interacting with these interfaces.

Duignan et al [7] in their comparative study of Reason and Ableton Live provide an analysis of the user interface paradigms found in modern DAW's. They discuss the various visual metaphors present in both software applications and observe both the affordances and limitations of such design choices for the user. They establish the "Rack" metaphor, as found within Propellerhead's Reason (Figure 2.2.1), along with the "Channel Strip/Mixer" metaphor found in Ableton Live (Figure 2.2.2). They then observe that the commonality across both of

these metaphors were the fact that they were “dependent on leveraging peoples real world knowledge” [7]. This observation underpins the roots of user interface design paradigms and traces back to the early development of audio software.

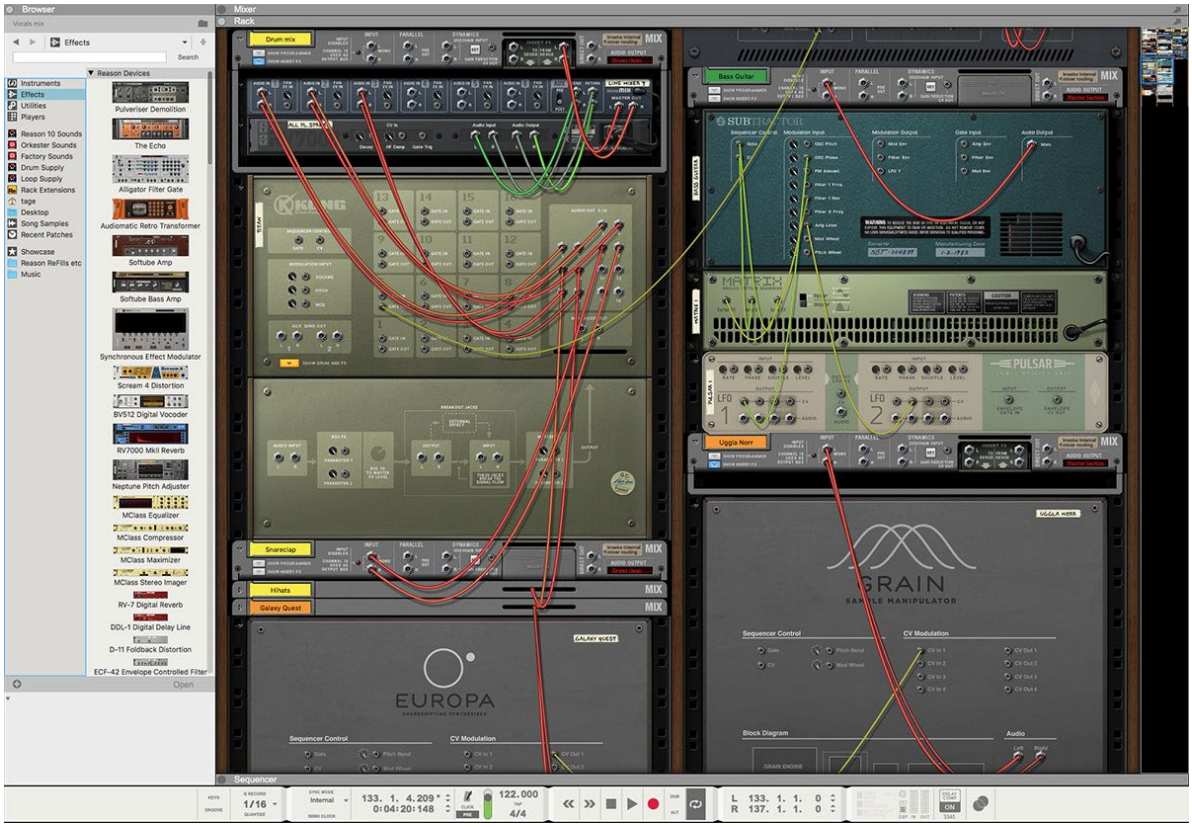

Figure 2.2.1 – Reason by Propellerhead, “Rack” metaphor emulating modular hardware

Upon the initial implementation of skeuomorphic music software in the late 20th century, software developers wanted to establish some form of familiarity between the DAW and sound engineer by visually emulating real-world, analogue equipment [8]. This method of design can still be seen today in many popular commercial works such as the *SoundToys Suite* [9], and Native Instrument’s *Massive* [10]. Whilst they are both unique pieces of software from an audio processing perspective, they lean heavily on skeuomorphic design for their user interfaces.

Figure 2.2.2 – Ableton Live 10 "Channel Strip/Mixer" metaphor emulating analogue recording desks

Research into skeuomorphic design in audio software [1], [2], [6], [7], [8], [12] indicates that this design approach provides two different types of value to the user:

1. Value of Understanding
2. Value of Aesthetics

The value of understanding is the level of functional familiarity the user has when interacting with a skeuomorphic component in an application. Value of understanding allows the user to utilise their knowledge of physical objects to interact with the interface [2]. However, referring back to the purpose of skeuomorphic design, this value is only effective if users have prior knowledge of what is being emulated.

Over the last three decades the proportion of users coming to audio software with prior experience with music hardware has shrunk [7]. This is where the justification of skeuomorphism in new audio software interfaces becomes problematic, as many of the target users who interact with these interfaces haven't encountered the analogue hardware

of which it is modelled [1]. This leads to the second value which skeuomorphism provides for the user, value of aesthetics.

The value of aesthetics is the visual value the user holds for the interface, and is informed by previous attachments to the analogue equipment from which the interface is modelled. This appeal is further incentive for software companies to model their user interfaces off analogue equipment as the user will connect the social appeal of using this equipment to the software used to develop their own creative works. This idea is displayed in Waves' *Abbey Road Collection* (Figure 2.2.3), where users are told the following:

> With the Abbey Road Collection, the legendary studio's signal chain is at your disposal – from the rooms in which magic happened, through the iconic vintage gear and techniques that ushered in the age of modern audio engineering [11].

In Dominik Schrey's *Analogue Nostalgia and the Aesthetics of Digital Remediation* [12], Schrey refers to this concept as the "fetishising of analogue media". Whilst he discusses this trend across multiple facets of modern western culture, Alan Williams [13] discusses analogue fetishisation in the context of contemporary music technology:

> Software emulations are not inherently nostalgic, though much of the marketing surrounding them capitalizes on the desire to harness the past [13, p. 9].

He then outlines the negative repercussions of this concept:

> The real crime is the commoditized dream that has been instilled in post-millennial audio enthusiasts. . .the growth market in audio technology is geared (pun intended) towards the budding recordist who has never known a world without ProTools and iTunes, and maximizing profit is dependent upon selling a nostalgia for someone else's 'good old days' [13, p. 9].

Subsequently, skeuomorphic design in audio software might be better understood as profit-centred rather than user-centred design.

Figure 2.2.3 – J37 from Waves Abbey Road Collection

Myllys [14], in *User Interface Paradigms in Digital Audio Workstations*, discusses the lack of innovation that has been seen within music software over the past decade in comparison to other aspects of modern technology such as the smartphone and laptop. They identify that whilst these pieces of technology used to rely heavily on skeuomorphic interface components, they have since moved away from such design principles as their audience has grown with the product. They then suggest that music software should also be attempting to move away from user interface metaphors due to the interactive limitations of hardware emulation. The impacts of these limitations on user experience closely relate to Csikszentmihalyi's model of flow.

Csikszentmihalyi's model of flow is applied to interface design when considering user interactions. One such of these applications can be found in Push, Turn, Move [6]. In this

publication Bjorn states, “Flow is a term artists and musicians instinctively understand; it refers to a state where they are working near to their skill level. In other words, the interface is offering no interruptions to the process, and is supporting and/or challenging the user in a positive way.” Skeuomorphic designs in software interrupt the creative process as it is the tactile nature of the hardware which afforded flow state in these original designs [6]. Therefore, non-skeuomorphic interfaces designed user-centrically will facilitate more creative interactions in the realm of audio software.

Having considered the value of skeuomorphic user interface design in music software and its role in past works, it is important to consider those who have questioned these conventions and displayed alternate models of user interface design in their own works.

2.3 Alternative User Interface Design in Audio Software

The most prominent piece of audio software which implements non-skeuomorphic user interface design is the modern parametric equaliser (Figure 2.3.1). This user interface has become the norm for many software equalisers such as the EQ in Logic Pro X [15], Fab Filter’s Pro-Q [16], and Equilibrium [17].

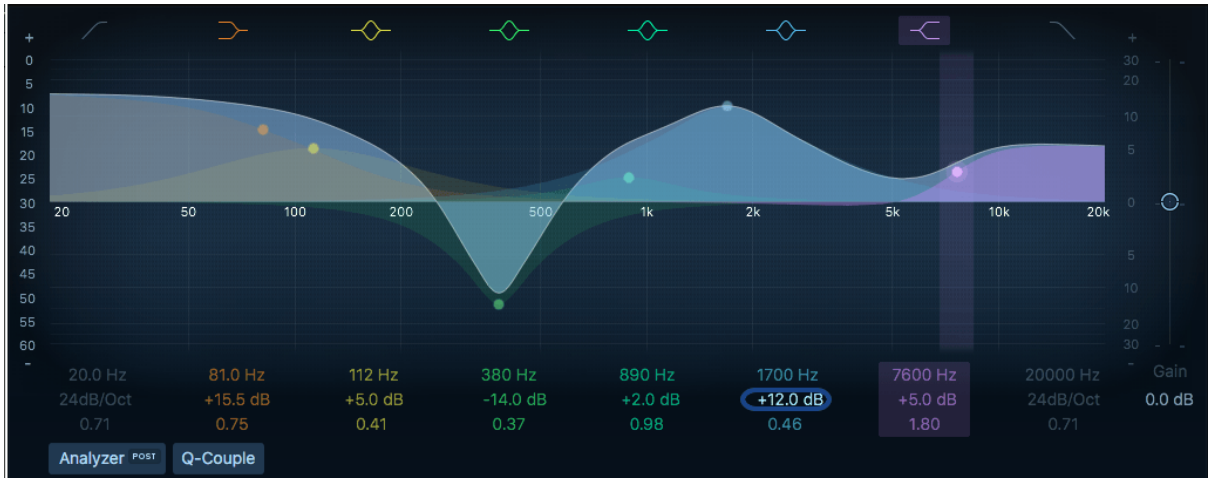

Figure 2.3.1 – Parametric EQ in Logic Pro X

In this user interface, instead of interacting with digital knobs and sliders to change parameters, the user can change the X (filter frequency) and Y (gain in dB) positions of nodes

around the interface to determine the frequency response of the filter. This node-based interface is seen in various other audio plugins such as Obscurium [18], and Slapper [5]. This interface is clearly more appropriate for the medium than the emulation of hardware parametric equalisers in software as the visual information for the user effectively communicates the audio output of the software. However, where this design falls short from skeuomorphs is in its use of visual space on the user's screen. Although appropriate use of screen space is an important factor when designing audio software interfaces, it is arguably more important that the interfaces that are designed prioritise interactions created specifically for the medium of audio software.

The modern parametric equaliser is an example of large-scale companies developing a non-skeuomorphic interface. However, the field of independent plug-in developers has seen much more in-depth exploration of this concept. An analysis of visual design trends in plugins such as these was conducted (see appendix for full study). Through this study the following design trends were observed:

1. Using XY pads to display and change parameters make it easier to create variety in an effect as it links multiple parameters.
2. Some interfaces would have an alternative UI "centre-piece", such as an XY pad, combined with skeuomorphic knobs/sliders. This resulted in an interface that was somewhat familiar to hardware users, yet inspired use of alternative UI elements.
3. Alternate metaphors to control parameters are useful for visualising more complicated relationships e.g. frequency & gain on an EQ.
4. Tabbed windows allow easy access to higher functionality without cluttering the workspace.
5. Colour matching various parameters with other visual components help the user to make mental connections between GUI objects.
6. Dark backgrounds with vivid colours on parameters make controls easy to understand/see for the user.

7. Most GUI's observed have less than 5 different colours in their scheme. Those with more will tend to have a dominant colour with small amounts of other colours to prevent the GUI from seeming too crowded.
8. Most text items are either all upper-case or all lower-case. This makes parameters easier to read.

Teenage Engineering's *OP-1* [19] is a strong manifestation of these trends. The *OP-1* (Figure 2.3.2) is a portable synthesizer, sampler, and controller with a digital display.

Figure 2.3.2 – Teenage Engineering's OP-1 utilising alternative metaphors in firmware

As the OP-1's software is paired with a hardware device, the display does not feature analogue music hardware emulations, rather the user interface utilises vector-based visualisations of alternative metaphors. This can be seen in the *CWO* filter module (Figure 2.3.3) which uses a line-art depiction of a cow to display the functionality of a modulating filter. The relationships between the hardware and UI are then clearly displayed through colour coding the hardware knobs with the software. These user interface designs demonstrate a user-centred design approach which encourages exploration and creativity through interaction with music technology.

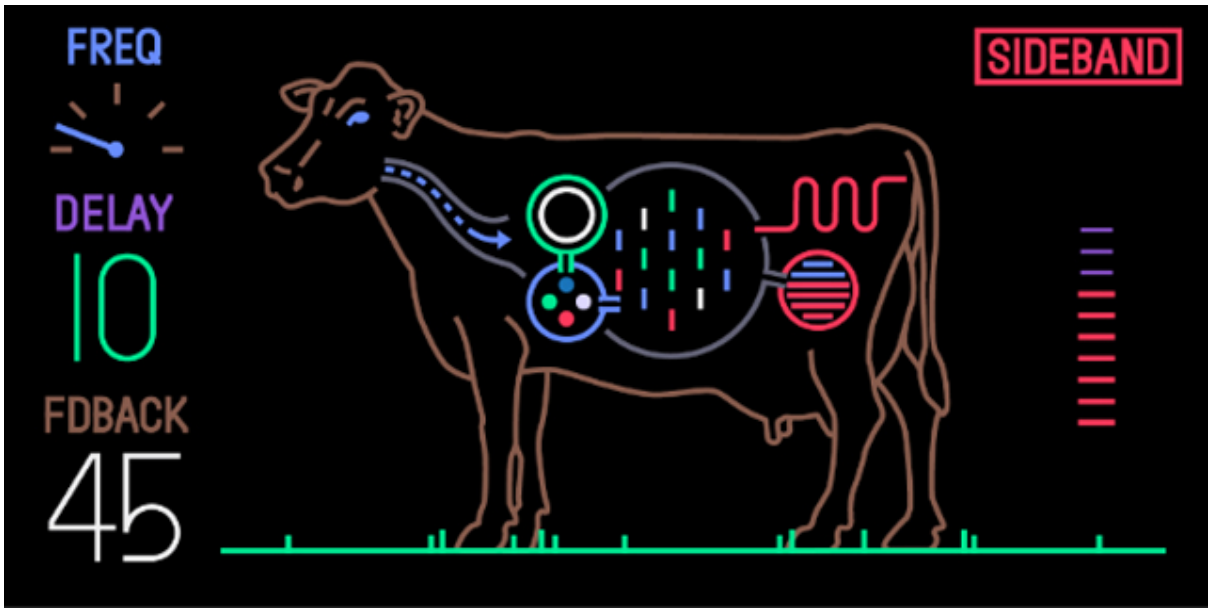

Figure 2.3.3 – Teenage Engineering's CWO – Alternate metaphor for representing a modulating filter

2.4 Conclusion

By reflecting upon works within academic and commercial fields, skeuomorphic user interface paradigms were revealed to be the most prevalent visual designs in audio software. However, through studying examples of alternate user interface designs in commercial products, trends were observed in the various approaches to non-skeuomorphic user interfaces. These design trends will be utilised for the development of the creative work in the subsequent chapters.

3. Methodology

3.1 Introduction

The *Nodal Plugin Duo* is a suite of two audio plugins utilising the node-based alternative user interface model. This node-based model was designed through observing and implementing trends in non-skeuomorphic audio plugin designs discussed in the previous chapter. The duo was developed in C++ using the JUCE application programming interface (API) and an Agile model was utilised for the lifecycle of the project.

3.2 Software Development Lifecycle

The three most common industry models for software development are Agile, V-Model, and Waterfall [20]. Agile is an iterative methodology which strengths lie in its flexibility but falls short in providing a timeline for clear deliverables. Waterfall, conversely, is a linear methodology whereby the whole development process is planned from start to finish and each stage is carried out sequentially. The rigidity of Waterfall provides a clear path from the start to end of development yet provides very little room to deviate from the original list of requirements. V-Model is an extended methodology of the waterfall model.

This project has used an Agile software development model to produce a suite of non-skeuomorphic audio plugins. Agile methodologies have been chosen because of the flexibility afforded by the iterative model and the alignment of this with the undefined project outcomes in the early stages of development.

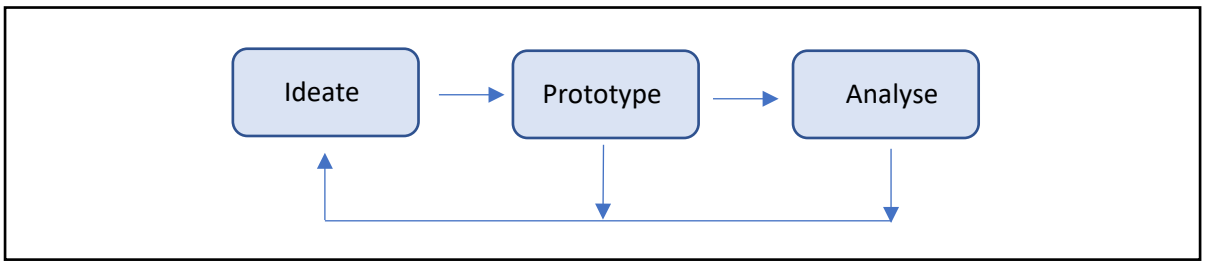

Figure 3.1.1 – Flow chart of iterative design cycle

Each plugin was developed over a series of iterations or “sprints” which can be broken down into the stages shown on Figure 3.1.1. The initial sprint for each plugin begins with a specific audio effect in mind. Initial user interfaces were then drafted for this audio effect. Once a user interface was decided upon, it was then implemented in C++ with the digital signal processing (DSP). Goals for the subsequent sprints are then formed based on the reflections of this prototype, and the next sprint begins.

The progression of the project was documented through the maintenance of repositories on GitHub. This allowed thought processes to be displayed alongside updates in the code itself. These thought processes formed the bulk of my reflections and therefore informed the features to be implemented in the subsequent sprints.

The two alternatives were to employ a waterfall development cycle, which follows a linear timeline, or a V-Model, which is an extended version of the waterfall model. The waterfall method was attempted initially but was unsuccessful as there were many unknown aspects of the project at the start of development, and therefore a linear timeline quickly became inappropriate to follow in the long term, ruling out both Waterfall and the V-Model.

3.3 Language and Application Programming Interface (API)

After selecting a lifecycle for the project, the next requirement regarded the programming language and API for development. In terms of language, C++ is the industry standard for digital signal processing (DSP) and was therefore chosen for this project. Secondly it was important to use an appropriate API that would not only simplify the plugin development process but would also allow for flexible graphical user interface (GUI) prototyping. JUCE was the most appropriate API as it fulfilled the above requirements and supported Windows, Mac, and Linux systems for VST2, VST3, AU, and AAX formats.

4. Implementation

4.1 Introduction

The *Nodal Plugin Duo* consists of two audio plugins with node-based user interfaces; *Lapse*, and *Space*. *Lapse*, and *Space* exhibit delay, and reverb audio effects. These time-based effects were selected as many of the alternate plugin designs researched in the case study of Chapter 2 effectively visualised time-based effects using alternate user interface elements. This indicated that these audio processing techniques were a valid starting point for alternative visualisations of DSP.

The initial goal of development was to create a suite of audio plugins which utilised alternate user interface design elements to re-imagine interaction with various audio effects. These designs were developed utilising the conclusions of the case study in Chapter 2 regarding past works of alternative user interface design. The development of these plugins will be discussed in this chapter from the initial development of a basic delay plugin, to the first node-based design for *Lapse*, all the way through to the final build of *Space*. The following chapter will be laid out chronologically, following the methodology presented in the previous chapter, with consideration of the visual design principles outlined in Chapter 2.

4.2 Lapse

Lapse was the first plugin developed for the *Nodal Plugin Duo. Lapse* automates delay and pan parameters using a *node-based* user interface synchronised to the bpm of its host.

The development of *Lapse* began with the goal of creating a basic delay plugin. The first sprint began by researching delay implementations. Through this research Daniel Walz's *ffTapeDelay* [22], a basic plugin with delay time and feedback parameters developed using JUCE, was discovered. The implementation of the audio processing in this plugin with a basic user interface (Figure 4.2.1) set a benchmark for the DSP of further iterations of the plugin.

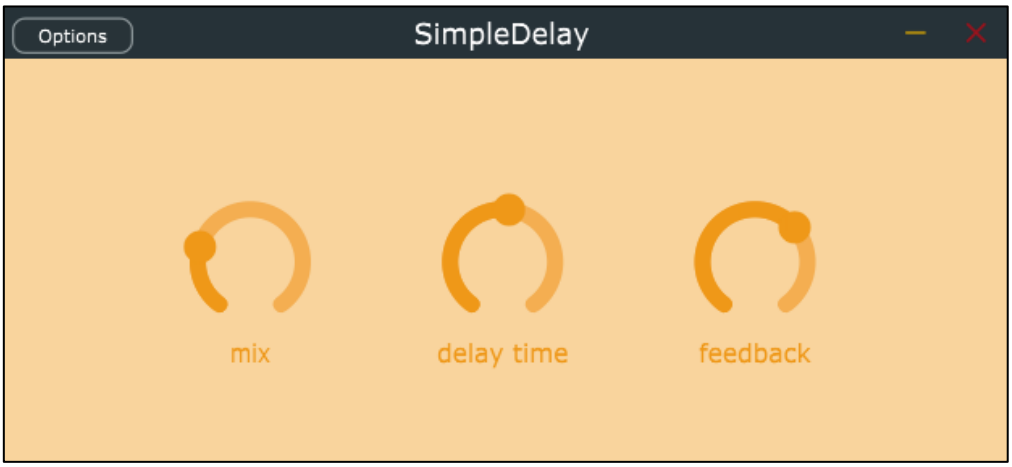

Figure 4.2.1 – Simple Delay Plugin using Daniel Walz's ffTapeDelay Implementation

Walz's delay effect consists of three main functions to implement the fundamental delay algorithm (Figure 4.2.2). In these functions input audio data is copied and added between the output audio buffer and a larger delay buffer to execute a classic delay effect.

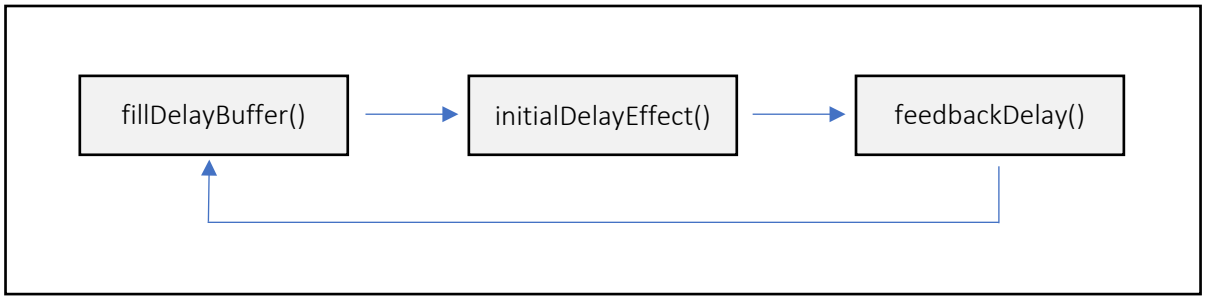

Figure 4.2.2 – Daniel Walz's ffTapeDelay Implementation

Following the implementation of delay in C++, the next iteration focussed on user interface design.

For the second sprint, a number of potential UI designs were drafted on paper, with consideration of the design observations from Chapter 2. One of these designs was then selected for digital development in Adobe Illustrator (Figure 4.2.3).

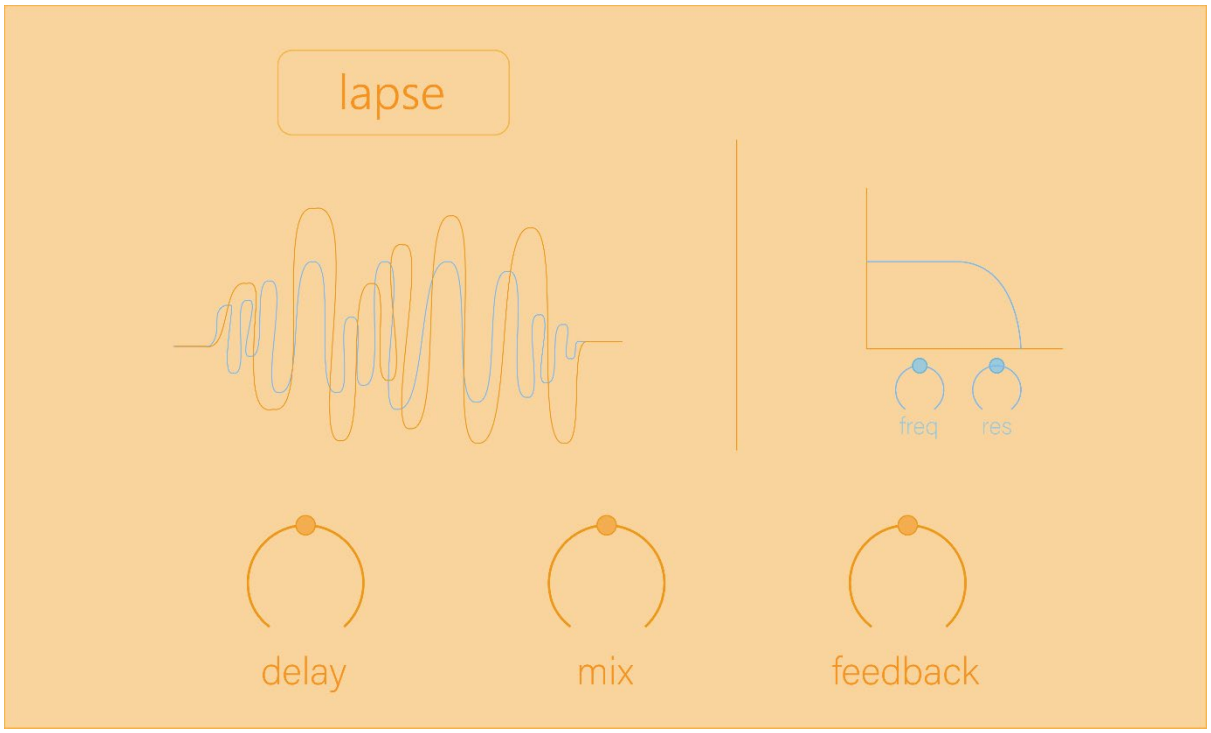

Figure 4.2.3 – Lapse (Sprint 2) – Flat UI design

The above design consisted of a flat, non-skeuomorphic user interface with vector-based line art displaying the input waveform in orange and the delayed signal in blue. Delay parameters could then be changed using the three knobs on the bottom third of the interface. When implementing this design in C++, additional DSP was implemented such as the state-variable filter.

With the second sprint of Lapse coming to a close, the current plugin's state was evaluated in terms of alternate user interface design against the project outcomes. Although the plugin was not skeuomorphic in appearance, the user interaction with the delay effect was still dictated by rotary sliders. As the goal of the project was to re-imagine interaction with audio effects through alternative user interface design, more experimentation with user interface was required to further develop this interaction.

Therefore, the third sprint began with new user interface design drafts, resulting in the initial node-based design (Figure 4.2.4) of the project. This design was heavily inspired by Cargo Cult's *Slapper* [5], a node-based multi-tap delay plugin.

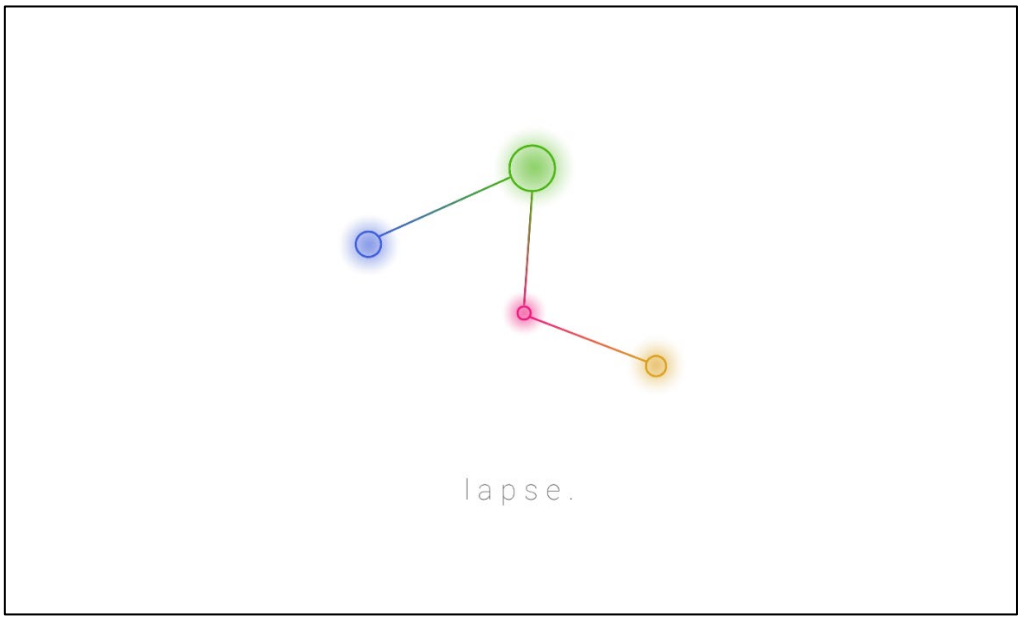

Figure 4.2.4 – Lapse 3rd Design Sprint; First Node-Based interface design of suite

This design featured the mapping of audio parameters to the x and y positions of each node. This position was intended to be changed by clicking and dragging on the node. The x-axis was intended to be coupled with the delay time of the effect, and the y-position was to be coupled with the feedback parameter. A timer locked to the bpm of the host would then cycle between nodes and their corresponding audio parameters.

After completing the UI design for this sprint, the GUI was implemented in C++. Following the previous sprint, there were three aspects of functionality to be implemented for the user interface to function:

1. Node-based GUI which responds to mouse-events
2. Synchronisation with the host bpm
3. Automation of both GUI and DSP parameters based on host bpm

Programming the node-based GUI was the starting point for the development phase of this sprint. This was continually iterated and improved upon throughout the course of both plugins. Developing this interface required the creation of a “Node” class which when passed mouse event information, could draw the nodes on the GUI window. Instantiating these

objects inside a std::vector data structure then afforded dynamic allocation and removal of nodes in response to user interaction.

Nodes could be added to the display by double-click and removed with right-click. Then clicking and dragging on a node would update the x and y position accordingly. This interaction seemed intuitive and instinctive based on common computer user interaction.

Once the nodes were successfully implemented and connected to the delay processing, it was time to evaluate the outcomes of the sprint against the goals of the project. Whilst the user interaction with the interface was quite unique and non-skeuomorphic, the visual design did not clearly communicate how the nodes affected the audio signal. Therefore, the sprint was concluded to further refine the GUI design.

The final sprint of *Lapse* involved redesigning the user interface as well as implementing pan and mix parameters, quantisation of delay time and automation to the bpm of the host. The final user interface design of *Lapse* resulted in splitting the UI into two separate fields and adding more visual cues to enhance user interaction (Figure 4.2.5).

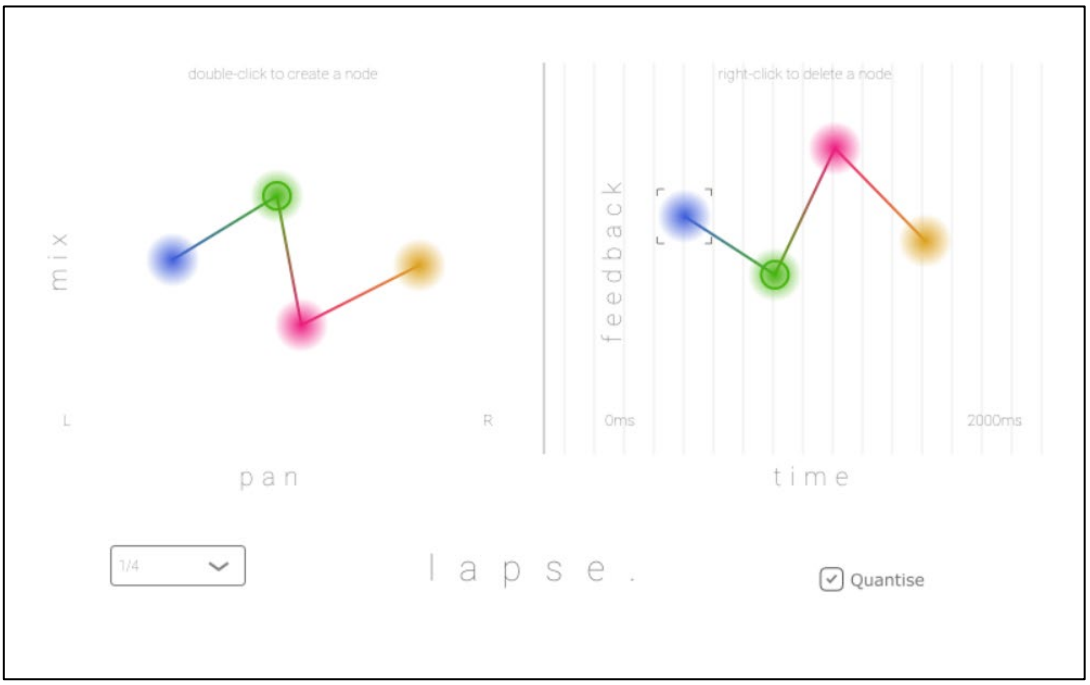

Figure 4.2.5 – Final Design of Lapse Prior to User Test

A further goal of the final sprint was to prepare *Lapse* for user testing which would be carried out upon the completion of the second plugin. Therefore, the plugin was tested in *Ableton Live 10* on Windows and Macintosh operating systems, as well as *Logic Pro X* on Macintosh, and Tracktion's *Waveform* on Linux. This revealed differences in how the various DAW's handled functions from the VST SDK. For example, Logic Pro X would pass different buffer sizes in prepareToPlay() than in processBlock(), whilst Ableton Live would provide consistent buffer sizes between both functions. These differences would need to be taken into account for *Lapse* and future plugins to function in multiple workstations.

As the node-based interface was successful in visualising delay processing, the next progression was to establish whether nodal design models were suitable to other audio effects. Following the trend of visualising time-based effects in the case study of Chapter 2, the second plugin was selected to be reverb. This resulted in the initial design for the plugin called *Space*.

4.3 Space

Space is a reverb plugin which uses node UI elements to visualise room size, dry/wet mix, and panning parameters. In the final design, the pink nodes represent the room being emulated by the reverb algorithm. Each of them, when selected and dragged with the mouse, increase or decrease the size of the room depending on drag direction. The blue node, restricted to the confines of the pink area, determines the dry/wet mix (y-axis) and panning position (x-axis) of the output signal from the reverb. When selected and dragged, the x and y positions of the blue node can be changed.

When designing Space, it was important that the user interface was strongly connected to the audio output of the system. In doing so, the interface would be easier to understand for the user, resulting in an interaction conducive to creative flow [6]. Therefore, the design process began with brainstorming different ways of visually representing the process of

reverb. By following the conclusions of the case study in Chapter 2, design elements such as colour, font, alternate metaphors, and XY pads were utilised to inform design decisions.

The initial visual design developed for Space (Figure 4.3.1) was very similar to that of Lapse in its visual connection between multiple nodes. This design was fully implemented in C++ yet, upon evaluation, it was decided that it did not intuitively represent the audio effect of reverb. This resulted in further thought into the function of reverb algorithms, which of course is to emulate a physical space within which to contextualise input audio. With this idea at the forefront of thought for the next design iteration, the final design was developed (Figure 4.3.2).

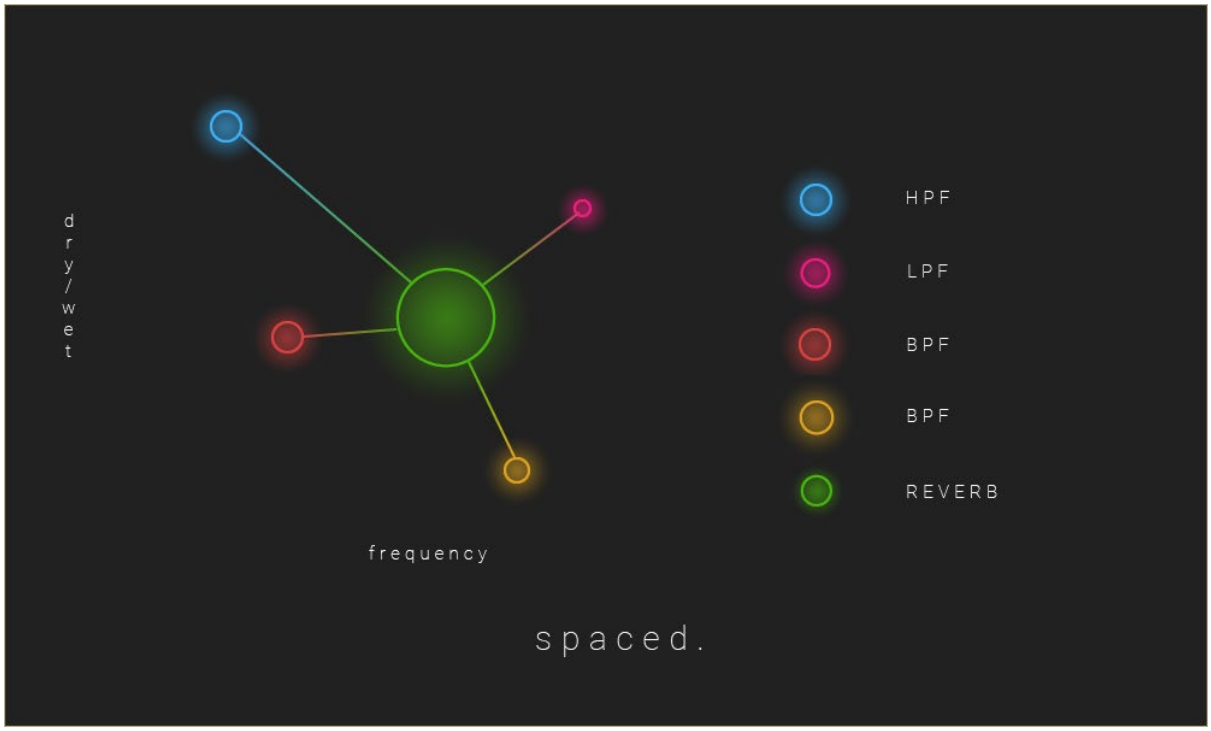

Figure 4.3.1 - First Design of Space

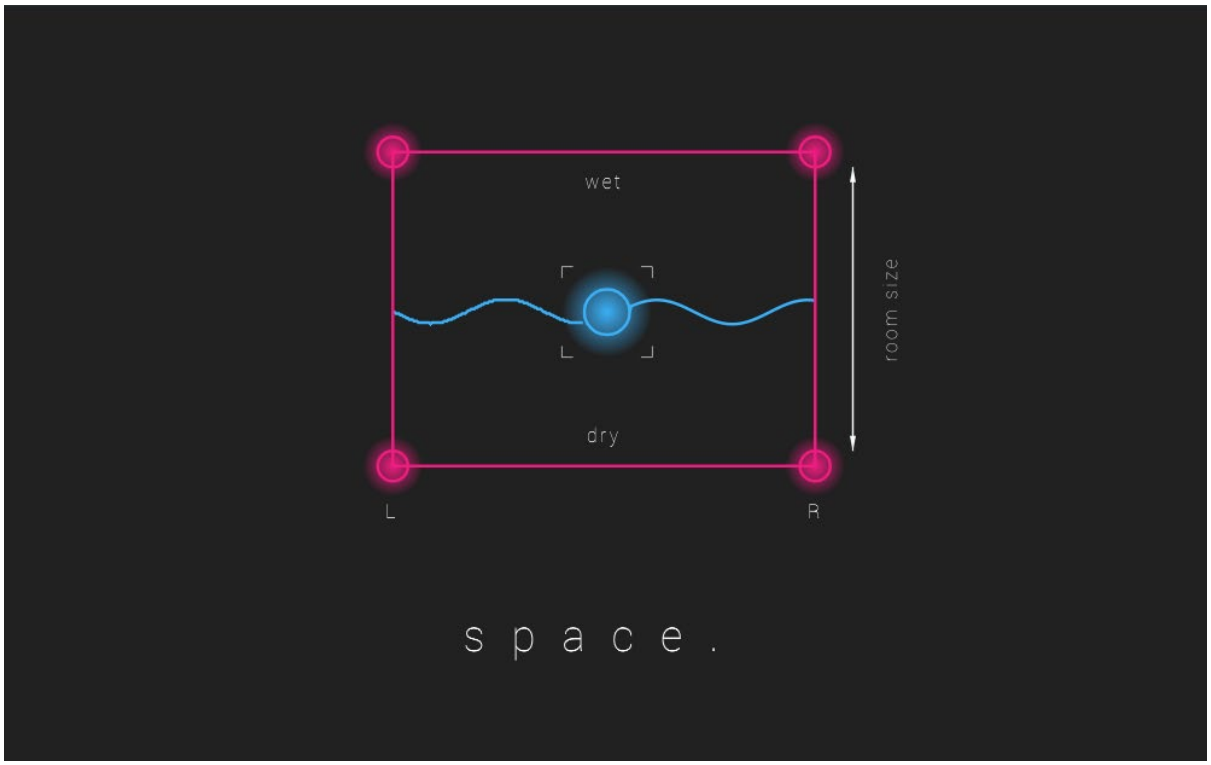

Figure 4.3.2 – Second Design of Space

In order for the user interface to function, the next step was to develop the DSP for the plugin. As the primary function of this research was to design intuitive user interactions through providing connections between user interfaces and DSP, it became clear that designing an algorithm from scratch was outside the scope of this project. Therefore, the reverb used for *Space* was JUCE's built-in reverb class; an open-source reverb implementation called *FreeVerb*. *FreeVerb* is a reverb implementation in C++ based off Schroeder's original reverb design comprising of a parallel bank of comb filters with a series of all-pass filters [21], [23]. However, whilst Schroeder's original design consisted of four comb filters in parallel and two all-pass filters, *FreeVerb* uses eight comb filters and four all-pass filters [23]. This results in a higher echo density and therefore a more convincing room sound when paired with the appropriate filter coefficients. This reverb implementation was

suitable for *Space* due to the quality of the algorithm, and the extensive documentation provided by the developers.

When applying *FreeVerb* to *Space*, the exposed room size parameter was scaled and assigned to the distance between a set of pink nodes, thus increasing or decreasing when a pink node is dragged. The dry/wet mix was assigned to the y-position of the blue node in relation to the room size. Panning was then applied to the reverbed signal using the same sinusoidal panning function in *Lapse*.

Upon completing *Space*, a user test was conducted of both plugins to provide external feedback on the user interface design and the interactive experience. This feedback would then be used to inform the ideation phase of the next iteration of the project.

4.4 User Test

The objective of this user study was to discover strengths and weaknesses in the usability and interactivity of *Lapse* and *Space* from the perspective of external users. This would provide information on whether the plugins were achieving the intuitive aspect of the final research goal; to develop a suite of audio plugins with alternative user interface design elements to explore intuitive user interaction with audio software. The results of the study would then inform further refinement of the two plugins to better address this research goal. This was achieved through assessing *Lapse and Space* in the context of musical composition/sound design.

The sample group for the user test consisted of a group of eight participants of various levels of experience with audio software. This sample size was ideal as the data received from the test was specific to each participant's experience with the software and reflected a range of skillsets and audio backgrounds within the target demographic of audio plugin users. However, the limitations of this sample were the lack of diversity in gender and age, having only one female participant to the seven male participants, and 87.5% of participants being between the ages of eighteen and twenty-four years.

The user test consisted of a qualitative survey comprised of both ordinal scales and open-ended questions. Participants completed these questions after use of both plugins in the participant's own native creative environments. This was designed to facilitate a natural interaction with the plugin from the user's perspective. Further documentation such as the information sheet, participant consent form, and the survey questions for the user test can be found in the appendix attached.

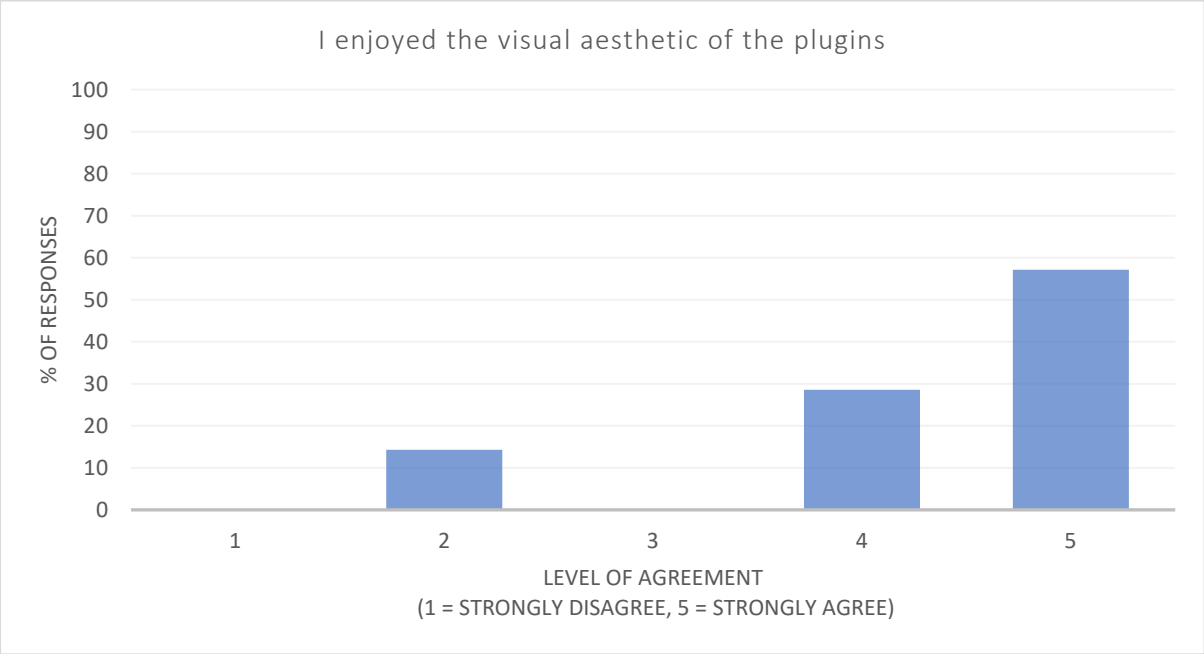

Figure 4.4.1 – User Study: Level of Enjoyment of Aesthetic

From the user test it became clear that the aesthetic of the plugins was enjoyable (Figure 4.4.1), with one user stating "Really nice GUI and awesome to play around with the features. Easy to use and looks great". However, the level of intuitiveness in comparison to traditional user interfaces varied between plugins (Figure 4.4.2, Figure 4.4.3).

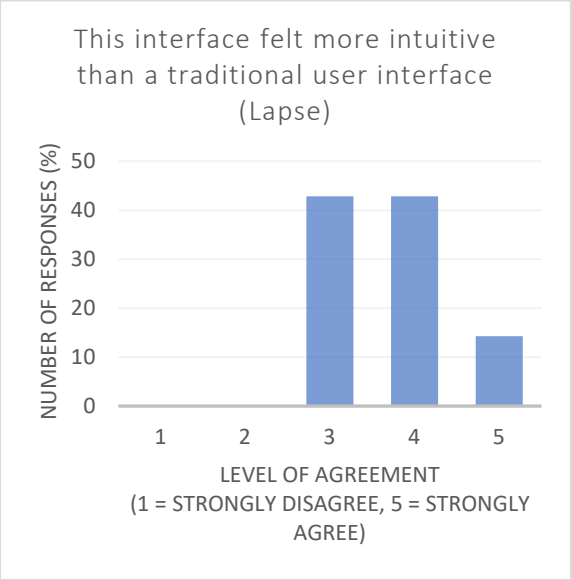

Figure 4.4.2 User Study: Lapse's Level of Intuitiveness

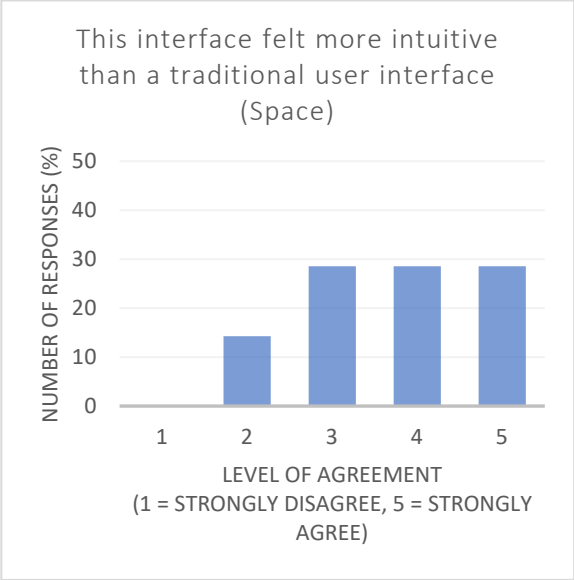

Figure 4.4.3 User Study: Space's Level of Intuitiveness

These results indicated that whilst most respondents found the interfaces intuitive in comparison to traditional user interfaces, there is still opportunity to improve upon the node-based UI designs. More experienced DAW users tended to request more functionality, with one participant suggesting a stereo widener in *Space* instead of its pan feature. Whilst less experienced users tended to request clearer visualisations of existing features, particularly in *Lapse* which is the more visually complex plugin. This data suggests that in the next iteration of the suite, there should be further refinement of visual design in both Lapse and Space, as well as a review of audio parameters for user interaction.

4.5 Lapse 2.0

Based on the user study, the final iteration of *Lapse* began with a list of features to implement. These features were as follows:

1. Remove the need for two node fields
2. Improve visual communication of quantise function
3. Utilise node diameter for functionality

As some users found the two node areas confusing, the visual design of *Lapse* needed to be simplified. Therefore, a new iteration of the interface was developed (Figure 4.5.1).

The final design of *Lapse* features a single node field with delay time and pan parameters. The mix parameter was completely removed from user interaction and instead set to a constant variable. Feedback for each node was then scaled to its diameter, which could be changed with shift-click and mouse drag. Line weight variation and line dashes were then added to better visually display the quantise grid's functionality.

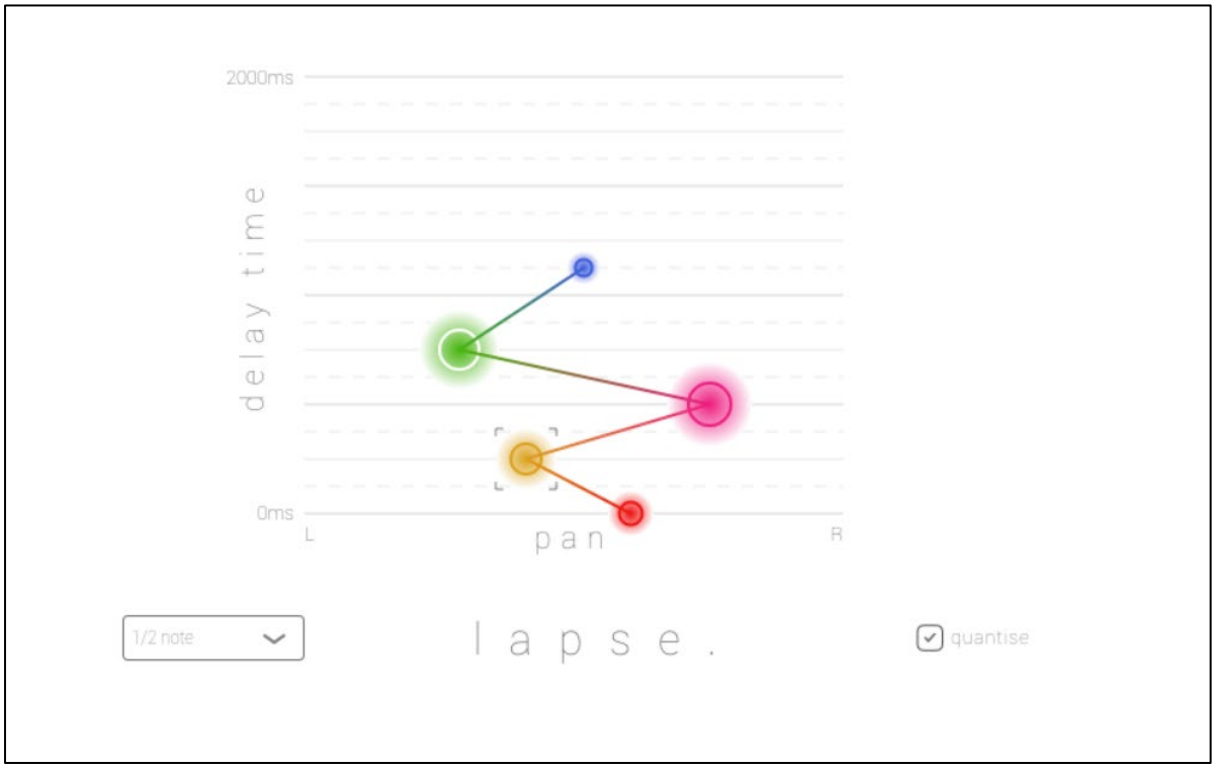

Figure 4.5.1 – Final Design of Lapse

4.6 Space 2.0

The main focus for *Space* following the user test was cleaning up visual glitches regarding interaction as well as improving functionality. From the user test the following changes were to be made:

1. Implement room width parameter to user interface
2. Alter panning function to only effect dry signal
3. Clean up visual glitches

These changes did not require a complete redesign of the user interface, rather they required minor adjustments to the user interaction with the pre-existing visual design. In order to implement width into the reverb, it was required to decouple the width and height of the reverb area. This allowed decay time to be attributed to the height of the area and stereo widening to the width. Through implementing this feature, the concept of the box representing a room translated much clearer in the interface. Combining this with a dry signal pan tied to the x-position of the blue node helped further communicate the analogy of a sound source being positioned inside the room. Figure 4.6.1 depicts the final design of *Space*.

4.7 Conclusion

This chapter has discussed the development and user testing of two new audio plugins which feature a node-based user interface. *Lapse* utilised x and y positioning as well as the diameter of multiple nodes to cycle through audio parameters in a delay DSP algorithm. *Space* visually represented reverb processing through assigning room size and width parameters to the height and width distances between four nodes. A single node's x and y positioning within that area represented the positioning of an audio source in a room using panning and dry/wet parameters. Both plugins went through an iterative process to reach the final user interface design, with a user test preceding and informing the final iteration of each plugin.

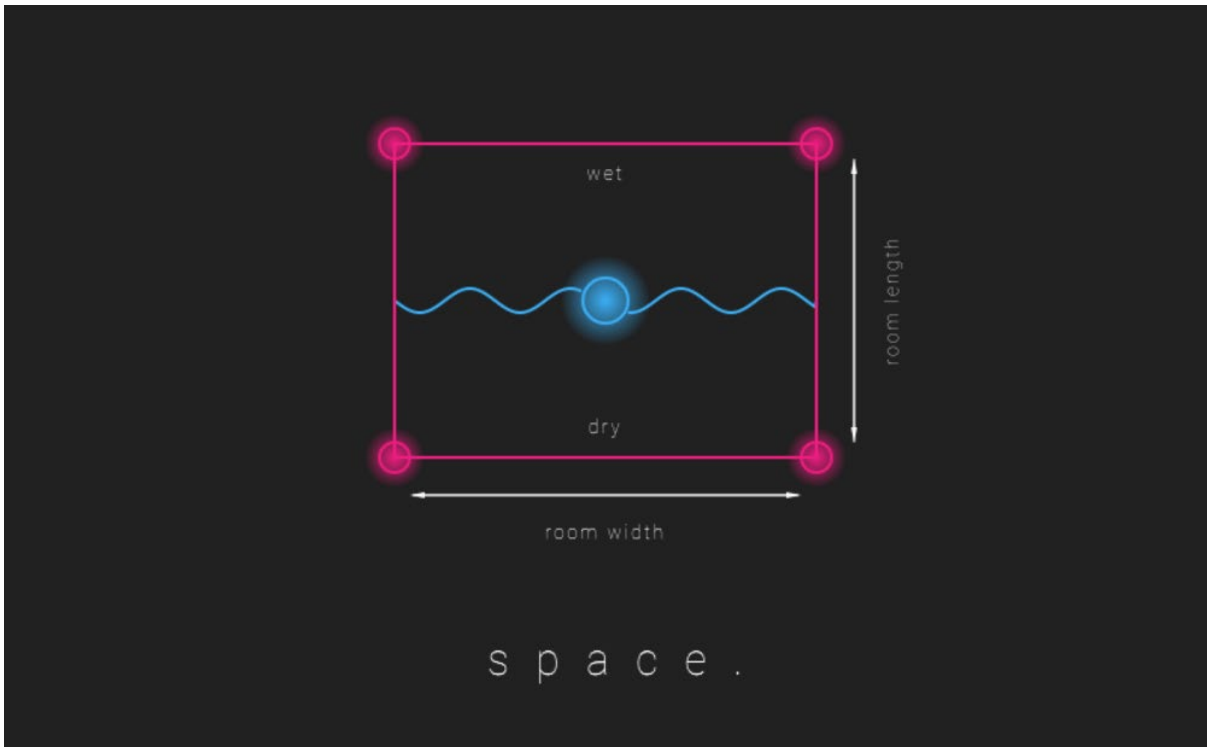

Figure 4.6.1 – Final Design of Space

5. Conclusion

5.1 Overview

This conclusion examines the development and final iteration of *Lapse* and *Space* in the context of the research goals stated in Chapter 1:

1. Investigate the paradigms of audio software user interface design.
2. Investigate the works of those who have questioned these paradigms.
3. Develop a suite of audio plugins with alternative user interface design elements to explore user interaction with audio software.

Following this examination, the implications of this study on the wider field of audio plugin development as well as the potential for future works is discussed.

5.2 Discussion

Current audio software user interface design leans heavily on skeuomorphic design principles in emulation of analogue audio hardware [8]. Whilst once useful for navigating new digital technology, skeuomorphic design in audio software has now been supplanted and is no longer user-centric in focus. However, the aesthetic value of skeuomorphic design in audio software is high for certain target audiences in the context of developing a commercial product [13]. Therefore, when developing new products, it is important to decide whether skeuomorphic design will help or hinder the user experience of the final product.

The *Nodal Plugin Duo* explored non-skeuomorphic user interface design in audio plugins. When developing the user interfaces for the duo, it was important to consider a variety of techniques to visually communicate ideas and relationships between user interface elements. Such examples of these are alternate metaphors, colour, font, and synchronisation between audio and visual elements. These concepts were applied to the development of both *Lapse* and *Space*. Both of these plugins exhibit node-based interfaces. Other examples of works

which utilise this interface model are Timothy Barraclough's *Pyxis Minor [4]*, The Cargo Cult's *Slapper [5]*, and many modern parametric EQ plugins on the market such as *Equilibrium [17]* and Logic Pro's *EQ* [15]. After several iterations of development, a user study was then carried out to measure the usability and interactivity of the *Nodal Plugin Duo*.

Based on the results of the user study, the node-based user interface design model proved to be very intuitive for the time-based effects of delay and reverb which were utilised in *Lapse* and *Space*. However, in order to further draw conclusions regarding the versatility of the node-based model, it would be beneficial to develop nodal plugins that utilise dynamics processing, such as compression. This would contribute towards a more robust user interface design model capable of being applied to a range of audio software not entirely limited to music production.

Node-based design models could be particularly effective in the field of game design, as nodes could be used to represent game objects as well as the 3D spaces which these objects are contained within. Integrating these tools into game engines such as Unity and Unreal Engine would allow for developers to more effectively express sonic environments and, furthermore, create more immersive user experiences. The viability of audio software user interface design in the industry vertical of game development is discussed further in the business plan which accompanies this exegesis.

5.3 Closing Remarks

This exegesis has examined the field of audio plugin development and investigated both conventional and alternative user interface design methodologies. From exploring these methodologies, the *Nodal Plugin Duo* was developed and tested by musicians, sound designers, and audio engineers. Through this process, it has been made clear that there is potential to expand upon these works and reimagine new, intuitive ways to interact with audio software.

6. Citations

[1] B. Guerrero, “An Analysis of Audio Mixing Console Designs, Recording Software, and a Look at Future Interfaces Using Intuitive Input Technologies,” p. 47, 2012.

[2] P. Barr, R. Biddle, and J. Noble, “A Semiotic Model of User-Interface Metaphor,” in *Virtual, Distributed and Flexible Organisations*, K. Liu, Ed. Dordrecht: Kluwer Academic Publishers, 2005, pp. 189–215.

[3] "What is Skeuomorphism?", The Interaction Design Foundation, 2019. [Online]. Available: https://www.interaction-design.org/literature/topics/skeuomorphism.

[4] T. J. Barraclough, D. A. Carnegie, and A. Capur, “Musical Instrument Design Process for Mobile Technology,” In Proceedings of New Interfaces for Musical Expression (NIME), 2015.

[5] J. Webster, "Slapper - The Cargo Cult", Thecargocult.nz, 2014. [Online]. Available: http://www.thecargocult.nz/slapper.shtml.

[6] K. Bjørn, M. Metlay, P. Nagle and J. Jarre, Push, Turn, Move. Copenhagen: Bjooks Media, 2017.

[7] M. Duignan, J. Noble, P. Barr, and R. Biddle, “Metaphors for Electronic Music Production in Reason and Live,” in *Computer Human Interaction*, vol. 3101, M. Masoodian, S. Jones, and B. Rogers, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 111–120.

[8] M. Marrington, “Paradigms of music software interface design and musical creativity,” p. 10, 2016.

[9] "Soundtoys 5", Soundtoys, 2019. [Online]. Available: https://www.soundtoys.com/.

[10] "MASSIVE", *Native-instruments.com*, 2019. [Online]. Available: https://www.native-instruments.com/en/products/komplete/synths/massive/.

[11] "Abbey Road Plugin Collection | Bundles | Waves", *waves.com*, 2019. [Online]. Available: https://www.waves.com/bundles/abbey-road-collection?gclid=Cj0KCQjw_5rtBRDxARIsAJfxvYD94ZuaO8eV-AcJ7PGx-hfEoGrwbmiWFQlQaB5V5CqLO2Te4urVkYUaAoMfEALw_wcB#legendary-analog-sound-abbey-road-collection.

[12] D. Schrey, “Analogue Nostalgia and the Aesthetics of Digital Remediation,” in *Media and Nostalgia*, 2014, pp. 27–38.

[13] A. Williams, “Technostalgia and the cry of the lonely recordist,” *Journal on the art of record production*, no. 9, 2015.

[14] P. Myllys, “User Interface Paradigms in Digital Audio Workstations,” p. 126, 2014.

[15] "Logic Pro X - Plug-ins and Sounds", Apple (New Zealand), 2019. [Online]. Available: https://www.apple.com/nz/logic-pro/plugins-and-sounds/.

[16] "FabFilter Pro-Q 3 - Equalizer Plug-In", Fabfilter.com, 2019. [Online]. Available: https://www.fabfilter.com/products/pro-q-3-equalizer-plug-in.

[17] "DMG Audio : Products : EQuilibrium", Dmgaudio.com, 2019. [Online]. Available: https://dmgaudio.com/products_equilibrium.php.

[18] "Obscurium | Synthesis Tool with VST Host & Generative Pitch.", Sugar Bytes, 2019. [Online]. Available: https://sugar-bytes.de/obscurium.

[19] "OP-1", *Teenage.engineering*, 2019. [Online]. Available: https://teenage.engineering/products/op-1.

[20] S. Balaji and M. Sundararajan Murugaiyan, "WATERFALL Vs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC", International Journal of Information Technology and Business Management, vol. 2, no. 1, pp. 26-30, 2012. Available: https://www.jitbm.com/Volume2No1/waterfall.pdf.

[21] W. C. Pirkle, *Designing audio effect plug-ins in C++ with digital audio signal processing theory*. Burlington, MA ; Abingdon, Oxon: Focal Press, 2013.

[22] D. Walz, "ffAudio/ffTapeDelay", GitHub, 2019. [Online]. Available: https://github.com/ffAudio/ffTapeDelay. [Accessed: 16- Oct- 2019].

[23] J. Smith, "PHYSICAL AUDIO SIGNAL PROCESSINGFOR VIRTUAL MUSICAL INSTRUMENTS AND AUDIO EFFECTS", Ccrma.stanford.edu, 2010. [Online]. Available: http://ccrma.stanford.edu/~jos/pasp/.

7. Appendix

7.1 Information Sheet

MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

Knobs and Nodes: A Study of User Interface Design in Audio Plugins

INFORMATION SHEET

Project Description and Invitation

Dear Participant,

I would like to invite you to participate in a research project called *'Knobs and Nodes: A Study of User Interface Design in Audio Plugins'*. The objective of this user study is to assess the usability, interactivity, and commercial viability of a suite of audio plugins for music production. The plugins themselves have been created with the target audience of music producers and sound designers in mind, which has led to the selection of you, the participant. Participants will be required to use the software in your own native creative environment and will provide information and insight on your use of the software via a survey. This information will then be utilised for the development of further iterations of the plugin suite.

Participation

Participants are invited to create music using a suite of audio plugins with node-based user interfaces. Throughout this process there will be a questionnaire/survey pertaining to the usability and interactivity of the Nodal Plugin Suite.

If any participants desire to withdraw from the user study at any point, it is perfectly within their rights to do so without further questioning. In order to withdraw, the participant should contact the primary investigator via the email address provided below, or in person if applicable.

All information acquired from the study will be evaluated and may be referred to anonymously within the exegesis for the project. All personal details of participants will be kept confidential and all opinions expressed throughout the course of the study will remain anonymous. Access to the data acquired as a result of this study will be restricted to only the investigation team listed below.

Participant's Rights

You are under no obligation to accept this invitation. If you decide to participate, you have the right to:

- *decline to answer any particular question;*
- *withdraw from the study (specify timeframe);*
- *ask any questions about the study at any time during participation;*
- *provide information on the understanding that your name will not be used unless you give permission to the researcher;*
- *be given access to a summary of the project findings when it is concluded.*

Context

Participants will be asked to undergo their own creative practice utilising the Nodal Plugin Suite. The Nodal Plugin Suite will comprise of three audio plugins presented in the VST format. Each plugin is designed with a node-based user interface which will afford an alternative means of controlling audio parameters and, hopefully, a more creative experience than traditional user interfaces. The investigators are therefore interested in gathering user opinions regarding their experience with the plugins, and furthermore, the practicality of the plugins within a creative environment.

Results

The final results from this study may be published in academic journals, and technical reports. The results will be published in the primary investigator's exegesis at the end of his Masters. If any participants in the study are interested in seeing the results, they are welcome to contact the primary investigator at the email address below for more information.

We very much appreciate your participation in this user study, and hope you enjoy using the Nodal Plugin Suite.

Researchers

Primary Investigator
Jonathan McGregor
Masters Candidate
School of Commercial Music and Creative Media Production
Massey University of Wellington
Email: jonny.mcgregor1@gmail.com

Investigator
Bridget Johnson
Supervisor
School of Commercial Music and Creative Media Production
Massey University of Wellington
b.d.johnson@massey.ac.nz

Investigator
Jon He
Supervisor
School of Commercial Music and Creative Media Production
Massey University of Wellington
j.he1@massey.ac.nz

7.2 User Study Survey

This questionnaire is designed to survey the overall user experience of interacting with the Nodal Plugin Suite. Please complete the questionnaire for each plugin after use in your own creative practice. Feel free to use the plugins in any way you see suitable.

Interaction

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
It was obvious how to interact with the plugin.					
I found it easy to change parameters within the plugin.					
The visual interface of the plugin suited the audio output of the system.					
This interface felt more intuitive than a traditional user interface i.e. knobs and sliders.					

User Satisfaction

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I was satisfied with the control I had over parameters within the plugin.					
I enjoyed the sonic result of the plugin					
I enjoyed the visual aesthetic of the plugin					
The plugin inspired my creativity					
I would use this plugin for my own projects outside of this study.					

If there were any features you thought should have been available, please list here:

If there were any issues or bugs with the plugin, please list here:

Additional comments? Please list here:

Low Risk Notification

This project has been evaluated by peer review and judged to be low risk. Consequently, it has not been reviewed by one of the University's Human Ethics Committees. The researcher(s) named above are responsible for the ethical conduct of this research.

If you have any concerns about the conduct of this research that you wish to raise with someone other than the researcher(s), please contact Prof Craig Johnson, Director, Research Ethics, telephone 06 356 9099 x 85271, email humanethics@massey.ac.nz.

7.3 Participant Consent Form

MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

Knobs and Nodes: A Study of User Interface Design in Audio Plugins

PARTICIPANT CONSENT FORM - INDIVIDUAL

I have read, or have had read to me in my first language, and I understand the Information Sheet attached as Appendix I. I have had the details of the study explained to me, any questions I had have been answered to my satisfaction, and I understand that I may ask further questions at any time. I have been given sufficient time to consider whether to participate in this study and I understand participation is voluntary and that I may withdraw from the study at any time.

1. I wish/do not wish to have data placed in an official archive. (if applicable include this statement)
2. I agree to participate in this study under the conditions set out in the Information Sheet.

Declaration by Participant:

I _____hereby consent to take part in this study.
[print full name]

Signature: _____ Date: _____

7.4 Audio Plugin Case Study

Plugin Name	Company	Plugin Type	Observations	Colours	Param Controls	Font
Backmask	Freakshow Industries	Reverse Effect	-Follows Freakshow's aesthetic of horror plugin both in terms of visuals and in audio produced from it. -Interface consists of knobs and buttons.	5 - Plain background and shadows Glowy red to make the gui pop	Knobs and Buttons	Sans, all Caps
Hologram	Sinevibes	Synth	-Tidy, minimal visual style. -Lots of sliders. -Frequency content represented by bar graph	4 - Dark grey background, Bright adjacent colours	Sliders	sans, all lower case
Delay Lama	Audionerds	Synth (meme)	- Animated 3D monk which sings in realtime with the audio and matches the vowel sound - Interface consists of knobs and sliders	<10 - Various shades of Red, Yellow and Blue	Knobs and Sliders	sans, all caps
Synplant	Sonic Charge	Synth	- ui follows the metaphor of growing a plant from a seed - click on the seed to grow branches from it - use sliders to modify parameters further	3 - Grey background, bright green and red stand out on backround	Circular XY pad, Sliders	sans, lower case params, all caps title
Cygnus	Krakli	Synth	- parameters are represented by stars - sparse labeling encourages experimentation	<10 - Full use of colour spectrum for different galaxies	Knobs, Buttons, Sliders	n/a
Vurtbox	Krakli	Synth	- "Droopy" aesthetic - Erratically placed params - Also focusses on experimentation through lack of labeling	3 - Red, blue and white/grey	Knobs, Buttons, Sliders	n/a
Meow Synth	Knobster	Synth (meme)	- Synth sounds like a cat - Cat is animated to move on midi note event - GUI consists of knobs and sliders	3 - Blue background, yellow param values, dark grey button, cat is also shades of yellow	Knobs and buttons	Sans, all caps
Cyclop	Sugarbytes	Synth	- Scifi/Robotic aesthetic - Control surface in centre provides information based on the module selected - Busy interface made clearer by bright colours on dark background	<10 - dark background, variety of brighter colours to make the display easier to handle for the user	Knobs, Buttons, Sliders	Sans, all caps
Obscurium	Sugarbytes	Arp/Sequencer	- Bright colours, dark background - Colours on main area match with corresponding effects down side bar	>10 - dark background, bright colours	XY Time Grid, Knobs, Sliders, Buttons	sans, all caps
OP-1	Teenage Engineering	Hardware Synth	- Line-art Aesthetic - Colour matching between visuals and knobs	5 - dark background, bright colours	Hardware Knobs, Hardware buttons	sans, all caps

Molekular	Native Instruments	Effects Unit	- Point of difference is the morpher unit in the centre - You can assign multiple parameters to the morpher knobs, then modulate between the knobs using the 2D pad. This movement is displayed by a moving dot with a motion trail behind it.	6 - Blues for the main panel, Muted colours for the other components	XY Pad, Knobs, a few sliders and buttons	sans, camel case
Tricky Traps: Radar	Sonic Faction	Sequencer	- sequencer displayed on pie graph - Colours on graph linked to corresponding notes	5 - Dark Background, bright gui elements	Buttons	sans, all caps
Hypermorph	Sonic Faction	Synth	- 4 sound sources - Mix between sources using XY pad - Automation on XY allows a sound that changes over time	5 - dark background, bright gui elements	XY Pad, Buttons, Sliders, knobs	sans, all caps
Waverazor	Mok	Synth	- Shape of output waveform shown in centre of gui - Futuristic/scifi aesthetic acheived through vivid colour scheme and circular shape language	3 - black background, red and blue gui elements	XY Pad, Knobs	sans, all caps
Gravity	Heavyocity	Synth	- Centre knob has a range of parameters simultaneously assigned to it - Futuristic aesthetic to represent the sounds produced from the plugin	3 - black background, yellow and blue gui elements	Knobs	sans, all caps
Movement	Output	Effects Unit	- 4 different effects processors - 1 processor on each corner of the XY pad - Dark background with bright gui elements - use of tabs to organise higher functionality whilst keeping the gui tidy	3 - dark background, blue and orange gui elements	XY Pad, Knobs, Buttons	sans, lower case params
Sparkverb	Uvi	Reverb	- reverb displayed on freq domain graph - Colour is used to represent frequencies present in reverb - Presets are plotted on graph, interpolation allowed between them, beats scrolling through a preset menu	Black background, full colour spectrum	Knobs	sans, all caps
Schep's Parallel Particles	Waves	Parallel Processng	- Futuristic/SciFi aesthetic - 4 knobs all mix levels for different parallel processing algorithms. - Bright GUI elements, dark background	6 - Dark background, blue, yellow, green, orange for knobs, splashes of purple	Knobs	sans, all caps
Flow Motion	Waves	FM Synth	- 4 osc synth - waveform displayed in centre - mainly blue colour scheme - submenus hidden until parameters are clicked on	3 - Blue, Black, Yellow	Knobs, Sliders, Buttons	sans, lower case