Evaluating the Usability of Software Synthesizers: An Analysis and First Approach

by

Curtis E. Rasmussen

A Thesis
presented to
The University of Guelph

In partial fulfilment of requirements for the degree of
Master of Science in
Computer Science

Guelph, Ontario, Canada

©Curtis E. Rasmussen, March, 2018
Software synthesizers play an incredibly large role in the production of modern day music. However, their usability remains largely ignored in the literature. Software synthesizers have complex user interfaces (UIs), and tend to eschew the design paradigms of more mainstream software. Instead their UIs are often recreations of those found on physical synthesizers. As a result, software synthesizers are difficult for new users to learn and use. This thesis investigates how synthesizers’ UIs affect usability for novice users, as well as the level of engagement felt by novice users. Novice participants were asked to use one of two commercial synthesizers over two sessions and to compete a survey measuring engagement. Results show significant usability issues in synthesizer design choices, as well as poor levels of engagement.
ACKNOWLEDGEMENTS

Thanks to my advisor, Blair Nonnecke, for giving me the opportunity to explore a research field of my choosing, and putting up with my procrastinating ways. To Dave Wickland, for being immensely helpful when refining my thoughts. To my family, for their constant support, as well as their constant nagging to hurry up and finish. To my friends, for never believing me when I said I was close to finishing, and pushing me to work when I was working through some tough times.
# Table of Contents

## List of Figures

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>vi</td>
</tr>
</tbody>
</table>

## 1 Introduction

1.1 An introduction to musical synthesizers ................................. 1
1.2 Goal of this thesis ...................................................... 2
1.3 Coming Chapters ......................................................... 3

## 2 Literature Review

2.1 Defining a Synthesizer ...................................................... 4
2.2 Software Synthesizer Usability ........................................... 6
  2.2.1 Evaluation Frameworks .................................................. 9
  2.2.2 Analysis of Interfaces ................................................ 11
2.3 Factors Affecting Software Synthesizer Usability ......................... 11
  2.3.1 UI Complexity and Feedback ......................................... 11
  2.3.2 Prerequisite Knowledge ............................................... 12
  2.3.3 Software Synthesizer Usability for Novice Users ................... 13
  2.3.4 Non-Standard Controls ............................................... 14
  2.3.5 Visual Design of the UI .............................................. 15
  2.3.6 Input Methods ....................................................... 23
2.4 Engagement ................................................................. 24
  2.4.1 Defining Engagement .................................................. 24
  2.4.2 Types of Engagement ................................................ 26
  2.4.3 Comparison to Video Games ......................................... 27
  2.4.4 Measuring Engagement ............................................... 28
2.5 Thesis Statement .......................................................... 30

## 3 Methodology

3.1 Two Synthesizers for Comparison ......................................... 31
  3.1.1 Synthesizer 1: Massive ............................................... 31
    Justification for Inclusion ............................................. 33
  3.1.2 Synthesizer 2: Rounds ................................................ 33
    Justification for Inclusion ............................................. 35
  3.1.3 Critical Differences ................................................ 36
3.2 Demographics ............................................................... 40
3.3 Equipment ................................................................. 41
3.4 Procedure ................................................................. 42
4 Results and Discussion
4.1 Usability Issues for Novices 46
  4.1.1 Visual Design & First Impressions 47
  4.1.2 Non-Standard Controls 49
  4.1.3 Building a Mental Model 52
  4.1.4 Understanding Current State 55
  4.1.5 Functionality Awareness 57
    Failure to explore 58
    Failure to find 59
    Failure to understand 59
  4.1.6 Action Feedback 60
  4.1.7 Importance of Presets 61
  4.2 Engagement 61
5 Conclusions 63
  5.1 Research Highlights 63
    5.1.1 Usability for Novices 63
    5.1.2 Engagement 65
    5.1.3 Evaluation Methodology 66
  5.2 Limitations of Work 66
  5.3 Future Work 67

References 68

Appendices 73
List of Figures

2.1 The VAZ Modular synthesizer by VAZ Synths, released in 1998. [9] . . . . 5
2.2 Image Line SimSynth was released in 1994, making it one of the earli-
est software synthesizers. Image Line continues to make computer music
software. [10] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
2.3 Seer Systems Reality is another synthesizer made in the 1990s, this one has
an interface consisting of operating provided widgets. [8] . . . . . . . . . . 7
2.4 Sequential Circuits Prophet 5, a hardware synthesizer. Compare with the
following image showing the digital recreation. [6] . . . . . . . . . . . . . . . . 8
2.5 Native Instruments Pro-Five, the software recreation of the Prophet 5 hard-
ware synthesizer. [5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
2.6 iraisynn.attinom Araucann Synthesizer, which has an overwhelming amount
of controllable options. [1] . . . . . . . . . . . . . . . . . . . . . . . . . . 12
2.7 The HY-MONO synthesizer, which makes no use of standard operating
system widgets [34]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
2.8 An example of the “knob” controls, this on from the Massive synthesizer
made by Native Instruments [43]. To change the value, a user must click
on the widget and drag their mouse up and down to modify the value. . . . 16
2.9 The entire UI of Massive, note the yellow controls in the bottom right of
the interface. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
2.10 Propellorhead’s Reason - the “front” of the synthesizers. . . . . . . . . . 19
2.11 Propellorhead’s Reason Software - the “back” of the synthesizers. . . . 20
2.12 The Mini V, created by Arturia to mimic the sound and UI of the hardware
based Minimoog. [4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
2.13 The Farfisa V, also created by Arturia, to mimic the Farfisa organs from the
1960s. [2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
2.14 Zebra 2, the tabs are circled in red. . . . . . . . . . . . . . . . . . . . . . 22
2.15 An example of the labels used by the synthesizer Dexed [50]. . . . . . . 23
3.1 Massive’s user interface when initially loaded . . . . . . . . . . . . . . . . 32
3.2 Rounds’ user interface when initially loaded . . . . . . . . . . . . . . . . . 34
3.3 How Rounds indicates that a module is disabled (left) versus enabled (right) 35
3.4 Massive’s ADSR Envelope Access Widget, highlighted in red . . . . . . . 37
3.5 Round’s Output Control Area, knobs with different functionality are high-
lighted with different colours . . . . . . . . . . . . . . . . . . . . . . . . . 38
3.6 Round’s Oscillator Control Area . . . . . . . . . . . . . . . . . . . . . . . 39
3.7 Massive’s Oscillator Control Area . . . . . . . . . . . . . . . . . . . . . . 39
3.8 AKAI LPK 25 MIDI Controller . . . . . . . . . . . . . . . . . . . . . . . . 41
3.9 The diagram used to explain the basic components of a synthesizer . . . . . 43
4.1 Massive’s use of “n.a.” to identify widgets with no functionality . . . . . 48
4.2 The bottom half of Rounds’ interface . . . . . . . . . . . . . . . . . . . . . 49
4.3 The non-delineated buttons present in the Rounds interface. The text cir-
cled in red are the only areas present which CANNOT be clicked on . . . . . 50
4.4 The mental model that the participant developed during usage . . . . . . 53
4.5 A depiction of the actual model for oscillator and filter signal flow . . . . . 54
4.6 The signal path diagram Massive provides, all the grey buttons are clickable 55
4.7 The area highlighted in red is the area that participants glossed over . . . . 58
4.8 The image on the left is an module that is enabled, while the image on the
right is a module that has been disabled . . . . . . . . . . . . . . . . . . . . 60
5.1 On the left, a control point on Helm when underneath the mouse pointer.
Compare it to the similar control on the right that is not selected. . . . . . . 64
Chapter 1. Introduction

1.1 An introduction to musical synthesizers

Since their proliferation in the mid 1900s, electronic synthesizers have had an unmistakable effect on the creation of music. Synthesizers gave musicians unprecedented control over timbre and sonic quality. Spawning entire genres of music, such as EDM (electronic dance music), IDM (intelligent dance music), trance, techno and drum-n-bass, synthesizers have also found success in mainstream music, from the poppy bass lines in 1980s pop, to the almost entirely electronic pop songs of today [36].

One of the technologies that allowed musical synthesizers to proliferate to such an extent were digital oscillators (a core component of synthesizers), which brought down the cost of hardware synthesizers dramatically. On the input side, the MIDI protocol opened up the synthesizers to many different types of controllers (e.g., electronic keyboards and drum kits), allowing companies to create more varied interfaces and control schemes. Today ubiquitous, fast home computers have enabled even more users to begin producing electronic music at home, using very sophisticated software synthesizers [3].

While software synthesizers are relatively inexpensive, sound manipulation is a difficult task; it is at its core a complex mathematical problem. Transforming this problem into a UI (user interface) that can be used by anyone is difficult. Many of the software synthesizers currently on sale have a large number of controls and options, leading to a multitude of
configurations, options, and decision choices for the user. To make things more difficult for the user, the terminology used in the industry and within the UIs tends to be unique to synthesizers and is often referred and labeled through mnemonics and abbreviations. The UIs are unlike other common software applications (e.g., Microsoft Word and iTunes). The end result is difficulty musicians, especially for the novice.

1.2 Goal of this thesis

Software synthesizers present an interesting research opportunity in HCI (human-computer interaction). They can be seen as distinct from more results or task-oriented software, instead providing a canvas for exploration and play. This difference in focus has led to synthesizers with UI controls and norms not seen in traditional software [57].

Interestingly, software synthesizers have been largely ignored in the HCI literature. Where there has been discussion, such as the more music-leaning literature, they are seen as being closely related to hardware synthesizers. From an HCI perspective, the difference between a physical interface and a software interface is stark. Fundamentally, one interacts with the two in different ways. Where a physical synthesizer can be manipulated directly, a software synthesizer will commonly be manipulated with a mouse. Furthermore, the design language found in software synthesizers is found to be very different than more mainstream software, with a large focus on skeuomorphism.

The unique characteristics of software synthesizers typically result in complex interfaces. It has been argued in the literature that this complexity is a beneficial attribute, to
both novice and expert users, and that it can result in increased engagement [41, 61]. Engagement is an important aspect of musical software use, as it represents the users willingness to use the program over a long period of time, in an activity that the user is choosing to do. This thesis investigated this claim by performing a usability evaluation of two commercial software synthesizers with novice users in order to find if there existed any usability issues, and how these issues affect engagement.

1.3 Coming Chapters

Chapter 2: Literature Review delves into a brief history of synthesizers, as well as an overview of software synthesizer usability, and the various factors that affect it. The literature provides a better understanding of user engagement, and how it has been measured and defined in online commerce and video game contexts. In addition, novice users are discussed as a primary focus for this research. Chapter 3: Methodology describes the protocol used to study the usability of software synthesizers, and Chapter 4: Results and Discussion provides an analysis of the results, as well as a discussion and insights gathered from the study. Finally, Chapter 5: Conclusions describes the value of this work, the limitations of the research, and a discussion of future work.
Chapter 2. Literature Review

This Chapter provides an overview of synthesizer workings, history, and current trends. An overview of the research done towards musical devices is presented, with a focus on an usability and human-computer interaction (HCI) context. The factors that affect software synthesizer usability are then developed and presented. Engagement is then defined and related to the development of usable devices.

2.1 Defining a Synthesizer

A musical synthesizer generates sound using electrical signals and may or may not provide a means for playing notes, e.g., a piano keyboard. They were developed in the late 1800s, early 1900s. Some notable examples are the Theremin, the Electronic Sackbut, and the Ondes Martenot [53]. There were even devices that experimented with sequencing, e.g., the Coupleux-Givelet Synthesizer (which, while not its official name, was the first device to be referred to as a synthesizer). Early devices were all based on tube circuitry, making them large, unwieldy, and unreliable. With the transistors becoming more and more prominent throughout the 1950s, Robert Moog and Donald Buchla independently developed the modern hardware-based synthesizers in the mid 1960s as documented in “I Dream of Wires” [30]. The Moog went on to commercial success, in no small part due to the release of “Switched-On Bach” by Wendy Carlos. Buchla’s synthesizer, the Buchla Modular Elec-
tronic Music System, did not manage much commercial success, but gained a following among academics and avant-garde musicians.

![VAZ Modular synthesizer](image)

**Figure 2.1:** The VAZ Modular synthesizer by VAZ Synths, released in 1998. [9]

The first forays into purely software synthesizers technically started in the late 1950s, when Max Mathew wrote a program that would compute a triangle wave (a triangle wave is similar to a sine wave, except it alternately ramps linearly upwards and then downwards), although it was not able to do so in real-time due to the low CPU speeds of the time [27]. In the 1990s CPU speeds started to become fast enough for home computers to run software synthesizers, and the first home software synthesizers became available. Some examples are SimSynth, Reality, and VAZ Modular (See Figures 2.1, 2.2 and 2.3 respectively).

An important advent in the history of synthesizers was Steinberg’s release of *Virtual Studio Technology (VST)* 2.0 specification [59].
VST is a software interface that integrates software audio synthesizer and effect plugins with audio editors and recording systems. VST and similar technologies use digital signal processing to simulate traditional recording studio hardware in software. Thousands of plugins exist, both commercial and free-ware, and a large number of audio applications support VST under license from its creator, Steinberg [25].

Initially developed in the mid 90s, it enabled the development of effects that could be plugged into existing digital audio workstation (DAW) software, extending a user’s options over and above the built-in functionality. The 1999 version 2.0 provided tools for generating audio, rather than just manipulating it. This resulted in an explosion among software synthesizers, with many today still running on a 2.X version of the VST spec (VST 3.5 having been released in 2013). Initially, many of these synthesizers were focused on recreating hardware synthesizers, complete with visually identical UIs. For example, Figure 2.4 and Figure 2.5 show the original hardware-based synthesizer and the software replica respectively.

2.2 Software Synthesizer Usability

This section covers previous work that attempted to evaluate any type of digital musical instruments (DMIs, a superset of software synthesizers, which also accounts for physical synthesizers and other electronic instruments). It covers work that provided frameworks for the evaluation of digital musical tools, as well as any attempts at analyzing software
Figure 2.2: Image Line SimSynth was released in 1994, making it one of the earliest software synthesizers. Image Line continues to make computer music software. [10]

Figure 2.3: Seer Systems Reality is another synthesizer made in the 1990s, this one has an interface consisting of operating provided widgets. [8]
Figure 2.4: Sequential Circuits Prophet 5, a hardware synthesizer. Compare with the following image showing the digital recreation. [6]

Figure 2.5: Native Instruments Pro-Five, the software recreation of the Prophet 5 hardware synthesizer. [5]
synthesizer UIs. This work provided scaffolding for the protocol used in this thesis, and also displays the lack of consideration software synthesizers have received in the literature.

### 2.2.1 Evaluation Frameworks

Previous studies of the UIs for DMIs have largely focussed on physical devices [38, 35, 33, 26, 29]. While physical and software synthesizers are similar in some ways, there are differences between them that make the exploration of software DMIs worth pursuing. One of the most obvious ways they differ is in how the user interacts with the instrument. With a physical device, controls are in fixed positions and they usually do not allow customization or modification. The controls cannot change location or size. Contrast this with a software UI, where controls can appear on various screens, or disappear entirely based on context. We must also consider what input device is being used, e.g., a mouse, keyboard, or some other device(s). Does the user have access to a MIDI controller, and what sort of controls does it have? If they have one, how have they mapped the software to their device? All of these change and affect how the user will experience the instrument. Software can increase functionality and complexity that would be infeasible on a physical instrument. For example, some software synthesizers allow the user to draw waveforms [13, 14].

Frameworks for evaluating the usability of DMIs generally assume that the object being analyzed is a physical synthesizer or instrument, which either stems from, or leads to, a large focus on the performer’s experience. This focus can be seen in the titles and abstracts of the various papers: Barbosa et al. “Towards an evaluation methodology for digital mu-
sic instruments considering performer’s view: a case study” [20]; Morreale et al. state that their “framework for musical interface design [is] grounded in the experience of the player” [42]; Hsu and Sosnick discuss a “testing procedure [that] covers rehearsal and performance environments” [32]. Marquez-Borbon et al. included a survey of previous methodologies for evaluating existing DMIs and found that they are regularly conducted with existing “instruments such as the Theremin, Radio Baton and Buchla Lightning”: these three instruments are all physical [38]. O’Modhrain’s framework again revolves around the idea of the DMI being used as a performance piece, i.e., from an audience perspective and not the user’s [49].

While software synthesizers can be used in a performance setting, with the appropriate controller, this thesis focuses on musicians (i.e. users) and the use of the synthesizer for creating and manipulating sounds. The software synthesizers undergoing evaluation are not designed with an audience in mind; even if it were to be manipulated in a live performance. In some respects, software synthesizers are similar in nature to traditional software applications, e.g., people are more interested in a Word document than watching someone using Microsoft Word to create the document.

This focus on performance does not mean that the frameworks cannot be used for evaluation of synthesizers. Interacting with a synthesizer is a creative endeavour, and the evaluation techniques proposed by the previous authors can be adapted to digital synthesizers. Barbosa et al. paper provided the foundation for our protocol, which is expanded upon more in Chapter 3.
2.2.2 Analysis of Interfaces

Alan Seago has done the most work in expert analysis of UIs, producing a wide-variety of technical papers on the various aspects. His work provides a basis for examining UIs as many of the controls have not changed since the papers were written in the mid-2000s [55, 58, 54, 57]. Seago also performed his own user testing, and this also informed the protocol used in this thesis. For an in-depth analysis, please read Chapter 3.

2.3 Factors Affecting Software Synthesizer Usability

Here we identify aspects of synthesizers that can affect usability, based on prior research as well as our own evaluations.

2.3.1 UI Complexity and Feedback

Modern synthesizers can easily have 30+ parameters, which can result in millions of different combinations [56]. Thus, understanding how changing a single parameter will affect the output can be difficult in the best circumstances, and sometimes impossible. Understanding cause and effect relates to the number of controls on a typical synthesizer, but is not related to their control or use. Instead, it is related to what these controls contribute to the eventual audio output. This is not always undesirable, as unexpected changes have been shown to enhance creativity and enjoyment of using a synthesizer [33]. As seen in Figure 2.6. this synthesizer has a large number of controls, and the designer intends for you to not
manipulate them manually, as they repeatedly state that it is meant to have its configuration randomized.

![Figure 2.6: iraisynn.attinom Araucann Synthesizer, which has an overwhelming amount of controllable options. [1]](image)

### 2.3.2 Prerequisite Knowledge

In addition to the complexity of the UI, users must also know how these controls can be used. This is where prerequisite knowledge comes into effect. Software synthesizers make use of specialized terms and abbreviations that are common in the field, but may be confusing to novice users. If a developer decides to use differing terms, it may also make it
more difficult for expert users to get situated. The type of synthesizer can also play a role, with different models of synthesis requiring different conceptual models. Without these conceptual models, it can affect the learning experience, and potentially cause users to not enjoy using the tool [61, 41].

To illustrate a model’s complexity the following description of a theoretical subtractive synthesizer was developed by myself. The three main components of a subtractive synthesizer are the oscillator, the filter, and the Attack Decay Sustain Release (ADSR) envelope. In a subtractive synthesizer, the oscillator is used to generate an initial harmonically rich waveform. This waveform is then passed through a filter, which will attenuate some harmonics, while leaving some unattenuated (thus the name subtractive synthesis, as we are selectively subtracting harmonics from our initial sound). This modified waveform is then passed to the amplifier, whose output will be modified by an ADSR envelope. The ADSR envelope represents a transformation of the volume over time, after the note is inputted: attack is the time taken from zero volume to a peak volume; decay is the following run down from peak to the; sustain level, the volume kept until the input is released; finally release, which is the time from input release to zero volume. Finally, the sound is passed through any effects the synthesizer may have before finally being heard.

### 2.3.3 Software Synthesizer Usability for Novice Users

At its most basic, sound synthesis is the representation and manipulation of sound, an essentially non-visual attribute. A picture of a waveform can be created, though Seago
notes that there is a “lack of any human understandable mapping between the subjective and perceptual characteristics of the sound in any detail and its visual representation on screen”.

An effective synthesizer must reflect the relationship between mathematics and sound in a way that is useful and meaningful to the user. The complexity of the mathematics does not mean that the tools must be confusing, or that confusion is inevitable. As Don Norman states, “confusion, not complexity, is the problem” [45]. If the synthesizer is confusing, this can hinder its adoption or use. It can cause people to give up on it entirely, either quitting the field entirely or moving to a competing product. Care should be taken in order to minimize confusion and thus promote greater engagement with users.

The primary research question (RQ) in this thesis is:

**RQ1. How usable are software synthesizers for novice users?**

### 2.3.4 Non-Standard Controls

Software synthesizers routinely use custom made interfaces and widgets for their interfaces, as can be seen in Figures 2.6 and 2.7. While a standard practice in the synthesizer industry, these interfaces do not align with any desktop UI guidelines. This can cause confusion among the uninitiated. Synthesizers make frequent use of the “knob widget”, a circular widget used much like a slider, and can be seen in Figure 2.8. These non-standard controls are not a large issue, because they remain consistent between synthesizers. If a user gains experience with the controls of one synthesizer, they will know how to use the majority of other synthesizers controls, with only some minor differences between them such as labels,
colouring and the overall layout of the UI.

Figure 2.7: The HY-MONO synthesizer, which makes no use of standard operating system widgets [34].

2.3.5 Visual Design of the UI

The visual design for many UIs focusses primarily on aesthetics. Many use a single colour scheme for their entire UI, eschewing the effects that colour can bring to the user experience. Colour can be used incorrectly by drawing attention to unimportant parts of the UI. For example, in Figure 2.9, the yellow widgets at the bottom-right corner of the screen seem important as the yellow covers a large portion of the screen real estate and draws the user’s eye. However, this section is not as important as its colour would suggest. It is used
Figure 2.8: An example of the “knob” controls, this on from the Massive synthesizer made by Native Instruments [43]. To change the value, a user must click on the widget and drag their mouse up and down to modify the value.

... to assign controls to an external MIDI controller, allowing users who possess such a tool to control certain parameters with their device. Users who do not have such a device do not benefit from, or use that section of the UI at all. The visual design of software synthesizer UIs can be examined from a number of aspects:
Figure 2.9: The entire UI of Massive, note the yellow controls in the bottom right of the interface.
**Skeuomorphism** In software design, skeuomorphism is a design principle where UIs were created to mimic their real-life counterparts, *e.g.* a contact application that mimicked the look of a paper contact book. The use of skeuomorphism in digital audio software is somewhat notorious. A common example cited is Propellorhead’s Reason software [52], which attempts to recreate an entire “rack” of hardware, complete with front and back panels, with wires that connect the “devices” (see Figure 2.10 and 2.11). Many software synthesizers and effects take the same approach, attempting to emulate exactly the devices that they are based on (see Figures 2.12 and 2.13). It has also been discussed in the literature by Marrington and Bell, Hein and Ratcliffe; as well as the news media [39, 22, 12].

**Modality vs Maximizing** An issue that may stem from the focus on skeuomorphism is an effort to place as much of the UI on one screen as possible. Physical devices have limited ability to change their interface based on context, which means a majority of the features must have physical controls. Of course, this is not uniform among digital synthesizers. For example, the Zebra 2 synthesizer [60] makes extensive use of modality, with 10 tabs at the bottom of the UI separating parameters, as well as having controls for the synthesizer modules appear and disappear depending on if they are being used (see Figure 2.14). However, the amount of information on display, even while using this modality, is still quite high.
Figure 2.10: Propellorhead’s Reason - the “front” of the synthesizers.
Figure 2.11: Propellerhead’s Reason Software - the “back” of the synthesizers.
Figure 2.12: The Mini V, created by Arturia to mimic the sound and UI of the hardware based Minimoog. [4]

Figure 2.13: The Farfisa V, also created by Arturia, to mimic the Farfisa organs from the 1960s. [2]
Use of Colour  Digital synthesizer UIs seem to use colour mainly for aesthetic reasons. Colour was not often seen being used for a usability purpose, for example, grouping widgets that belong to the same module, or differentiating controls based on their function.

![Figure 2.14: Zebra 2, the tabs are circled in red.](image)

Labels and Controls  As mentioned in the previous section, synthesizers make frequent use of non-standard controls. These knobs and sliders, again, are very skeuomorphic in nature, as it cannot be argued that manipulating a knob with a mouse is natural. La-
belts are used very extensively. However, often the labels are shorthand or acronyms (see Figure 2.15). While good for expert users, as they take up less space, they can hinder novice users who do not know the meanings. They can also hinder expert users if they have never seen a specific term or abbreviation before.

![Figure 2.15: An example of the labels used by the synthesizer Dexed [50].](image)

2.3.6 Input Methods

Using a mouse and keyboard can limit the creative output of one using a software synthesizer. This is because it can be hard to manipulate the controls in real-time. Manipulating the sound in real-time is important because it allows the user to comprehend how changes to controls will sound while the instrument is playing. To use an analog example, one can play a note on a trombone with the slide at one position, stop, move the slide and play another note, but that would not allow them to learn the distinctive trombone slide. This issue is exacerbated with a synthesizer because the number of controls is much greater than
2.4 Engagement

2.4.1 Defining Engagement

To engage someone, as defined by the Oxford English dictionary [11], is to “to attract and hold fast (attention, interest)”. This ambiguity in the term has led to many different usages among the HCI community. Peters, Castellano, and de Freitas note that there is a “great variability, overlap and [...] vagueness” in defining engagement [51]. Indeed, they note that engagement had been referred to as “... a process; a stage in a process, or the overall process; as an experience; as a cognitive state of mind; an empathic connection; as a perceived or theorised indicator describing an overall state of the interaction.” O’Brien postulates that “[e]ngagement is a quality of user experiences with technology that is characterized by challenge, aesthetic and sensory appeal, feedback, novelty, interactivity, perceived control and time, awareness, motivation, interest, and affect [47].” O’Brien and Peters et al. in their attempts to define engagement, do not refer to one single aspect of interaction, rather they encompass a wide array of concepts and phases. O’Brien and Toms conducted interviews with users of four different applications in order to determine the differences in engagement between the different areas [48]. They discovered four theoretical frameworks for analyzing user experiences in HCI: flow theory, aesthetic theory, information interaction, and play theory. Attributes drawn from these theories were used to identify attributes of
engagement. Their goal was to determine the viability of the indicators as quantifiers of engagement. Their initial attributes were: aesthetic and sensory appeal, attention, challenge, endurability, feedback, interactivity, positive effect, and perceived user control. From these interviews, O’Brien and Toms developed a comprehensive model of engagement. This model identifies four stages of engagement: point of engagement, period of engagement, disengagement, and re-engagement. These periods all have specific attributes associated with them, which results in many more attributes than initially proposed. These final attributes are:

1. Interest
2. Motivation
3. Attention
4. Challenge
5. Feedback
6. Aesthetics and Sensory Appeal
7. Awareness
8. Novelty
9. Perceived Control
10. Perceived Time
11. Interactivity

Similar to the work done by O’Brien, Wallis, Ingalls, Campana and Vuong developed seven factors they believe encourage long-term engagement [61]. Long-term engagement can be thought of as years of tool use (in our case, a DMI) without recompense, while deriving enjoyment from the activity. The authors looked at the factors of instruments that
inspire long-term engagement in their users, and identify attributes that can be applied in an HCI context. They developed seven attributes that contribute to long-term engagement, and further subdivided them into three super-groups, called motives. The motives and attributes are:

1. Mastery (incrementality, complexity, immediacy)
2. Autonomy (ownership, operational freedom);
3. Purpose (demonstrability, cooperation)

There is overlap in the attributes proposed by Wallis and O’Brien, e.g., interest, motivation = ownership; perceived control/time, aesthetics and sensory appeal = operational freedom.

### 2.4.2 Types of Engagement

There are two conflicting thoughts surrounding engagement: engaging for novice users versus engaging for mastery [41]. Engaging novice users is important because it can increase growth and participation. Users act as ambassadors for products they use and like, promoting them on social media platforms, creating and sharing learning material, and other such benefits. By engaging new users with an accessible UI design, users are more likely to stick with and promote a product in a way that a more immediately complex product will not. “In a world full of choice where the fleeting attention of the user becomes a prime resource,
it is essential to design engaging experiences.” [37] At the same time, products must remain engaging as users continue to develop skills while using a synthesizer. Too simple, and users grow bored and feel hamstrung; too complex, and they feel frustrated that their effort will not result in appreciable results [41]. Maintaining this balance is a fundamental design struggle. Functionality and mechanisms for using that functionality must be made available to the user as they move from novice to expert, but must be done in a way that does not hamper the initial engagement of the new user.

2.4.3 Comparison to Video Games

O’Modhrain made the observation by noting the similarity between DMIs and video games, although still in the context of a performance, stating that “HCI research on the evaluation of interactive game playability is relevant in a discussion of the experience playing DMIs” [49]. Video games and digital musical instruments have many similarities, yet are also different. Barr, Noble and Biddle note that “video games are not made to support external, user-defined tasks, but instead define their own activities for players to engage in” [21]. On the face of it, DMIs are not similar: they are explicitly used to support user-defined tasks, and they do not present their own activities for musicians to engage in. However, DMIs, like video games, are not productivity applications - they do not benefit from, or at least benefit much less from, a productivity-centered evaluation. Instead, like games, they “focus on the process of use [...] rather than the results of that process” [21]. In other words, the journey is as (or even more) important than the destination.
With this in mind, one may wonder where synthesizers fall. They are entirely DMIs, lacking the focus on performance aspects traditionally associated with DMIs, and they are not entirely productivity based software, although they lend themselves towards more task-based evaluations than video games and DMIs. And they are definitely not video games, lacking explicit activities to engage users. They seem to lie at an intersection of all three of these classifications.

2.4.4 Measuring Engagement

Knowing that both long-term and short-term engagement are processes, how can we go about measuring engagement? Wallis et al. note that there are no optimal methods to measure long-term engagement, they provide notes on what they consider best practices for its study [61]. These include: participant recruitment based on interest; no payments or rewards, because engagement should come intrinsically, and compensation provides an extrinsic motivator; accurate time tracking of participants; and an evaluation based on the attributes described previously (See Section 2.4.1). O’Brien, with the help of other researchers, discuss how to measure engagement in multiple papers [46, 47, 48]. Their focus revolves around the development of a comprehensive survey for measuring user engagement in e-commerce environments [48]. Again, their results suggested using questions related to their proposed attributes. Although not directly related to the topic of digital music, their research was valuable for developing questions for the protocol used in this thesis. Carol, Latulipe, Fung and Terry also developed a tool for measuring creativity in a research
setting. Based on NASA’s Task-Load Index (TLX) [31] they identified six functions that they believe promote creativity the most, and rank them on a scale of 1-10. These are:

1. Results Worth Effort

2. Expressiveness

3. Exploration

4. Immersion

5. Enjoyment

6. Collaboration

As well as providing questions to measure these aspects, they also had subjects rank each metric against each other in order to decrease the weightings for aspects subjects did not think helped the creative experience. These scores and weightings were then used to create a final score from 1-100. This focus on the final score is misplaced - for researchers, it would seem that being able to compare different aspects of the interface would be more useful than a single number score. Additionally, the inclusion of collaboration is dubious; not all applications have support for collaboration, and it also caused confusion among test subjects who asked if the section should be ignored, or it was relevant. The users also gave it seemingly random scores. Despite these shortcomings, the work played a role in the development of questions for this thesis and is expanded upon in Chapter 3.
Boberg, Karapanos, Holopainen and Lucero developed a system for measuring playfulness of interactive digital products [23]. Called the PLEXQ Questionnaire, they found 17 aspects of playfulness, with questions to measure these different aspects. While some of these aspects can be applied to multiple domains, many are quite specific to video games. However, some of the questions could be used in a musical context. Given their small number, and the author’s reliance on the final combination of these aspects, their questions were not used in this experiment.

This brings up the final RQ to be investigated in this thesis:

**RQ2. Do software synthesizers engage novice users?**

### 2.5 Thesis Statement

This thesis examines the usability of software synthesizers with a focus on novice users. The following research questions were developed in this chapter and will be used in Chapter 4 to organize the results and discussion:

**RQ1.** How usable are software synthesizers for novice users?

**RQ2.** Do software synthesizers engage novice users?

Although it has been argued that the complexity of synthesizers is inherent to their design, and that this complexity is a benefit to engagement, a synthesizer which attempts to address the usability issues found above will be more immediately usable for novice users, and encourage more engagement than one that does not address those issues.
Chapter 3. Methodology

To answer the questions posed by the thesis statement, a protocol for comparing two software synthesizers was created and carried out. The choice of synthesizers is discussed in this chapter, as are the participant demographics, equipment used, and the protocol.

3.1 Two Synthesizers for Comparison

3.1.1 Synthesizer 1: Massive

Massive is a product developed by Native Instruments [43]. It was released in 2007 and continues to be one of the most popular synthesizers on the market, placing highly on the Splice list of most popular synthesizers [7], as well as being rated 7th in Music Radar’s list of best software synthesizers in the world [18]. Reviews of Massive indicate that it was meant to have an “understandable user interface” [16] and “a deceptively simple user interface” [17]; its own product description states that the interface is “[c]learly laid out and easy to use” and that its “features have been implemented with a focus on usability” [15]. Massive is a synthesizer that combines many types of synthesis techniques: wavetable synthesis, subtractive synthesis, frequency modulation and ring modulation, among others. Massive also comes with over 1300 presets. Presets are saved configurations of the musical synthesizer that are included with the product. These presets are usually developed by pro-
fessional musicians, and are included to provide the buyer with an idea of what is possible with the program.

Figure 3.1: Massive’s user interface when initially loaded

The Massive interface presents 37 knobs, 10 sliders, 15 dropdown menus, and over 40 buttons when first loaded (see Figure 3.1). These widgets are grouped into various modules. For example, the OSC1 (Oscillator 1) Module (see Figure 3.7) has three knobs (Wt-position), one slider (the right-most widget), three buttons (blue circle to the left of OSC1, two arrows to the right of Squ-Sw1), two dropdown menus (Squ-Sw1 and Spectrum
widgets), and the pitch control, which can be modified by sliding the cursor up and down after clicking and holding on the widget, or by clicking on the arrows above and below the numbers.

The tabs in module in the centre of the interface (OSC, KTR OSC, KTR FLT, etc.) are configuration panels, and contain options that affect how the synthesizer operates. This area is discussed in-depth in Section 3.1.3. The 4 blue tabs each contain an ADSR envelope, and can be assigned to almost any parameter by clicking and dragging from the 4-way arrow icon beside its label, and then dropped into the empty box of the parameter you wish to have it affect.

**Justification for Inclusion**

Massive is a very popular synthesizer in the software synthesizer space, and reflects many of the trends present in this field. As well, it is said to be usable by third-party sources, and is recommended as a software synthesizer suitable for beginners. For these reasons I believe it is a suitable candidate for use in the protocol.

### 3.1.2 Synthesizer 2: Rounds

Rounds was the second software synthesizer chosen, and it is also developed by Native Instruments [52]. Despite being received positively by the industry press, Rounds is less known. Rounds is described as “deceptively simple, but after spending some time with it you will see that this synth is much deeper than meets the eye.” [19] Rounds comes with a
comparatively low 233 presets.

Figure 3.2: Rounds’ user interface when initially loaded

The Rounds interface initially presents 26 knobs, and many buttons (see Figure 3.2). The pitch control works similarly to Massive’s, but does not have the arrow buttons. The interface is broadly divided into two parts: the upper half, which controls how sounds are played when the user plays a note; the bottom half holds all parameters which affect the two synthesis engines.

The bottom half is used to organize the controls used by Rounds’ synthesizer engines. Unlike Massive, Rounds has icons which display the rough shape of the oscillators, filters,
and LFO (low-frequency oscillator) shapes. Rounds is more restrictive in how its parameters can be modulated. Where Massive lets you use one modulator on multiple parameters, Rounds allows only one parameter, specified using the buttons on the right-hand side of the Mod module. However, Rounds allows the use of many more layers of configurations.

Near the middle of the screen are the numbers 1-8 twice, one in purple and one in green. The purple and green represent the synthesizer engine, and the numbers represent distinct configurations of each. These configurations can then be layered using the eight circular widgets above, and there are options which control how the synthesizer moves through these widgets, and how notes are redirected to them. In Rounds, when a parameter in the top section is disabled, it is dimmed (see Figure 3.3).

![Figure 3.3: How Rounds indicates that a module is disabled (left) versus enabled (right)](image)

**Justification for Inclusion**

Unlike Massive, Rounds did not reach a high level of popularity. However, it takes a different approach from Massive in its design and layout. These differences, detailed in
the following section, allow us to evaluate many different factors and how they affect the usability of the synthesizer.

3.1.3 Critical Differences

**Modality vs Maximalty** The contrast of modality and maximality was noticed during the evaluation of these two interfaces. While both attempt to present a large amount of functionality to their users at one glance, Massive takes it further. This can be seen in the number of controls immediately present on each interface. One of the main interactive widgets used by both synthesizers are knobs; Massive has 37 knobs present when the interface is first loaded, compared to Rounds’ 25. Furthermore, functionality that Massive has present by default is hidden by Rounds - namely, functions related to MIDI controller support, and audio effects controls. And while both interfaces use modality, that is presenting information and controls only behind specific panels of the interface, Rounds uses it much more. In addition to the controls mentioned previously, each of Rounds’ 16 synthesizer configurations are modal; each reconfigures the controls are the bottom when the mode is selected. Massive also uses modality to present in-depth configuration menus, and confusingly, the ADSR envelopes are behind modal windows. Not only is this critical functionality not immediately present, it is hidden behind a non-standard access method. This can be seen in Figure 3.4, the area of interest circled in red. Not only are these tabs placed beneath another row of tabs, they are not in the same style as the tabs above them.
Figure 3.4: Massive’s ADSR Envelope Access Widget, highlighted in red

Use of Colour  Both interfaces attempt to make strategic use of colour, with differing results. Both use colour to highlight important features of their use. Rounds use of colour is more effective from an usability standpoint. Rounds main use of colour is to distinguish between its two synthesizer engines, where pink is for its Analogue engine, and cyan for its Digital engine. When editing a synthesizer configuration, the important controls are coloured in the engine’s colour. When configuring the sequencer, the colours are used to display which engine is being used. Furthermore, Rounds also uses shades of grey to distinguish controls that relate to different functionality. Looking at Figure 3.5, one can see how this is put into practice in the Output controls.
Figure 3.5: Round’s Output Control Area, knobs with different functionality are highlighted with different colours

Massive also uses colour to highlight some functionality, but it is reserved for only three features. Referring back to Figure 3.4, one can see that overall Massive is very monotone, using only shades of blue and grey, outside of the neon green, blue, and yellow that are used quite sparingly. The neon green and blue are used to bring attention to Massive’s modulation system. A dull blue colour is also used to indicate if a component is active or inactive. The yellow is used to indicate which parameters are able to be controlled by an external MIDI controller.

**Control Size** The last main difference between the two interfaces is the use of size in of their controls. Massive does not use size to differentiate between importance of parameters. That is, each parameter, no matter it’s importance, always has the same size widget. This is in contrast to Rounds, which extensively uses size to highlight
importance in functionality. Below, you’ll find Figures 3.7 and 3.6, where one can see how this is put into practice. Here one can see how in Rounds, the most important control is coloured and larger. By comparison, Massive, has fewer controls of a similar size and colouring.

![Figure 3.6: Round’s Oscillator Control Area](image)

**Preset Access** Massive has a much more in-depth system for browsing and experimenting with presets. Massive’s provides an entire preset browsing interface, allowing the user to browse through presets by tags. These tags include moods and feelings, genres, emulated instruments, and other various properties of the final sound. Multi-

![Figure 3.7: Massive’s Oscillator Control Area](image)
ple tags can be selected to further narrow down the choice. Rounds, in contrast, has a very spartan browsing tool. It is a list, that is grouped into various categories or folders that are only ever one level deep. In both programs the actual names of the presets hardly ever give an indication of what they will sound like. However, Massive counteracts this by providing tags. Rounds has only the folder names.

3.2 Demographics

10 participants took part in the evaluation protocol. Eight of the participants were undergraduate students, and two were graduate students. All participants attended the University of Guelph. Participants were between the ages of 18 and 28. Five participants identified as male, and the remaining five as female. Participants received no compensation for participating in this experiment. None of the participants had majored in a music related field. Five participants had experience recording music on their computer and eight participants had heard of digital musical synthesizers, but none of them had any experience using a digital musical synthesizer. One participant did have experience using a hardware musical synthesizer. All of the participants who started the study completed it, however, two of the participants did terminate a playing session early.
3.3 Equipment

The software for the experiment was run on a 2013 MacBook Pro, running OS X El Capitan. The computer was hooked up to a 24” widescreen monitor, running at a resolution of 1650x1080. The two software synthesizers were run through Ableton Live. The users were able to interact with the synthesizer via mouse, keyboard, and AKAI LPK25 MIDI controller (see Figure 3.8). Users were presented with only the synthesizer, which was situated in the middle of the external monitor. Ableton Live was minimized to prevent the user from interacting with the DAW, or otherwise becoming distracted by the additional interface.

Figure 3.8: AKAI LPK 25 MIDI Controller
3.4 Procedure

When participants first arrived at the experiment location, they were asked to read and sign an informed consent document, detailing the rights of the participant with regard to the study. The researcher took this time to remind the participants of their right to withdraw from the study at any time. After signing and agreeing to participate, they were asked to complete a pre-experiment survey (see Appendix 5.3, and reference the Research Ethics Board case 16FE022).

Before showing the user the interface, they were given a brief overview of the basic controls and signal flow of a theoretical typical synthesizer. This description centred on what I felt were the basic essentials for learning and using a synthesizer. This included: oscillators, filters, the amplifier, and ASDR controls. We wanted to have the user “use” the interface, and not become overwhelmed by the terminology and complexity of the task, or at least not so much that they could not achieve any output. The graphic can be seen in Figure 3.9, and the description given mirrors the one discussed in Chapter 2.

Papers by both Hsu and Barbarosa were used in the development of the experiment [32, 20]. Both papers discuss how a rehearsal approach, that is, having the participants use the synthesizers freely, without tasks, is beneficial because it emulates how the synthesizers will be used in reality. Barbosa also discusses how both guided-exploration and unguided-exploration are beneficial to uncovering users thoughts and feelings towards the interface. Guided versus unguided exploration refers to the amount of help the participant is able
Figure 3.9: The diagram used to explain the basic components of a synthesizer

to request. In unguided exploration, the participant is unable to ask for any help with regards to the interface, while guided-exploration allows them to ask questions and be answered. Comparing two synthesizers was based off of the work Seago doing a similar test on hardware synthesizers [54].

Thus, participants were instructed to explore the interface without any specific tasks. Two 15-minute rehearsal sessions were used, one consisting of unguided-exploration and the other guided-exploration.

In order to explore user feelings towards both of these activities, the participant was asked to first explore the interface, and explore the sound design tools that their respective synthesizer offered. After having used the musical synthesizer for 15 minutes they were required to fill out a two-page survey, which is discussed in the following paragraph, before being allowed to move on to the second rehearsal. During the second rehearsal, the participant was asked to play along to a 16 bar drum pattern, again until 15 minutes had passed. The 16 bar drum pattern was included as an attempt to see if it would affect the engagement
of the participants. Would a backing instrumental lead them to treat the interface more as an instrument, or would they instead continue to only browse and explore the interface?

After completing these activities, the participant was presented with a two-page survey asking them to describe their impressions of the software. The first page had questions based on the work done by O’Brien, who developed general engagement measuring questions [48]. In her research, these questions were very numerous, being used to measure the engagement process for an e-commerce platform. These were cut down from 33 questions to 13, because many of the questions developed centred specifically on e-commerce, and the specific platform being examined, as well as containing questions that were not relevant to a digital musical synthesizer. The second page had questions developed by Carroll, Latulipe, Fung, Terry and Cheriton, and these are discussed more in the next section [24]. This time was also used by the experimenter to engage in a brief discussion of their answers, as well as allowing them to ask questions.

The Creativity Score Index (CSI) was developed by Carroll et al. in-order to develop a better measure for the perceived helpfulness of programs in a creative environment [24]. I decided to use the questions they developed in this experiment, because their approach of developing a scale based on measuring creativity and engagement is novel, and highly related to the goals of this thesis. However, I report on the results differently. The authors developed a system for combining the results into a singular score. I did not include this combined score because I was interested in comparing the different aspects of the synthesizers against each other, something that is not possible when only an aggregated score is
presented. Additionally, their scale between 1-10 was reduced to a scale of 1-4, in order to maintain consistency with the first set of questions, and because it was thought that textual labels would help the participants articulate their thoughts more, versus a plain number line [28]. The Collaborative question of the score was also omitted: the authors state in their paper that it can cause confusion among participants when the program/activity is not collaborative in nature, and can safely be removed because I did not calculate the final CSI value.
Chapter 4. Results and Discussion

Applying the protocol described in Chapter 3 generated data and observations about the usability of synthesizers and the level of engagement of novice users, i.e.,

RQ1. How usable are software synthesizers for novice users?

RQ2. Do software synthesizers engage novice users?

Section 4.1 will examine the usability based on how well the synthesizers’ visual and conceptual design informed participants. Section 4.2 will examine whether the synthesizers were able to engage the participants, and how the usability factors affected their engagement.

4.1 Usability Issues for Novices

Based on the observations gathered during the participants usage of the synthesizers, multiple usability issues were found and are organized around the the following themes:

1. Visual Design & First Impressions

2. Non-Standard Controls

3. Building a Mental Model

4. Understanding Current State
5. Functionality Awareness

6. Action Feedback

7. Importance of Presets

4.1.1 Visual Design & First Impressions

Two of five participants found the Rounds (R) interface “very aesthetically pleasing”. None found Massive (M) pleasing. In contrast, 8/10 participants (5/5 (M); 3/5 (R)) made statements regarding the apparent complexity or confusing nature of the UIs when first introduced to them, with comments like “It is complicated. Where do I start” (M), “little confused, little overwhelmed” (R), and “too many knobs on this thing” (M).

Massive has 28 identically-sized knob widgets visible when the interface is first loaded. Participants wondered why all the knobs were the same size, and why knobs not in use (marked with an “n.a.”, see Figure 4.1) were the same colour and able to be moved. Participants (M) noted that due to the above features, they had trouble determining where to start, and how to determine what was an important part of the interface. Comments concerning not knowing where to start were not heard for Rounds, which itself has 25 knob widgets visible when first loaded. This can be attributed to the layout of the Rounds interface, and how it distinguishes functionality. It has an obvious area of control, the bottom half of the interface. This area contains visually distinctive controls, and groupings with labels that they would have learned in their introductory explanation of musical synthesiz-
ers. Furthermore, this area is organized such that the control groupings mirror the signal path. These attributes build a solid, visible foundation of functionality, that the participants gravitated to, see Figure 4.2. This is supported by the survey data collected: Massive users experienced the bigger jump in understanding when they were able to receive help from the researcher. When analyzing question "Do you understand how the synthesizer functions?", three of the participants move their answer from not understanding to somewhat or definitely understanding. Comparatively, none of the Round’s participants felt that they definitely understood the interface, and only one answer changed from not understanding to somewhat understanding. This shows that the Rounds participants had a better grasp of the interface before being allowed to ask questions.

![Figure 4.1: Massive’s use of “n.a.” to identify widgets with no functionality](image-url)
4.1.2 Non-Standard Controls

Both Massive and Rounds had issues with the controls. Participants for both UIs leaned toward rating the controls hard to understand, with one Massive participant and two Rounds participants feeling they were somewhat easy to understand.

**Rotary Controls (knobs)** Like hardware synthesizers, knobs (rotary controls) are a primary means of interacting with a software synthesizer. Not wanting the operation of rotary controls to unduly restrict participants’ usage of the synthesizers, correct usage of the knob widget was demonstrated prior to the evaluations. However, problems operating the knobs were present during the evaluations. One participant continually attempted to drag the mouse in an arc to interact with the widget. Other participants would simply forget the direction of control for the widget, *i.e.* they would incorrectly drag the widget up in order to lower the parameter. Although this issue was present at the beginning of the evaluation, fewer mistakes were made as time went on. This did not seem to impact the evaluation, as it never amounted to more than a minor annoyance. Once the participants were comfortable with the control, it quickly
faded into the background.

**Non-Delinated Buttons** The Rounds UI frequently uses buttons without borders (see Figure 4.3). These controls were frequently ignored (5/5). Participants indicated that they thought the buttons were static labels rather than interactive controls. Without differentiation, it can be difficult for a user to realize that they are missing out on functionality provided by the musical synthesizer.

![Figure 4.3: The non-delineated buttons present in the Rounds interface. The text circled in red are the only areas present which CANNOT be clicked on](image)

**Signalling Functionality** Both synthesizers lack visual feedback typically used to indicate functionality. For example, neither indicate that a control’s functionality is available when the cursor is placed over the widget, e.g., by changing the look of the cursor and/or widget when the cursor is placed over the widget. This is important because it is different from much of the software in use today, where either the cursor will change, or the widget will react in some way. The software, by not incorporating them, is pushing against the subconscious expectations of the user. It also hinders
the user, because unless the user randomly attempts to use a control, or looks at doc-
umentation outside of the program itself, they will never learn that the functionality
the control exposes exists.

**Self Documentation/Explanation** One participant expressed a desire for the program to
have tooltips, so that they could get an idea of what each control was meant to do.
By not providing tooltips, the developers missed an opportunity to provide a tool for
understanding the acronyms (this is expanded on in the following section) without
having to exit the program. Self-documentation would also allow explanations for
concepts and features inside the application, allowing the user to teach themselves
while remaining engaged with the program.

**Terminology Used in UI** One participant noted that they did not understand the acronyms
present in the UI. When prompted by the evaluator, all users had difficulty explain-
ing terms and symbols. Difficulty explaining these features indicate an area of their
mental model which has not developed fully. Below are two notable areas that par-
ticipants had difficulty explaining:

- (R) Participants were unsure of the term APM ENV and ADSR labels above
  the knobs in the Output module. These controls are for the synthesizers enve-
  lope functionality, and is concerning because the ADSR envelope was a part
  of their introductory explanation. By being unable to explain this control, the
  participant was missing a large portion of functionality.
• (M) The OSC modules of the interfaces had trouble being explained. Although all participants identified that it was an oscillator module, many (4/5) were unable to identify the functionality described in the introductory explanation (waveform type), nor were they able to describe the controls present in the module (Wt-Position, Intensity or Amp). For waveform type, this may be because the name of the waveform is condensed significantly from its full name. For example “Square-Saw I” becomes “Sq-SawI”. When side by side the change seems obvious, however when placed in a busy interface, may be lost.

Many others areas could not be explained. For example, in the Oscillator module on Rounds, none of the participants were able to explain what the X-MOD, PW or FINE knobs did. However, this is relatively minor functionality, and would not affect usage in the same way as those mentioned previously.

4.1.3 Building a Mental Model

When asked, no participant developed, nor believed they developed, an accurate understanding of the UIs after phase one of the experiment. For example, participants were unable to explain how the signal flows, and could not map the example signal flow and controls to the actual controls present on the synthesizer. Understanding how the signal flows through the synthesizer is essential to using each of these devices. For all participants, there was a disconnect between the functionality, layout of the interface, and signal flow. For example, one participant incorrectly mapped the signal path to the UI (M. After
session 1, they believed that Filter 1 and Filter 2 were directly after Oscillator 1 and Oscillator 2 in the signal path, and that Oscillator 3 did not have a filter (this and the actual path can be found in Figures 4.4 and 4.5).

Figure 4.4: The mental model that the participant developed during usage

To aid the user, Massive comes with a figure displaying the signal path in one of its tabs (see Figure 4.6). The one participant who discovered this feature was unable to make use of it. They spent some time looking at it, and thought it was how the synthesizer was organized, but was unable to map the controls onto the diagram presented. They were confused by the buttons present on the interface, wondering if they were changing some of the functionality, but ultimately concluded that they were having no effect. Some of the controls were having an effect, although it may have been marginal based on the current
Figure 4.5: A depiction of the actual model for oscillator and filter signal flow

configuration. However, one of the controls definitely had no effect, because it was a control labelled “n.a.”, which is the label massive uses when it disables a control.

Massive participants initially had a harder time building a mental model: after their first usage, all of the participants felt that they did not have a good understanding of how the synthesizer functioned. Comparatively, Rounds had two participants who felt that they had a somewhat good understanding of how it functioned. However, when asked again at the end of the experiment, Massive participants had a large surge, with one participant stating that they were very confident in how the synthesizer functioned, and two feeling somewhat confident. Rounds participants did not experience this, with only one user moving to somewhat confident. This is likely due to functionality awareness, which is discussed
4.1.4 Understanding Current State

The inability for the participants to map their understanding onto the UI is a failure for both synthesizers. Without a correct mental model, users cannot make correct assumptions about cause-and-effect, and feel that they have little control over the actual output of the device. Through a review of the participants’ interactions with the synthesizer, three aspects of the UIs affected the development of participants’ mental models. State is the current configuration of a synthesizer parameters, which control the signal flow through the system. The large number of interacting parameters present in synthesizers results in a
very large number of states. Therefore, managing and displaying this state is an important aspect of musical synthesizer UI design. Both Massive and Rounds had issues communicating this to the participants. Lack of visual feedback reflecting the current state of the synthesizer was a continual complaint among participants. As a result, participants were not sure what parameters were changing over time, or how the controls that modulate those changes were functioning. This led to participants developing an incomplete mental model, often misunderstanding why a sound was changing when they thought that they had configured it to remain fixed over time, e.g., one participant said the following “This feature I don’t understand did something crazy and I don’t know why.” Participants asked why the controls were all the same size, and why controls that were currently not in use (marked with an “n.a.”, see Figure 4.1) were still changeable. Participants noted that due to the above features, they had trouble determining where to start, and how to determine what was an important part of the interface.

Rounds participants had more trouble with how the synthesizer communicated its multiple sound layers. Each layer has its own mode and accompanying display, and participants did not realize this. Thus, they would make changes and hear only minor changes as they were in the wrong mode. This led to much confusion among participants. Even after the second trial, where the layers were explained to them, participants would sometimes forget which layer they were modifying, or begin to modify a layer that was not part of the final sound. The UI was made in such a way that the mode was not obvious for novice users. For both synthesizers, the lack of visual feedback was a continual complaint among par-
participants. Participants were not sure what parameters were changing over time, or how the controls that modulate those changes were functioning. This lead to users developing an incomplete mental model, often misunderstanding why a sound was changing when they thought that they had configured it to remain fixed over time. For novice users, understanding cause-and-effect is poorly supported by these UIs. This failing resulted in participants being unable to understand the impact of their changes, a critical part in learning how to use the devices and developing a useful, robust mental model.

### 4.1.5 Functionality Awareness

When using a new tool, the amount a user knows can be divided into three categories [44]. There is known functionality, things that a user knows the program can do. There is known inability, things that a user knows the program cannot do. And then there is unknown functionality, things that the program can do that is unknown by the user. In the case of novice users trying to use a synthesizer, their inexperience about what is possible is compounded by the synthesizer’s large number of interconnected components.

Awareness of functionality was visible in many of the interactions participants had with both UIs. In the case of Massive, it was the inability for the participants to use the ADSR envelope, as it was hidden. In the case of Rounds, some of the participants ignored the top half of the screen, cutting themselves off from a large portion of the functionality. This idea was briefly touched upon in Section 4.1.2, when users could not map the controls to the functionality each provided, demonstrating a lack of knowledge regarding the controls and
functionality possible. When asked specifically about various controls users were unable to detail the exact function for many of the controls.

**Failure to explore**

In one circumstance, a participant using Rounds admitted they did not explore or use the top 50% of the interface (see Figure 4.7). When asked why they had done this, they explained they hadn’t realized that the top part was able to be interacted with.

![Figure 4.7: The area highlighted in red is the area that participants glossed over](image-url)
Failure to find

Similarly, Massive’s ADSR envelope is hidden by default, and the widget to access it is non-standard. The access control is a row of tabs beneath another row of tabs, and they have different styling than the tabs above (See Figure 3.4). As a result, no participant using Massive discovered the ADSR envelope, and thus were unable to use its functionality. When shown how to access it, all five participants felt that they would not have found the control on their own. This lost functionality is a critical part of effectively using Massive.

Failure to understand

When asked about various controls, users were unable to correctly describe the function of many of the controls of both UIs. In the case of Rounds, the ADSR control (see Figure 3.5) was not understood by any of the participants. However, two of the participants used functionality of the ADSR envelope, but without realizing that they had. This is a much better situation than Massive, because although they were unable to link the concept and the control, they were able to use it in practice. For Massive, many of the knobs were modified, and a change in sound was noted. However, it was often the case that the participants could not accurately describe what a control was doing, except in the somewhat simple cases, e.g. the volume control. Furthermore, in the first survey, no participant felt that they had a good understanding of how the synthesizer functioned.
4.1.6 Action Feedback

In both UIs controls often have very little effect on the overall sound, due to configuration somewhere else. For Massive, all participants made modifications to widgets that had little to no appreciable effect on the sound. Because knobs in Massive are all the exact same size, users cannot use size to determine the importance of parameters. Massive controls are also very similar in colour, again providing no hints to the importance of controls. Without these visual cues, participants were required to guess at a parameter’s importance, and they frequently guessed incorrectly. Massive also uses a very hard to see indicator for showing if a module is active or not (see Figures 4.8). One participant ended up modifying an inactive module in his attempt to hear differences.

Figure 4.8: The image on the left is an module that is enabled, while the image on the right is a module that has been disabled

Both Massive and Rounds had trouble illustrating appropriate state-based feedback to participants, whether it was highlighting controls which would have a strong effect on the output; letting users know which mode they were in; or even alerting users to controls that
were currently disabled.

4.1.7 Importance of Presets

Each of the synthesizers comes with presets. These presets are configurations that are developed by experts, which show off the different abilities of a synthesizer. Presets were used extensively by the participants in order to explore the interface. Participants of both synthesizers were surprised by both the number of presets available, and the variety of sounds that they could make. The surprise at the types of sounds produced is interesting, because it suggests that the participants did not grasp the versatility of these. All participant explored the presets to some degree. Notably, one Massive participant would switch between two presets that were similarly categorized. This allowed them to look for changes in the configuration, and see how the sound changed based on those changes. Presets seem to be very important for novice users, because they help the user understand what the synthesizer is capable of, and what is possible with it.

4.2 Engagement

Most participants indicated that they found the experience fun and engaging. Wallis, Ingalls, Campana and Vuong note that enjoyment can lead to long-term engagement, because you cannot have long-term engagement if the activity is not enjoyable [61]. Two participants who were using Massive terminated during the second phase of the experiment
before time had expired, expressing frustration with using the synthesizer. One of the two, paradoxically, said “I had fun using the synthesizer, but I didn’t enjoy it.” When asked to expand, they noted that the task was fun, but the interface was not. One user of Rounds also asked if they would be finished as soon as time expired for the phase. However, other than the two who quit, all others expressed that they had fun exploring the programs, despite being frustrated. Massive’s UI, designed with expert users in mind, polarized novice participants with 2/5 ending their session early. In contrast, no Rounds participants quit prematurely. The complexity of Massive’s UI appeared to reduce short-term engagement. While more study is needed to determine why this is the case, we believe that the issues identified above contribute to the disengagement. More confirmation of this can be seen in the qualitative analysis of the data, where there was one response of “Somewhat not” to the question of “Did you enjoy using this synthesizer?” after both phases. This is surprising, given that two of the participants asked to stop early. Given the results of the previous section, I believe that the UI plays an integral role in fostering that engagement. A UI which does not present an understandable model provides more opportunities for the user to lose engagement. However, if these issues are addressed appropriately, software synthesizers can be engaging for novice users.
Chapter 5. Conclusions

This chapter provides research highlights, including a list of suggestions for designers of software synthesizers. Limitations of the work are discussed, and the chapter ends with a discussion of future work.

5.1 Research Highlights

5.1.1 Usability for Novices

Overall, the look and layout of the synthesizers hindered the participants more than it helped. The new controls and terminology place an extra burden on the user at a time when they are also attempting to develop a mental model for further use. The design of the interface could benefit from a more empathetic view towards beginner users. Two suggestions for improving this aspect are provided below, with the intent that these would benefit novice users, while also not burdening or hampering advanced users. Firstly, an attempt to indicate through the UI when the user is mousing over a control that can be interacted with. Indeed, some synthesizers are already doing this, such as the open-source synthesizer Helm [40] (see Figure 5.1). Some of Helm’s widgets change colour when moused-over, indicating that they can be interacted with. Furthermore, tooltips, or a similar system for providing information and context to controls would be an immense help to novice users.
As experts, we are prone to jargon and assumptions of terminology, much of which new users do not know. By implementing ways for them to learn inside of the application we can ease the experience burden. Tooltips can also be beneficial to expert users who are using a new synthesizer, as it will typically have features or usage patterns not found in their synthesizer of choice. These suggestions are meant to address the issues found in 4.1.2, which are low-hanging fruit for improvements.

Developing a complete mental model is a long-term goal. Even with this in mind, participants struggled to meaningfully understand the underlying system of either synthesizer, and did not display a well developed mental model after their first use. While Round’s experienced a better initial ranking, this was hampered as participants discovered its layering functionality. Conversely, Massive had initial troubles being understood, but the participants felt a large growth in comprehension during the time spent with the interface. Rounds suffered in connecting it’s more advanced layering interface with its easy to understand synthesizer configuration, while Massive struggled to provide a starting place for the new users. One reason that Rounds’ UI was better received could be due to the way it organizes its configuration screen. By presenting the information in a more understandable
manner, it makes the UI more likely to be understood, and thus engaging. However, Massive shows us that providing a large amount of information may be better long term, and that a combination of a large amount of well presented controls may be the ideal approach. Furthermore, showing the changing values of controls over time could allow the novice users to quickly grasp the myriad of connections present in an interface.

The importance of presets cannot be understated. Presets can be used to show novice users what a synthesizer is capable of, and any and all interesting sounds should be included. Thus, a good interface for browsing presets should also be included, which is something that Massive excelled at. However, this is the only tool that the synthesizers provide in application to assist with the learning process. An initial tutorial, as well as tools to better inform the user of the current state of the synthesizer (discussed above) would all be welcome additions.

5.1.2 Engagement

Despite UI issues, a majority of the participants enjoyed their experiences with the interfaces. The act of making music may be fun enough to override some UI concerns that users will experience. However, enjoyment is only one facet of engagement, as can be seen by the Massive participant who enjoyed the activity, but also dropped out early. Thus, it is important to design these interfaces with usability in mind, as novice users can be engaged with synthesizers.
5.1.3 Evaluation Methodology

Much of the literature on musical device usability evaluations focuses on theoretical evaluations of usability tests. By contrast, as shown in this thesis, usability testing can be used to evaluate software synthesizer UIs. By showing that these UI testing methods resulted in valuable information that reinforced previous research, e.g., the work done by Barbosa et al. in their proposed evaluation framework, and the benefits to having guided and un-guided sessions.

5.2 Limitations of Work

The number of participants is the greatest limitation in this work. While 10 participants was enough to get a qualitative grasp on the subject, a more in-depth analysis will require more participants. More participants will strengthen the results drawn from the data.

In addition, participants were not able to use the interfaces for an extended period of time. They were only able to use the UI for 20 minutes. While this was a suitable amount of time to get their initial impressions, it does not allow us to see how their perception of the UI changes over time. Relatedly, the number of sessions for each participant was also a limitation. By only having the participants interact with the UI in one session, we miss out on how time affects how they see the UI (e.g., learning it), or if they grow to like certain controls more (i.e., become more expert). Further sessions would also allow us to evaluate the process of novice users becoming more well versed in the tools, and how their attitudes
toward the controls change.

This research only focussed on the experience of novice users. By ignoring expert users, I could not compare how a design choice that novices dislike may instead benefit users in the long run. That is, something that novice users may find hard to comprehend or conceptualize may, with practice, end up saving time or energy in the future when the user is no longer a novice.

5.3 Future Work

By developing a prototype synthesizer, even a simple one, we can further re-evaluate the suggestions to developers (see Section 5.1), and if not, determine where it can be improved. We can also find and compare synthesizers who do implement similar features, and study them to determine if they are beneficial to the user experience.

More information regarding expert users can be helpful, especially non-technical experts. Much of the research and development is performed by extremely technically minded musicians, who are able to develop tools for their own use. A study of non-technical experts may reveal faults or insights into the current mode of thinking, and either confirm or break assumptions. An additional experiment where novice users are asked to compare and contrast two or more interfaces may prove useful. We could find the ideal controls by allowing the participants to identify aspects of the interface they enjoy or prefer.

As a longer-term study, exploring the shared attributes between the short-term and long-term engagement, in order to develop a model for the relationship between the two.
References


Appendices

Demographics Survey

Gender

- Male
- Female
- Other

How often do you use your computer daily?

- Less than 1 hour
- 1 - 2 hours
- 2 - 4 hours
- 4 - 8 hours
- 8+ hours

What is your area of study, or current career?

Have you ever used a computer to create or record music?
Have you heard of digital musical synthesizers?

- Yes
- No
- Can’t remember

Have you ever created music using any musical synthesizer?

- Yes, hardware-based
- Yes, software-based
- Yes, hardware- and software-based
- Maybe
- No

Are you an avid music listener?

- Definitely yes
- Somewhat yes
• Somewhat not

• Definitely not

Are you interested in creating music?

• Definitely yes

• Somewhat yes

• Somewhat not

• Definitely not

Are you excited to commence with the experiment?

• Definitely yes

• Somewhat yes

• Somewhat not

• Definitely not
Post-Session Survey

Did you find the user-interface appealing?

- Definitely yes
- Somewhat yes
- Somewhat not
- Definitely not

How long do you think you would have to spend using this synthesizer to master it?

- 1 day
- 1 week
- 1 month
- 1 year
- More than 1 year
- I would never be able to master it

Did you find the controls easy to understand?

- Definitely yes
• Somewhat yes

• Somewhat not

• Definitely not

Did you enjoy using the synthesizer?

• Definitely yes

• Somewhat yes

• Somewhat not

• Definitely not

Do you think other synthesizers are more or less complex than the one you used?

• Much more complex

• More complex

• Equal complexity

• Less complex

• Much less complex

Do you feel that you understand how the synthesizer functions?

• Definitely yes
• Somewhat yes

• Somewhat not

• Definitely not

Would you like to continue using this synthesizer?

• Definitely yes

• Somewhat yes

• Somewhat not

• Definitely not

Would you like to explore more digital synthesizers?

• Definitely yes

• Somewhat yes

• Somewhat not

• Definitely not

Do you feel that you have a better understanding of how electronic music is made?

• Definitely yes

• Somewhat yes
• Somewhat not

• Definitely not

Did you feel that the synthesizer was limiting in any way?

• Definitely yes

• Somewhat yes

• Somewhat not

• Definitely not

Did the synthesizer surprise you in any way?

• Definitely yes

• Somewhat yes

• Somewhat not

• Definitely not

Would you purchase this synthesizer?

• Definitely yes

• Somewhat yes

• Somewhat not
Did you have fun using this synthesizer?

- Definitely yes
- Somewhat yes
- Somewhat not
- Definitely not

What I was able to produce was worth the effort I had to exert to produce it.

- Definitely yes
- Somewhat yes
- Somewhat not
- Definitely not

I was able to be very expressive and creative while doing the activity.

- Definitely yes
- Somewhat yes
- Somewhat not
- Definitely not
It was easy for me to explore many different ideas, options, designs, or outcomes.

- Definitely yes
- Somewhat yes
- Somewhat not
- Definitely not

My attention was fully tuned to the activity, and I forgot about the system/tool I was using.

- Definitely yes
- Somewhat yes
- Somewhat not
- Definitely not

I enjoyed this activity and would do it again.

- Definitely yes
- Somewhat yes
- Somewhat not
- Definitely not