DESIGN AND EVALUATION OF A MULTI-USER COLLABORATIVE AUDIO ENVIRONMENT FOR MUSICAL EXPERIMENTATION

By

Tyler Sammann

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: COMPUTER SCIENCE

Examining Committee:

_____

Barbara Cutler, Thesis Adviser

_____

Jonas Braasch, Member

_____

Ben Chang, Member

Rensselaer Polytechnic Institute
Troy, New York

July 2013
(For Graduation August 2013)

CONTENTS

LIST OF FIGURES v

ACKNOWLEDGMENT vi

ABSTRACT vii

1. INTRODUCTION 1

1.1 Digital Audio Workstation Conventions 2

1.1.1 Sliders and Knobs 2

1.1.2 Mouse and Keyboard Use 3

1.1.3 Tracks 4

1.1.4 Progress Bar (Ruler) 4

1.1.5 Effects Bus 5

1.1.6 Piano Roll and MIDI 5

1.1.7 Drum Sequencer 6

1.2 Contributions and Thesis Outline 7

1.3 Summary 7

2. RELATED COMMERCIAL IMPLEMENTATIONS 9

2.1 Traditional DAWs 9

2.1.1 Pro Tools 9

2.1.2 Ableton Live 10

2.1.3 Linux MultiMedia Studio (LMMS) 10

2.1.4 Audacity 10

2.2 Non-Traditional Audio Environments 11

2.2.1 reacTable 11

2.2.2 Jam-O-Drum 12

2.2.3 Argos 12

2.2.4 Pure Data and Max/MSP 13

2.3 Summary 14

3. RELATED ACADEMIC WORKS 15

3.1 Collaboration and Multi-User Interfaces 15

3.2 Audio Interface Design 16

3.3 Summary 19

4. DESIGN OF THE INTERFACE 20
4.1 Simultaneous Interaction 20
4.2 Elements and Features 21
4.2.1 Button 22
4.2.2 PolyButton 23
4.2.3 Effect 25
4.2.4 FadeLabel 25
4.2.5 Track 26
4.3 Interface Structure 27
4.3.1 Track Region 27
4.3.2 Bank Region 28
4.3.3 Control Region 30
4.3.4 Recycle Bin 31
4.3.5 Recording Dialog 32
4.4 Summary 33
5. USER STUDIES 36
5.1 Study Organization 36
5.2 Preliminary User Study Design 37
5.3 Preliminary User Study Results 39
5.3.1 Observations 39
5.3.2 Exit Survey 39
5.3.3 Preliminary User Study Conclusions 42
5.4 Changes and Optimizations After Preliminary User Study 43
5.5 Revised User Study Design 43
5.5.1 Recording Phase Goals and Design 43
5.5.2 Effects Phase Goals and Design 44
5.6 Revised User Study Results 45
5.6.1 Observations 45
5.6.2 Exit Survey 46
5.6.3 Revised User Study Conclusions 49
5.7 Summary and Conclusions 50

6. SYSTEM IMPLEMENTATION 52
6.1 Simultaneous Mouse Input 52
6.2 Interface Implementation 55
6.2.1 PolyButtons 55
6.2.2 Tracks 56
6.2.3 Track Region 56
6.2.4 Interface Connection With Sound Back End 57
6.3 Sound Back End Implementation 58
6.3.1 Pure Data 58
6.3.2 Linux MultiMedia Studio 60
6.4 System Limitations 60
6.5 Summary and Future Work 61
7. CONCLUSION AND DISCUSSION 62
7.1 Presented Contributions 62
7.2 Discussion 63
References 64

LIST OF FIGURES

1.1 LMMS with labeled DAW conventions 3

1.2 A piano roll and drum sequencer from LMMS 5

4.1 Overall layout of the interface 21

4.2 Design of Cursors in down-state and up-state 22

4.3 Expanded PolyButtons 25

4.4 Adding a new Track to the Track Region 29

4.5 The interface with the Recycle Bin expanded 30

4.6 The interface with the Recording Dialog opened 34

4.7 The interface compared to LMMS 35

5.1 Images of three users interacting with the system 37

5.2 Graph of average user understanding of interface elements 41

5.3 Graph of users' average previous experience 46

5.4 Graph of each user's previous experience 51

6.1 A system diagram for the entire audio environment 53

6.2 A system diagram of the PolyMouse interface 54

ACKNOWLEDGMENT

I would like to thank my thesis advisor Barb Cutler for her dedication to me and my projects, for always motivating me, and for providing a terrific work environment. I would like to thank my family for the continued support that made this and my education possible. I would like to thank my friends for their support and encouragement. I would like to thank my thesis committee for their support and commitment. I would like to thank the RPI Office of Graduate Education for their flexibility and commitment to my education on an individual level.

ABSTRACT

This thesis presents a new type of digital audio environment based on musical experimentation and exploration rather than high quality digital audio production. The system is designed for users with and without experience in the realm of audio software and music in general. The interface for the system is designed for multiple simultaneous user interaction. Multiple functional mouse cursors are displayed on a single screen, and are controlled by users via individual USB mice. Simultaneous user interaction harbors collaboration between users, and allows less experienced users to learn from other users. Unique elements of the interface are designed around the concept of parallel user interaction.

This thesis presents PolyButton interface elements. These multiparametric sliders use the vertices of polygonal shapes to represent and modify numerous sound parameters that are categorically similar. Rules governing the selection and ownership of PolyButtons ensure that multiple users can simultaneously interact with different PolyButtons, but an individual PolyButton can only be used by a single user at a time. Finally, this thesis presents methods for the user studies conducted with the system, and presents an evaluation of the system based on the user study results and user feedback.

CHAPTER 1
INTRODUCTION

Digital Audio Workstations (DAWs) were the computer based response to the problems with traditional tape based multi-track recording systems. They enabled those with professional and personal computers to record and edit multiple tracks of audio without the limitations of physical media. Many variations of this type of audio system emerged in the early 1980s, and it wasn't long before Digidesign, the company that would go on to create Pro Tools, became the dominating force in the digital audio production market. Today, owned by Avid, Pro Tools is regarded as the industry standard for professional audio recording and editing [1].

Now, more than thirty years after the advent of the Digital Audio Workstation, hundreds of software options exist for professional and amateur use. But, despite the number of software options available, the variety of functionality amongst these systems remains relatively low. The impact of systems like Pro Tools is widespread. Those professionals and amateurs who have taken the time to learn about a DAW like Pro Tools start to think about sound editing and recording processes in terms of the interface and functionality of that system. This argument offers one explanation as to why the developers of DAWs have overwhelmingly chosen not to stray far from the established conventions.

Convention is not inherently bad. It allows those with previous knowledge of DAWs to quickly understand the functionality of any new system that follows the same conventions. The problem lies with new users, who have no choice but to devote time to learning these established conventions, for which the learning curve can be steep. In a commercial setting, it is rational for a software company to stick to established conventions, considering that the bulk of their customer base has used a DAW or audio production hardware before. Additionally, for new users who are serious about digital audio production, there is clear motivation and value in learning the conventions of these types of systems.

Individuals or groups with an interest in music, sound, and composition that

lack a serious interest in digital audio production are often left without a software product that fits their needs. This thesis presents a system created for such users, as well as users with previous experience in digital audio production and knowledge of typical DAW conventions. The system is also designed to maximize collaborative possibilities for simultaneous users with varied levels of experience. Such communication and collaboration can further alleviate the learning time required to successfully and effectively utilize the system. The system caters to the needs of those individuals, or groups of individuals, who would benefit from a simplified tool that aids them creatively while composing. The multi-user aspect of the system further enhances the collaborative possibilities of the system for groups of individuals. It thereby provides a new kind of creative platform for bands, disc jockeys, or other groups of individuals who create music collaboratively. Given its purpose, the system is not designed for polished audio production. It is designed to be used as a collaborative sketchpad for compositional ideas or creative exploration, rather than a traditional detail-rich audio production tool.

1.1 Digital Audio Workstation Conventions

The following is a brief description of some of the conventions used in modern DAWs and other digital audio production software. Examples of the conventions described can be found in Pro Tools software, as well as most other Digital Audio Workstations.

1.1.1 Sliders and Knobs

Sliders and knobs are used to modify parameters within the editing environment, and can control anything from track volume to specific parameters of a particular filter or effect. The two types of controls are essentially interchangeable in terms of functionality. Both allow users to adjust one parameter in terms of a single dimension on a sliding scale, from *lowest* to *highest*, or from *least* to *most*. Figure 1.1 depicts the interface of a traditional DAW and labels sliders and knobs. Convention sometimes dictates a particular parameter use either a slider or a knob, because sliders and knobs are interface metaphors for similar features of analog au-

Figure 1.1: This figure shows a screen capture of the LMMS interface, with labels pointing out conventions used in most traditional DAWs [2].

dio production multi-tracking hardware. Often, many sliders or knobs are placed close together in order to be used in conjunction, allowing users the ability to finely control many different parameters of a certain filter or other feature. For example, some visual equalizers use many sliders placed side-by-side (like their analog counterparts), giving users fine control over a large number of frequency ranges in the spectral domain.

1.1.2 Mouse and Keyboard Use

In most modern DAWs, control is provided to one user at a time through the operating system's standard mouse and keyboard. Sliders and knobs are typically controlled with the mouse one at a time, and give the user control in one dimension with a dragging motion. Any control or radio buttons available may be clicked and utilized one at a time. Many DAWs implement drag and drop capabilities with some of their features, especially where effects and filters are concerned.

Computer keyboards allow for specifying values or names, and search functionality, as well as pre-defined or user-defined shortcuts. Keyboard shortcuts give users fast access to frequently used functionality. Learning about shortcuts when learning about a DAW can greatly enhance productivity, but the learning process can be very steep. Though some shortcuts may carry over from standard conventions included in many other software interfaces (such as `ctrl` + `s` for *save*), others are unique to specific DAWs and potentially esoteric (such as `alt` + `Page Down` to scroll the edit window to the right in Pro Tools). In most cases, keyboard shortcuts have functionality that is available by way of an alternative series of mouse interactions, but these are often cumbersome and time consuming.

1.1.3 Tracks

Tracks are a widely used and relatively abstract convention. They have origins in analog sound editing and mixing hardware by name, and are essentially containers for sections of audio. They visualize and distinguish different pieces of recorded audio, or MIDI notes serving the same purpose. While the tracks can be muted and modified separately, the audio playback that users hear is generally a mix of all the current tracks. Figure 1.1 depicts the interface of a traditional DAW and labels its tracks. Sometimes a special *master track* (or *master fader*) is used to represent the mix of all other tracks. Tracks often represent the audio they contain from left to right in regards to time. They are typically arranged vertically, allowing users to align audio segments in time [3].

1.1.4 Progress Bar (Ruler)

The progress bar or ruler occasionally provides large amounts of functionality, but at its most basic level, it provides visual indications of time. Convention dictates that tracks sonically progress through time from left to right, the left most point indicating the beginning of the track. As such, the progress bar provides ruler-like time markings from left to right, and a cursor indicating which part of the track is currently playing in relation to those time markings. Culturally, the progress bar follows the Western left-to-right reading pattern. Figure 1.1 depicts the interface of a traditional DAW and labels the progress bar.

Figure 1.2: This figure shows a screen capture of a piano roll and a drum sequencer from the LMMS interface, with labels pointing out where MIDI notes are used. MIDI notes in both are fixed to a grid pattern horizontally to conform to the beat. MIDI notes in the piano roll are vertically aligned according to pitch. MIDI notes in the drum sequencer are vertically aligned according to instrument/drum sample [2].

1.1.5 Effects Bus

Though this convention varies more than the others from application to application, most follow the same basic functionality. In general, an effects bus gives the user the ability to "place" effects or filters somewhere, and send particular channels or tracks through this bus. The output sound for these channels or tracks is then affected in real time by the filters or effects in the bus. In some cases, an application will directly pre-process sections of the recorded audio rather than using a real-time effects bus. Such is the case with Audacity software [4]. Figure 1.1 shows the modification interface for a few different effects in a traditional DAW.

1.1.6 Piano Roll and MIDI

The use of the *Musical Instrument Digital Interface* (MIDI) protocol is very common in digital audio applications and DAWs. It is a standard that specifies

ways in which music hardware and devices can communicate with various types of software. The protocol standardizes the notation of music in time, as well as the ways that various parameters can be modified [5]. Many DAWs include tracks that can contain MIDI data instead of audio recorded with a microphone. These tracks use synthesized virtual instruments for audio playback.

Piano roll is a tool in many DAWs that allows users to create and modify the arrangement of MIDI notes in a track. It uses a progress bar, and many thin vertically arranged track-like bars which each correspond to a single pitch on a piano (in the Western 12-tone equal temperament scale). An image of a piano is often presented on the left of the editing window with the keys arranged vertically as a user aid. Vertical bars across the tracks correspond with Western note durations, thus forming a grid. When a rectangle is filled by the user, a MIDI note is created; its pitch based on its location vertically, its duration based on its width, and its occurrence in time based on its horizontal location. Timis et al. define a piano roll tool as part of a US Patent [6]. Figure 1.2 shows the piano roll from a traditional DAW and points out the location of MIDI notation.

1.1.7 Drum Sequencer

A drum sequencer or drum machine is a MIDI virtual instrument or tool that allows a user to specify a drum beat for an entire piece, or sections of a piece. The sequencer sometimes contains a specific piano roll for custom beat editing, but often allows users to choose from large banks of previously created MIDI notations that produce different types of drum beats. The user-selected drum beat simply loops for the duration of the piece. Drum sequencers allow less experienced users to create an often necessary drum beat or rhythm track for their piece, and they allow experienced users to do it more quickly and easily. Because the same drum beat loops for the entire piece, the cost of using a drum sequencer is the lack of customization throughout the piece. Figure 1.2 shows the drum sequencer from a traditional DAW and points out the location of MIDI notation in the sequencer.

1.2 Contributions and Thesis Outline

This thesis presents the following contributions:

1. The PolyMouse interface: a novel implementation allowing for multiple simultaneous user input from multiple USB mice. Custom mouse cursors are displayed on a single screen and control elements of the same interface.

2. PolyButtons: polygon-shaped multiparametric two-dimensional sliders with few-to-many mappings. These interface elements promote fast and easy experimentation rather than precise decision making and fine tuning.

3. Group users studies and results for the audio environment are presented in this thesis. The results point out areas of potential improvement, but confirm the intuitiveness and success of PolyButtons, PolyMice, and the collaborative aspects of the audio environment.

This thesis will first present and discuss related commercial implementations in Chapter 2, including both traditional and non-traditional audio environments. In Chapter 3, related academic works will be discussed in the fields of multi-user interfaces and audio interface design. Chapter 4 will discuss the design of the interface for the audio environment presented in this thesis. Chapter 5 will detail the design, execution, and results of a preliminary user study and revised user study. Chapter 6 will describe the implementation details for system, including the implementation of PolyMice, the audio environment interface implementation, and the implementation of the back end applications that handle all recording and sound production for the audio environment. Finally, Chapter 7 will summarize the topics and contributions presented by the thesis and offer final conclusions about them.

1.3 Summary

This chapter has discussed the problems with currently available audio editing software and audio environments, and provided necessary motivation for the creation of a system designed for a broader range of users that want an audio experimentation scratch pad rather than a tool for precise audio production. Conventional elements

of traditional DAWs were explained for comparison purposes in the rest of the thesis document. Finally, this chapter listed the contributions presented in this thesis, and an outline of the remainder of the document.

CHAPTER 2
RELATED COMMERCIAL IMPLEMENTATIONS

This chapter describes existing pieces of audio software and digital audio interfaces, and discusses their advantages and disadvantages. Traditional DAWs are discussed first, followed by non-traditional audio environments.

2.1 Traditional DAWs

Traditional DAWs are among the most complex digital audio environments, and allow experienced users to fine tune digital audio for the purposes of sound and music production. They tend to favor power and control over simplicity and intuitiveness.

2.1.1 Pro Tools

Pro Tools and its predecessors were among the first software products to ever don the title of *Digital Audio Workstation*, and Pro Tools is arguably most influential piece of audio software in existence. It is the industry standard for professional audio production, but some versions of the software were designed for home studios and experienced amateurs [3]. It is feature rich and complex, allowing for precise control and providing a wide range of sound modification options. It is also very extensible, allowing for a multitude of plug-in capabilities. The Pro Tools interface utilizes each of the DAW conventions mentioned in Chapter 1.

Pro Tools, and the Pro Tools interface is specifically designed for both professional and amateur use, but its complexity may be intimidating and overwhelming for an amateur. It is designed for a single experienced user that wants the most state-of-the-art audio editing software available. Its learning curve is steep, and the price for this proprietary software is very high [1].

2.1.2 Ableton Live

Ableton Live is another popular DAW, and though it implements many of the same features of Pro Tools, it has a slightly more gentle learning curve, a lower price, and is designed for a different type of user. It has fewer and less complex features in regards to professional audio editing, but a greater focus on disc jockeying, and live music production. Its on-the-fly capabilities make it a powerful tool for professional performing musicians [7]. Though it is easier to learn than Pro Tools, it is still complex and potentially overwhelming for amateur or novice users. Its interface is not designed for simultaneous user interaction [8].

2.1.3 Linux MultiMedia Studio (LMMS)

LMMS is a free and open source DAW built for Linux and Windows users. It has fewer features than both Ableton Live and Pro Tools, is less extensible, and is designed for amateurs and novices more than professionals. It implements all of the DAW conventions specified in Chapter 1, but it does not include the capability to record raw digital audio. While sampling digital recorded audio is possible in LMMS, it is most prominently designed for MIDI and virtual instrument use, and allows for high amounts of external MIDI control. Its interface is not designed for parallel interaction. It is designed for amateur use, but the interface, though simpler than Pro Tools, might still be overwhelming to a novice or amateur unfamiliar with DAW conventions [2]. Figure 1.1 labels different elements of the LMMS interface.

2.1.4 Audacity

Audacity is a free and open source audio editing application and is much simpler than all of the previously mentioned DAWs. It is available for Linux, Windows, and Apple users [4]. Its interface uses some of the DAW conventions detailed in Chapter 1, but lacks a standard piano roll, drum sequencer, and effects bus. It specializes in recording and manipulating digital audio samples directly, and has very few features related to MIDI and virtual instruments. Instead of using an effects bus to process sound during playback, any effects added to a track (or portions of a track) are pre-processed and original audio data is overwritten with the affected

audio data.

Because the interface is simple in comparison to most other DAWs, Audacity is a good choice for a novice or amateur. It has a learning curve, but it is much more gentle than most DAWs. Like the other DAWs mentioned in this section, its interface does not support simultaneous user interaction, and though unique, some of its functionality is non-intuitive.

2.2 Non-Traditional Audio Environments

The following audio environments are very different from traditional DAWs, and have different target demographics than DAWs. Many of these environments behave like instruments for live performance, or give users more flexibility and customization potential than DAWs.

2.2.1 reacTable

Jordà et al. present a breakthrough system in terms of collaboration and tangibility in musical interfaces [9]. Their reacTable system is a live musical production tabletop interface, with cameras and projectors hidden inside the table itself. Unlike many other MIDI controllers, it does not attempt to simulate traditional musical instruments, and by the same token, it does not utilize traditional HCI equipment like keyboards, mice, or joysticks. Rather, it uses specially designed cubes with recognizable and distinct geometric patterns. They provide a wealth of different types of cubes, each with multiple adjustable parameters of capability. It allows for parallel user interactions, and collaboration is especially enhanced by its tangible tabletop design with a single display surface.

Its complex but intuitive design allows users to learn how to use the system with practice, in the same kinds of ways that one might learn how to play a traditional musical instrument. Despite this intuitiveness, it is not specifically designed for novices. It is a professional grade tool for performing musicians, but it effectively demonstrates the collaborative possibilities for non-standard audio interfaces.

2.2.2 Jam-O-Drum

The Jam-O-Drum Interactive Music System designed by Blaine and Perkis is an experimental live musical creation system designed specifically for collaboration between experienced and novice users [10]. It is a tabletop system, that allows for simultaneous user interaction, and encourages collaborative improvisation between all participating users. It is designed to simulate an improvisational drum circle with six users. Each user is presented with a section of a tabletop that includes a virtual drum head, and a display with visual feedback.

Various methods were tested to enhance the experience for novice users, and allow them to more successfully participate in the creative process. Some of these include adjusting the MIDI data from user input and applying rhythmic quantization (aligning off-tempo input with nearest on-tempo beat), and rhythmic emphasis weighting (increasing the loudness of user input that is closer to being on-tempo).

The results from their various user studies outline some important aspects of designing collaborative audio interfaces for novice users. It was found that visual cues were particularly important for novice users in terms of learning and comprehending mistakes. It was harder for these users to associate correctness with non-rhythmic parameters like effects. Additionally, the most successful interaction methods were the simplest and most direct. Follow-the-leader type games were particularly effective in terms of novice user participation and learning.

2.2.3 Argos

As a response to other expensive proprietary solutions to the multi-user collaborative musical interface problem, like the reacTable, Diakopoulos et al. present an open sourced multi-touch tabletop system for building and using audio interfaces. Argos is designed for amateur use, and promotes exploration and collaboration for musical performance and provides tools for collaborative design of custom audio interfaces [11].

The system includes one mode for rapid user interface prototyping, and another mode for use of the interface prototypes. Argos is a modular front-end tool that sends out Open Sound Control (OSC) messages (A standardized wrapper for

User Datagram Protocol (UDP) socket messages) in order to control an audio back end, which can be completely customized by a user. The back end they designed for testing the system uses an OSC to MIDI converter application, and forwards those MIDI messages to a customized session in Ableton Live. Such a system has the advantage of flexibility, because the front end and back end remain independent. Also, using Ableton Live sessions for a back end allows for rapid back end prototyping and modification.

The interface elements are designed with highly contrasting color schemes and simplicity for use on many different types of screen, and allow for touch input from multiple simultaneous users. They acknowledge that problems can arise in the interface prototyping mode involving large amounts of screen clutter. Though the users of the interface can be novices, the interface designers, and especially the audio back-end designers would need to be somewhat experienced with DAWs and other audio software and specifications like MIDI and OSC.

2.2.4 Pure Data and Max/MSP

Pure Data and Max/MSP are two sister applications, originally envisioned by Miller Puckette at the University of California, San Diego [12]. They are visual *data flow* programming environments that give users without formal programming training the ability to create, or extend their own audio performances, projects, or applications. Their interfaces involve a large collection of *objects* which can be placed inside different *patch* windows, and connected with virtual *patch cords*. These visual links carry data between objects are related to the patching abilities of analog synthesizers.

These applications allow for high levels of control, and provide users with tools to create nearly any type of computer music project they can conceive, even if they are not experienced with programming. They give users the ability to create multiple user interfaces and applications, and essentially bypass the conventions of traditional DAWs. It does not require experience with digital audio production, but both Pure Data [13] and Max/MSP [14] can be challenging to learn, especially for those without a general music background. The learning curve with these applications is

much greater than with traditional DAWs, but so is the number of potential audio applications and the amount of application flexibility. Pure Data is very similar to Max/MSP functionally, but has the advantages of being open source, free, and available for Windows, Linux, and Apple systems.

2.3 Summary

Though some DAWs are not as complicated as Pro Tools, and some are designed for amateurs, no DAW is simple enough for quick and easy experimentation and audio exploration, or created for that purpose. Some non-traditional implementations include elements that would be useful for such such a purpose, like separated front and back ends, and multiple simultaneous users, and educational aspects. But, none of these implementations are designed for this purpose specifically, and most are highly expensive or inaccessible. The next Chapter describes more related academic work with less of a focus on the commercial aspects of existing software.

CHAPTER 3
RELATED ACADEMIC WORKS

The following sections describe some of the previous work in the field of collaborative multi-user interfaces, and document important principles and guidelines in relation to the design of user interfaces, especially concerning interfaces for audio environments.

3.1 Collaboration and Multi-User Interfaces

Interfaces utilizing multiple mouse cursors on a single output screen have existed for many years. A computer mouse is a classic, cheap, widely understood, functionally simple, and robust user input device, making it the preferred human input device when multi-user interaction was first conceptualized.

Bier and Freeman describe a system built for multiple users and a single display, using a mouse for each user, and individual cursors for each mouse [15]. The system allows for dynamic creation of new *user instances*, each assigned an individual *home area*. Input events from the mice are handled with an event record queuing data structure that stores time stamp, as well as device ID and state information. The text editing and windowing application built around the system was functionally very simple, but provided proof of concept for such a system, and introduced important multi-user interface elements. Such elements included the distinction of cursors based on color, and interface elements (windows) with nested scopes, allowing for parent-child management relations between elements, and element "ownership."

Sample applications for the systems that followed were often functionally simple. Stewart et al. present applications for their own multi-cursor single display architecture, coining the term *single display groupware* (SDG) [16]. They enumerate potential benefits of a SDG paradigm, the most notable being the enrichment of existing computer collaboration, and the ability for learning and teaching scenarios between users. One application for their system, named *KIDPAD*, allowed multiple users to draw simultaneously. The application presented the notion that each user

have the ability to select their own local tool (like a paintbrush) to make edits to the drawing. One of the tools available allowed for the removal of changes made by a particular user, furthering the concept of locality and ownership in the application.

Though modern operating systems are capable of handling input from multiple mice simultaneously, this does not result in the respective number of individual operating system level cursors appearing on the screen. Rather, every participating device controls the same single cursor. Pawar et al. discuss this issue, and resolve it by suppressing the primary (operating system) mouse cursor, and displaying custom multi-mouse cursors within the application itself [17]. Because the Win32 API in Windows XP ignores the unique identifiers given to each USB mouse, they utilize the RawInput API in Windows XP, which gives them access to data-rich mouse data including unique IDs, as well as callback access for each of the mice.

Pawar et al. present a second study using the same architecture for multiple mice, and more conclusively prove the failings of a system with multiple users and a single cursor [18]. The SDG paradigm was taken into the educational domain and used to create applications that extend the educational and collaborative effectiveness of one computer by allowing simultaneous users with cheap USB mice [17]. A later study showed the educational advantages and disadvantages of having strictly competitive applications and strictly collaborative applications. It was demonstrated that learning retention was much higher in a collaborative setting (vote oriented) than a competitive (race to the correct answer) one. This was especially true with groups of boys [18].

3.2 Audio Interface Design

For the design of interfaces in general, Barr et al. present a taxonomy of metaphors for use in user interfaces, and make suggestions as to the best uses for different types of metaphoric attributes [19]. They argue that using metaphor in UI design is an effective way to leverage existing user knowledge, enabling users to understand how to use a feature without being prompted or educated about it first. They break down UI metaphors into subcategories: *orientational* metaphors explain concepts in terms of space, *ontological* metaphors translate concepts into

physical objects or substances, and *structural* metaphors are similar to ontological metaphors, but use more specific real-world objects to describe concepts. They distinguish all UI metaphors as either conventional, or novel. Barr et al. continue to list suggestions for working with UI metaphors in the context of the taxonomy that they describe. The following list outlines some of these suggestions.

- When using conventional structural metaphors, ensure that it is done with knowledge of the structure involved, and the metaphoric elements such a structure entails.
- When using novel metaphors, ensure that all metaphoric features are clear to the user.
- When using orientational metaphors, ensure the metaphor fits into the conventions for that particular conceptual framework. For example, when using sliders, up is understood to mean more of something, rather than less.
- Understand the culture of the intended users, and metaphoric subjectivity, before designing metaphoric elements.

Reeves et al. describe guidelines for multimodal interface design [20]. Such interfaces might have multiple possibilities in terms of input and output, with the advantages of flexibility and extensibility. They are designed to interface with more than one type of input device, and may support multiple simultaneous input devices. They are also capable of outputting information across a variety of different devices and in different formats (i.e. auditory, visual, etc.). Reeves et al. suggest that in terms of input, each modality's advantages should be maximized, and that output should match corresponding input modalities. To help prevent errors in interfaces with mixed modalities, users should be given control over the selection of modality used. Users should also be given the ability to easily undo their actions, and when completed, the interface should output and exit from that task clearly.

In terms of designing interfaces for music and sound, there was a realization that traditional interfaces, even those that were meant to be used in real-time, were

fundamentally very different from the ways that humans interface with traditional instruments.

In response to this, Hunt and Kirk discuss strategies for mapping musical parameters in interfaces meant for musical live performance [21]. The implementation of their interface involves multiparametric elements, with one-to-many or many-to-one mappings, noting that traditional instruments are multiparametric, and almost never have one-to-one mappings between a single parameter like volume, and a single action like horizontal arm motion. Their interface also requires users to inject physical energy or motion into the system to produce sounds, as traditional instruments also require physical energy for sound production. They tested an interface that used both a computer mouse and two sliders. The mouse alone controlled three different parameters, and only produced a sound while moving horizontally (thereby requiring user energy). Their user studies showed that interfaces with multiparametric mappings, as opposed to one-to-one mappings, had the fastest learning rates. Though a small number of users preferred using one-to-one mappings (especially for volume parameters), users found multiparametric mappings more engaging.

In a later article on the same subject, Hunt and Wanderley further investigate multiparametric mappings with sound production interfaces [22]. In this second study, a similar interface was created, and it was found that the multiparametric elements allowed users to think gesturally, and think about the sounds and sound production in terms of visual shapes. The following list describes some of the guidelines they provide for creating mappings for real-time musical performance interfaces, and for testing them with users.

- The complexity of the mapping should be aligned with the user's amount of experience.
- The complexity of the user's task should be aligned with the functional capabilities of the input device being used.
- The musical tasks for the user should be well defined.
- The mappings should remain consistent over time (to aid learning and retention).

Steiner presents methods for an amateur to build a custom real-time musical instrument and corresponding interface using Pure Data software [23]. He acknowledges the potential restrictions of traditional DAWs in terms of live musical production, and encourages using Pure Data or Max/MSP coupled with MIDI message sending as an alternative. Like Hunt and Kirk [21], and Hunt and Wanderley [22], Steiner provides some guidelines for musical performance based interface design. He argues that providing users with too many dimensions of control can make instruments difficult to play, and can hamper creativity, but providing users with too few dimensions of control can heavily limit instrument capability. He therefore recommends using the similar numbers of controllable dimensions as traditional instruments, stating that four dimensions is appropriate because a violin and a piano each have four dimensions of control. He also recommends the use of *Open Sound Control* (OSC) messages as a means of abstracting multidimensional mappings in the interface.

3.3 Summary

This Chapter discussed related academic works, in the fields of multi-user interfaces, and user interface design principles mostly in relation to audio environments. The design of the system presented in this thesis involves simultaneous multi-user interaction capabilities, and promotes collaboration rather than competition.

The presented audio environment attempts to follow conventional frameworks for structural and orientational metaphors, and and follows cultural metaphoric expectations where applicable (such flow from left to right). In terms of design aspects, the interface utilizes the advantages of multiparametric few-to-many mappings, uses established conventions like drag and drop, and features simple error recovery mechanisms.

CHAPTER 4
DESIGN OF THE INTERFACE

The interface for the system presented in this thesis is designed to be simple, minimal, and user friendly for multiple simultaneous users. It includes many features which are highly simplified versions of similar features used in traditional DAWs. The interface is designed to be intuitive and easy to learn, even for non-experienced or novice users. Figure 4.1 shows the general layout of the interface, and Figure 4.7 highlights some of the most functionally comparable areas of this interface to the interface used in LMMS, a traditional DAW. The names of features and elements specific to the interface presented in this thesis will be capitalized throughout the document for disambiguation.

The interface is designed to allow users to record individual *Tracks* of audio and hear all of these recorded audio Tracks mixed together during playback. Individual Tracks can be modified separately, or all together using the *Master Track*. A non-removable *Drum Track* allows users to select and modify an acceptable drum beat or rhythm for their piece. Elements can be removed from the piece, but nothing can be permanently deleted.

4.1 Simultaneous Interaction

Multiple users are able to simultaneously interact with the interface. It is an example of *single display groupware*, described by Stewart et al. as software with one display and an interface that can be simultaneously manipulated by more than one user [16]. Interaction devices can include USB mice, but could easily include laser pointers, or other devices. For each USB mouse or other input device used, a custom cursor is displayed on the interface. The cursor has a unique color to distinguish users from each other. Each cursor can be moved horizontally and vertically, and can be in an *up-state* or a *down-state*.

When a cursor is in a down-state, an outline appears around the cursor with a color slightly lighter than the color of the cursor. This highlights the cursor,

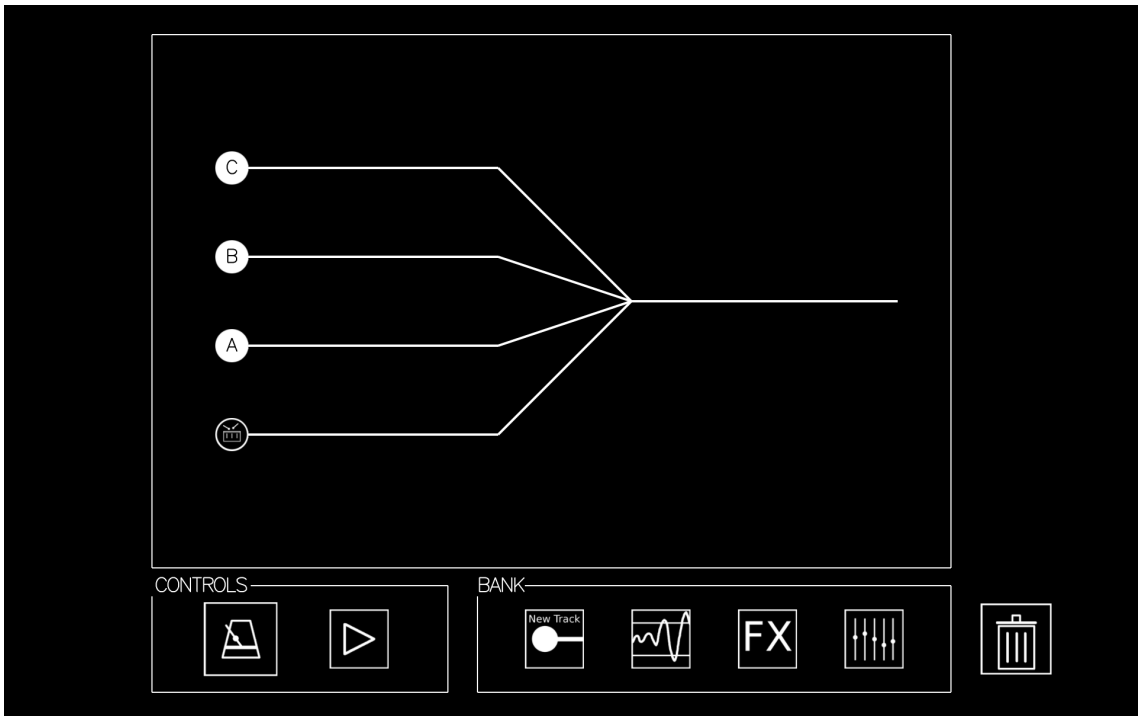

Figure 4.1: This figure shows a screen capture of the interface, depicting its overall layout and structure. Three Normal Tracks have been recorded and added to the Track Region.

providing feedback to all users and informing them of the cursors are in a down-state. This is especially useful for the users of the other cursors, because they would otherwise have a hard time determining the states of the other cursors. This functionality can be seen in Figure 4.2.

At least one user in each group must use an instrument like input device (either a MIDI controller, an electric traditional instrument, or microphone input) to produce sounds and record new audio data for Tracks. The ability to handle input from multiple instruments in the back end dramatically increases the demographic of users capable of recording Tracks effectively.

4.2 Elements and Features

Buttons and PolyButtons, expanding polygonal extensions of a Buttons, are discussed in the next section, as well as the FadeLabels that are used to label the vertices of expanded PolyButtons. Effects and Track handles, implemented as

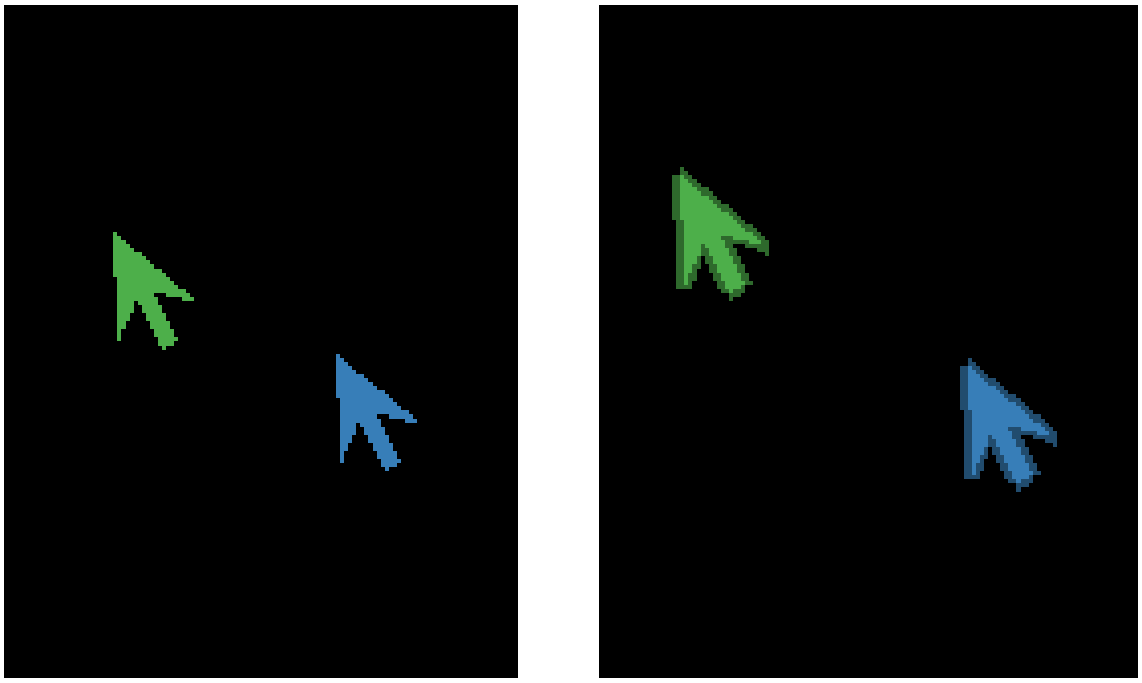

Figure 4.2: This screen shot depicts how individual Cursors are rendered on the display. The left image shows Cursors in an up-state position, meaning that the left mouse button is up. The right images show the same Cursors in a down-state position, meaning that the left mouse button is down. The outline around Cursors in a down-state indicates the current state of any PolyMouse to all users of the system.

PolyButtons, are explained in this section. Finally, this section discusses the three different types of Tracks in the system.

4.2.1 Button

The interface uses and extends a relatively simple *Button* element. The Button element is a simple rectangular or circular shaped feature, which can be clicked on or moved by a cursor. The Button can be rendered with a with a loaded image, with customizable text, or with a solid color. When a Button is selected by a cursor, a translucent border for the Button is displayed in the color of the cursor that selected it. This gives immediate visual feedback to the user as recommended by Blaine and Perkis [10], not only informing users that the Button was selected, but distinguishing the user that currently controls the Button. A Button can only be selected by one

cursor at a time. Buttons can be seen in the Bank Region of the interface (Bottom Right section) in Figure 4.1.

4.2.2 PolyButton

The *PolyButton* element is an extension of the Button feature. It has two different states. The collapsed initial state is the same as that of the Button feature; a rectangular or circular element that can be rendered with an image or text. Selection of a PolyButton is different than a regular Button element. In order to select and expand the PolyButton, it must be clicked and released. Depending on their role in the interface, certain PolyButtons can be moved like regular Buttons. When released, if the cursor is close to its original clicking position, the PolyButton is expanded, otherwise it remains collapsed. This helps ensure that the PolyButton does not expand after a user has dragged it somewhere else in the interface. This is not an issue with PolyButtons that users cannot move by dragging.

In the PolyButton's expanded state, the Button element from the collapsed state is not displayed. Instead, a polygonal shape is displayed. The outline of the polygon is multicolored and each vertex is assigned a different color and label. The color of the outline between two vertices is a gradient interpolation of those two vertex colors. The background color of the polygon corresponds to the color of the cursor that expanded the PolyButton. This allows users to quickly understand that there is a correlation between a specific Cursor and an expanded PolyButton, and provides visual feedback so users can quickly determine the PolyButton ownership, a concept described by Bier and Freeman [15]. A small white circular button resides within the polygon. Lines are drawn between this central circular button and the vertices of the polygon. The colors of these lines are based on the color of the respective polygon vertex, and the line's width is based on the proximity to the respective polygon vertex. A line is thickest when the button is closest to its respective polygon vertex, and will not even be displayed if far enough from its respective polygon. The polygon is always equilateral, and can have two or more vertices. A PolyButton with only two vertices is displayed as a line with a central button that can be moved from one vertex to another. Figure 4.3 shows the

expanded PolyButton design.

Functionally, an expanded PolyButton element with three or more vertices acts as a multiparametric slider interface with the ability to modify three or more parameters with only two dimensions of control. The is an example of a few-to-many mapping interface, as described by Hunt and Wanderley [22]. When a PolyButton has only two parameters, it functions more like a traditional single-parameter slider. Each vertex of the expanded PolyButton's polygon corresponds to a specific parameter. When the central button is closer to a polygonal vertex, its respective parameter value is increased, thereby allowing multiple parameters to be modified simultaneously by moving the central button within the polygon. The thickness of the lines drawn between the central button and the vertices provides users with visual cues as to the levels of the different parameters they are controlling.

In terms of interaction, while the PolyButton is expanded, it can only be used by the cursor that expanded it. This cursor shall be referred to as the *owner* of the PolyButton while it is expanded. If the owner of the expanded PolyButton clicks anywhere within the bounds of the polygon, the central button will jump to that location, and the parameter values will be adjusted accordingly. Once the polygon has been clicked, the central button can be dragged around the inside of the polygon for gradient control (if the cursor remains in a down/pressed state), but the central button cannot be dragged outside of the polygon. To collapse the PolyButton, and release it for use by other cursors, the owner must click outside of the expanded PolyButton's polygon. Expanded PolyButtons are restricted to one user at-a-time in order to discourage conflict between two users over the same element, and improve collaboration by encouraging the other users to interact with different elements of the interface.

Because of the design of the multiparametric mapping with PolyButtons, users are not able to represent all the possible configurations for the parameter values like they would with traditional sliders with one-to-one mappings. This restriction is intentional, and is meant to allow users to easily and quickly explore different types of sounds without being overwhelmed by too many dimensions of control, as is suggested by Steiner [23]. Because of this restriction though, the layout of the

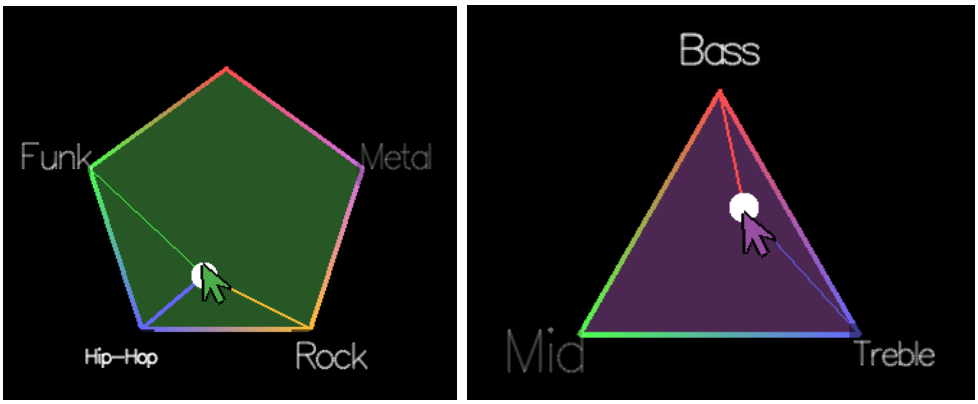

Figure 4.3: This screen capture displays two expanded PolyButtons. The PolyButton on the left is the rhythm selection PolyButton for the Drum Track. The PolyButton on the right is an equalizer Effect PolyButton. Notice that each PolyButton's vertices are labeled with FadeLabels, having different levels of opacity based on Cursor proximity.

parameters in relation to the others is important, as their relative positions dictate which parameter value configurations are possible.

4.2.3 Effect

An *Effect* is a specific type of PolyButton. It allows users to control and combine multiple traditional effect parameters with a single tool. For example, an equalizer Effect might be represented by an expanded PolyButton with a different parameter and vertex for each modifiable frequency range. Particular Effect parameters are stored within the Tracks, regardless of whether or not the Effect is active. Because of each Effect's parametric complexity and broad range of configuration possibilities, each Track can only have one active Effect for each Effect type. This design choice also favors overall interface simplicity, and attempts to curb the amount of clutter and over complication. Figure 4.3 displays an expanded equalization Effect on the right.

4.2.4 FadeLabel

A *FadeLabel* is like a traditional interface label. It displays customizable text at a specified location of the interface. FadeLabels have white text with a black

background. They also have the ability to gradually fade out when no cursors are located near them. The opacity of the FadeLabel is determined by the proximity of the nearest cursor. If no cursor is near enough, the FadeLabel is not visible. This allows for many labels to be present on the interface without it becoming cluttered and overwhelming. It also encourages collaboration because users must work together to read the FadeLabels for different parts of the interface. FadeLabels can also be specified to remain completely opaque. FadeLabels that fade based on cursor proximity are used to label the parameters of each expanded PolyButton, and are located near the respective polygonal vertex for that parameter. Figure 4.3 shows FadeLabels of different opacities around PolyButtons, based on cursor position.

4.2.5 Track

A *Track* element is used to represent a specific portion of the total sound arrangement heard during playback, be it a recorded instrument, the drum beat, or the mix of all the other tracks. All tracks are displayed as a thick horizontal white line. The line represents the track's audio portion, but not spatially with respect to time, as opposed to tracks in most traditional DAWs. All tracks have the same (relatively short) audio duration for simplicity. Effects can be added to any track by dragging and dropping. The Effect element will reside on the Track line after being added, and effects added to a track will affect that Track's audio during playback. There are three track types used in the interface: *Normal Track, Drum Track,* and *Master Track*. Figure 4.4 shows this process, and shows the how Tracks displayed.

Normal Tracks Normal tracks are created when a new audio track is recorded by a user. These tracks have a white circular PolyButton at the left side of their track line. When expanded, this PolyButton allows users to control the virtual instrument of the track during playback. Each vertex of the expanded PolyButton represents a different instrument. This PolyButton also acts as a handle for the track, and can be dragged and moved by a cursor, effectively moving the entire track. When collapsed, the PolyButton displays the ID of the Normal Track (a capital letter, from A-to-Z).

Drum Track There is only one Drum Track in the interface. It exists when then interface is launched. Like a Normal Track, it has a circular PolyButton element at the left side of its track line, but its PolyButton displays an image of a drum set. When expanded, this PolyButton allows users to control the drum beat of the track. Each vertex of the expanded PolyButton represents a different sounding drum set and rhythmic pattern. The PolyButton cannot be moved like a normal track.

Master Track There is only one Master Track in the interface. It exists when then interface is launched. It represents the mix of all other existing tracks in the interface. It does not have a PolyButton at its left side like the other tracks, and cannot be moved like a Normal Track.

4.3 Interface Structure

The overall structure of the interface can be broken down in five different sections, spatially and functionally. A rectangular *Track Region* occupies the largest portion of the display, a rectangular *Control Region* and *Bank Region*, located beneath the Track Region, occupy smaller portions of the display. A small *Recycle Bin* is also displayed, but has expansion properties to allow for interaction with removed Tracks, and when prompted a *Recording Dialog* is centrally displayed. The Bank Region, Control Region, and Recording Dialog are all labeled with non-fading FadeLabels accordingly. These sections, apart from the Recording Dialog, can be seen in Figure 4.1.

4.3.1 Track Region

The Track Region occupies the majority of the display window as it contains all Tracks and most PolyButtons, and is the primary area for user interaction. It contains all of the active tracks for a *session*. The term *session* will be used to describe one instance of a group using the interface. The Master Track occupies the central right half of the Track Region, and the Drum Track occupies the central left half of the screen. As suggested by Barr et al., this design choice considers the culture of its users, following Western left-to-right reading conventions for the

metaphoric flowing of sound through Tracks [19].

The right side of the Drum Track and all Normal Tracks are connected to the left side of the Master Track. This layout was chosen to help users understand that the Master Track is really nothing more than a combination of the other Tracks. In this way, tracks can be metaphorically seen as something more like an analog instrument or patch cable, with sound flowing through them, rather than a visual depiction of the sound itself at different locations in time. As is expected with such a metaphor, modifying the Drum Track or a Normal Track will only affect that particular Track, while modifying the Master Track will have an effect on all Tracks in the Track Region.

Normal Tracks can be added to the Track Region, and after being added, reside on the left half of the Track Region, vertically aligned with the Drum Track. Regardless of the number of tracks added, the Tracks on the left side of the Track region are always arranged to have an average vertical position in the center of the Track Region, but are appropriately and evenly spaced given the numbers currently being used in the session. Figure 4.4 shows how Tracks are arranged in the Track Region.

4.3.2 Bank Region

The Bank Region contains a group of horizontally arranged Buttons. Most of the Buttons available represent Effects, and display an icon describing that particular Effect. One special *New Track Button* in the group allows users to record and add a new Normal Track to the Track Region. The New Track Button displays an icon depicting a Normal Track handle and includes the words "New Track" for clarity.

Effects can be added to Tracks with a drag and drop motion. When a cursor clicks an Effect Button in the Bank Region, a duplicate Button is created that the user can drag around the display. If the Effect Button is released while hovering over a Track's line, that Effect will be added to the Track (In the form of a PolyButton), otherwise it will simply be deleted. The Buttons in the Bank Region however cannot be removed.

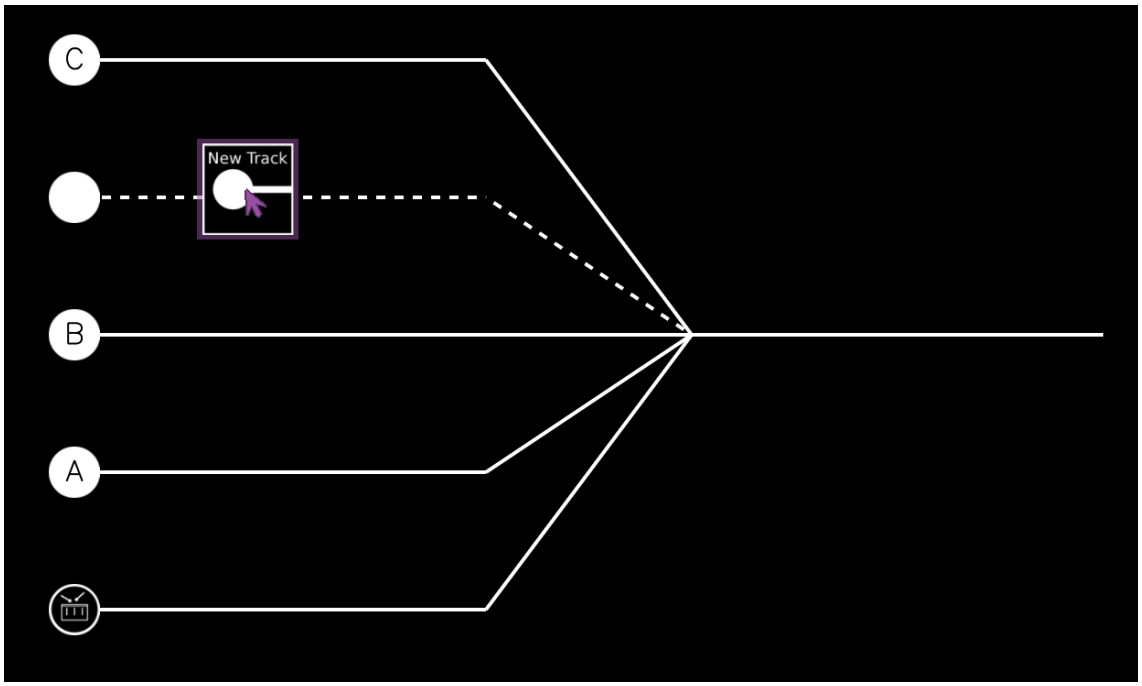

Figure 4.4: This screen capture displays the Track Region. A user is attempting to add a new Normal Track to the Track Region, and the dotted line displays where that track will be added if the user releases the New Track Button.

When a cursor clicks the New Track Button in the Bank Region, a duplicate Button is created that the user can drag around the display. If this Button hovers over the left side of the Track Region, a new Normal Track with a dashed line is added to the group of Tracks at the cursor's vertical location, and the vertical locations for all Tracks, except the Master Track, are shifted accordingly. This allows users to decide where a new Normal Track should be placed amongst the current Normal Tracks. The exception is the Drum Track, which always remains at the lowest vertical location, regardless of the number of Normal Tracks added. If released while hovering in this area, the Normal Track with the dashed line becomes a new Normal Track in the Track Region with a solid line, and the Recording Dialog window pops for recording the newly added Normal Track. If released while hovering elsewhere, the New Track Button is simply deleted and no interface action is taken. Figure 4.4 shows a user attempting to add a new Normal Track to the Track Region.

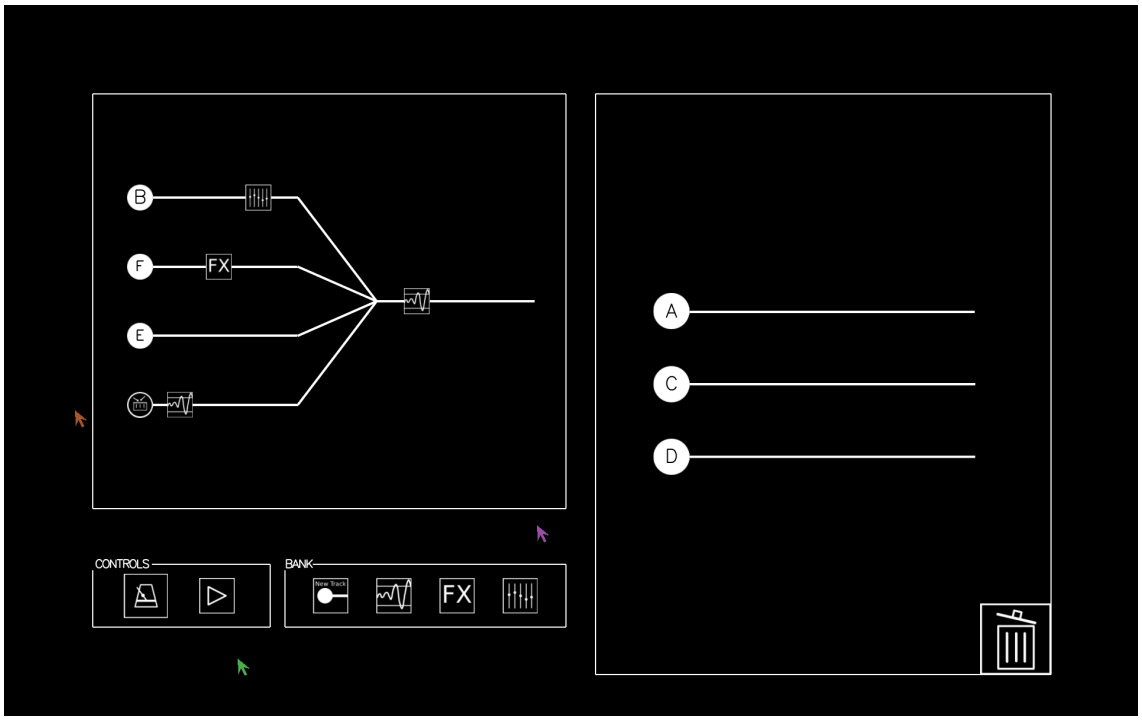

Figure 4.5: This screen capture displays the interface after the Recycle Bin has been expanded. Note the Normal Tracks that have been removed from the Track Region and reside within the Recycle Bin.

4.3.3 Control Region

The Control Region, visually similar to the Bank Region, contains controls for the session. It contains a *Tempo* PolyButton, and a specialized Button that controls the starting and stopping of session playback. The Tempo PolyButton, when expanded, has only two vertices, and therefore acts like a traditional slider. The Control Region can be seen at the bottom left in Figure 4.1. The choice to include only two control objects was one of simplicity. Playback controls and tempo controls are arguably the most important controls in more traditional audio environments, but options like fast-forward, rewind, loop, mute, nudge, etc. are not vitally important for simple audio experimentation purposes.

4.3.4 Recycle Bin

It was important that users were not given the option to permanently remove a Track from the Track Region or an Effect from any Track. In general, this allows users more freedom and gives them more confidence to explore, knowing that they cannot make unrecoverable errors. This instilled freedom and confidence is especially important for users with less experience.

From the Track Region, both Effects and Normal Tracks can be dragged to the Recycle Bin. Doing this removes that Normal Track or Effect from the current session or Track respectively. When an Effect is dragged away from a Track, if it is released somewhere outside of the Recycle Bin, it will float back to its previous position on the Track it came from. The same thing occurs with a Track's PolyButton handle when a Track is dragged from the Track Region. The floating back animation provides feedback to users, informing them that they were unsuccessful in removing the feature they attempted to remove. This immediate feedback and clear exit from the user's task, as recommended by Reeves et al. [20], helps to prevent later errors and confusion.

If the Recycle Bin is clicked by a user, a region for the Recycle Bin expands, and takes up the right half of the display. Accordingly, the Track Region, Control Region, and Bank Region are resized and moved to the left half of the display. All regions remain usable and functional regardless of whether or not the Recycle Bin is expanded. If expanded, Normal Tracks and Effects can be dragged anywhere within the Recycle Bin's region to be removed. When the Recycle Bin is collapsed, an icon depicts a closed recycling bin, and when expanded, an icon depicting the same recycling bin with an opened lid resides in the bottom right corner of the region. Figure 4.5 shows the interface when the Recycle Bin is expanded.

Because Effect parameters are stored and saved within each Track, regardless of whether that Effect is active, Effects are not added to, and stored in the Recycle Bin when removed. The Effect PolyButton simply disappears and the Effect is deactivated for the Track in question. If the Effect is re-added to that particular Track, its parameters will be the same as they were when the Effect was removed. This design was chosen to allow users to easily undo an Effect removal without

needed to search for a particular instance of an Effect in the Recycle Bin.

When removed from the Track Region, Normal Tracks are stored within the Recycle Bin, because as opposed to Effects, they each contain fundamentally different and non-modifiable recorded instrument data. The Normal Tracks contained in the Recycle Bin do not affect session playback, and are displayed in a vertical arrangement within the expanded region of the Recycle Bin. When collapsed, the Tracks are not displayed. These Normal Tracks can be freely added back to the Track Region by dragging and dropping them back to the Track Region (in much the same way that a new Normal Track is added to the Track Region). This functionality allows users to experiment with combinations of different Tracks, and allows them to easily undo any changes they make regarding the Tracks in the Track Region. No Track can be permanently deleted, preventing unrecoverable user errors, as suggested by Reeves et al. [20].

4.3.5 Recording Dialog

The Recording Dialog appears on top of all other regions immediately after a new Normal Track has been added to the Track Region. When displayed, all other elements of the interface are disabled. It initially contains two Buttons. A cancel Button allows users to cancel the recording, remove the Record Dialog from the display, and re-enable the functionality of the interface. This also removes the newly added Normal Track from the Track Region.

A record Button starts playback of the session from the beginning, and starts recording audio from an instrument given to one of the users. The recording time is always a fixed length, and recording continues until that fixed amount of time has passed. A length of fifteen seconds was chosen for the system. While lengths of thirty and twenty seconds were experimented with, fifteen seconds seems to be long enough to contain most melodies, and short enough that improvising users don't run out of ideas. that users While recording, the interface is completely disabled, and the Recording Dialog displays a countdown of the seconds left in the current recording. The record and cancel Buttons are removed while recording, and replaced by a progress bar, depicting the progress of time through the recording. When record-

ing has finished, the Record Dialog disappears and the interface is re-enabled for user interaction. The new Normal Track is given an alphabetic character identifier, unique from any of the previously recorded Normal Tracks, regardless of whether they reside in the Track Region or the Recycle Bin. Alphabetic identifiers were chosen because they allow for 26 unique single character identifiers. Numeric identifiers were considered, but could be confusing than alphabetic characters when arranged out of order. Figure 4.6 shows the initial Recording Dialog, and the Recording Dialog while a Normal Track is being recorded.

4.4 Summary

The design of the interface for the system presented in this thesis was based on the ideas of simplicity and minimalism, but balances these ideas with enough underlying complexity to provide users with an audio exploration platform with a wide range of sound capabilities. This chapter explored the various features and elements of the interface, and how they were specifically designed for collaboration and use by multiple simultaneous users. The overall design was also compared to the designs of more traditional digital audio environments and traditional DAW conventions. The next chapter presents user studies that test a full system using the described interface, which help to gauge the success of the system's design and implementation.

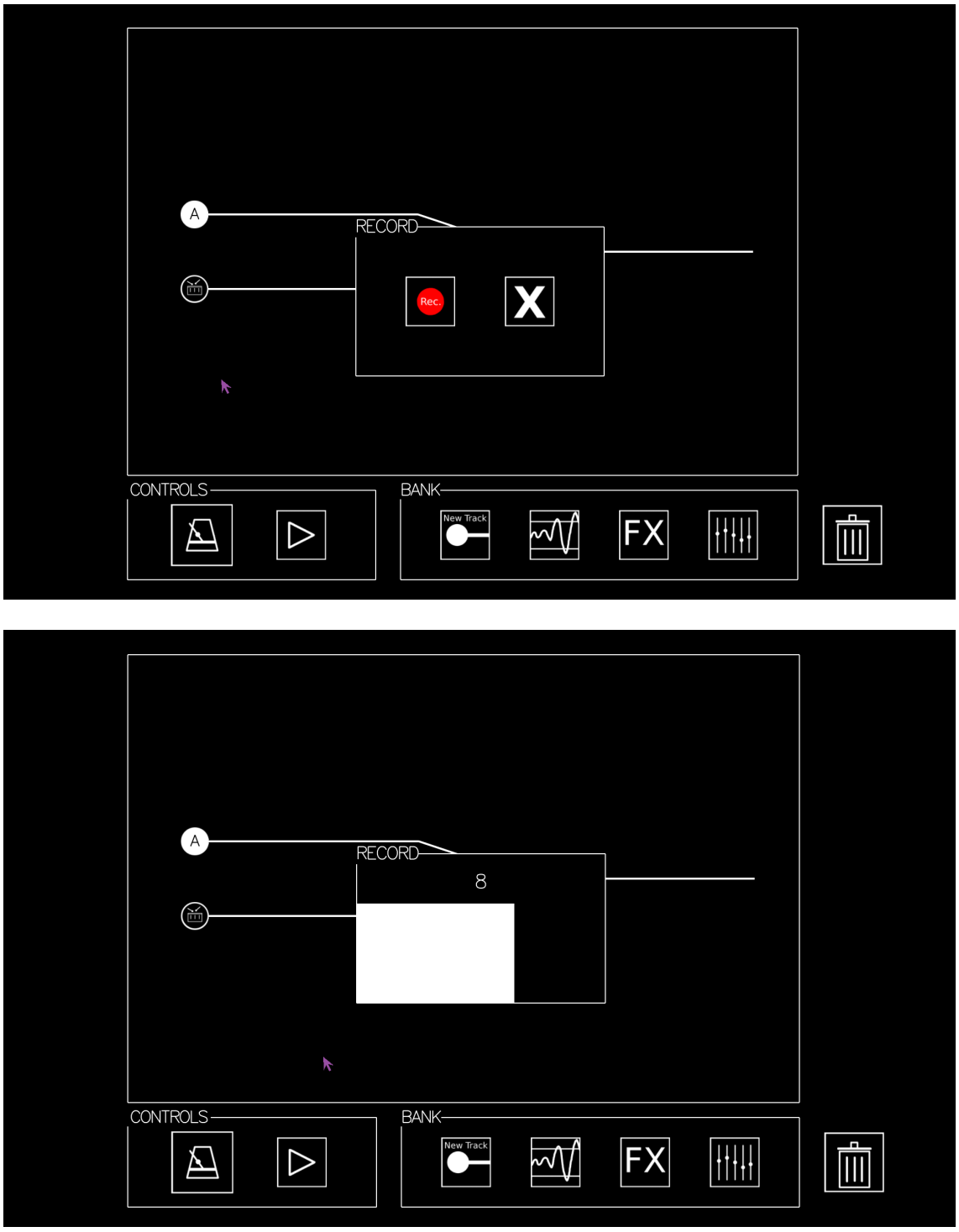

Figure 4.6: The first image is a screen capture of the interface after a new Normal Track has been added, and the Recording Dialog appears. The second image is a screen capture of the interface while recording is in progress. The white block is animated, and grows from left to right, indicating the progress of the recording session.

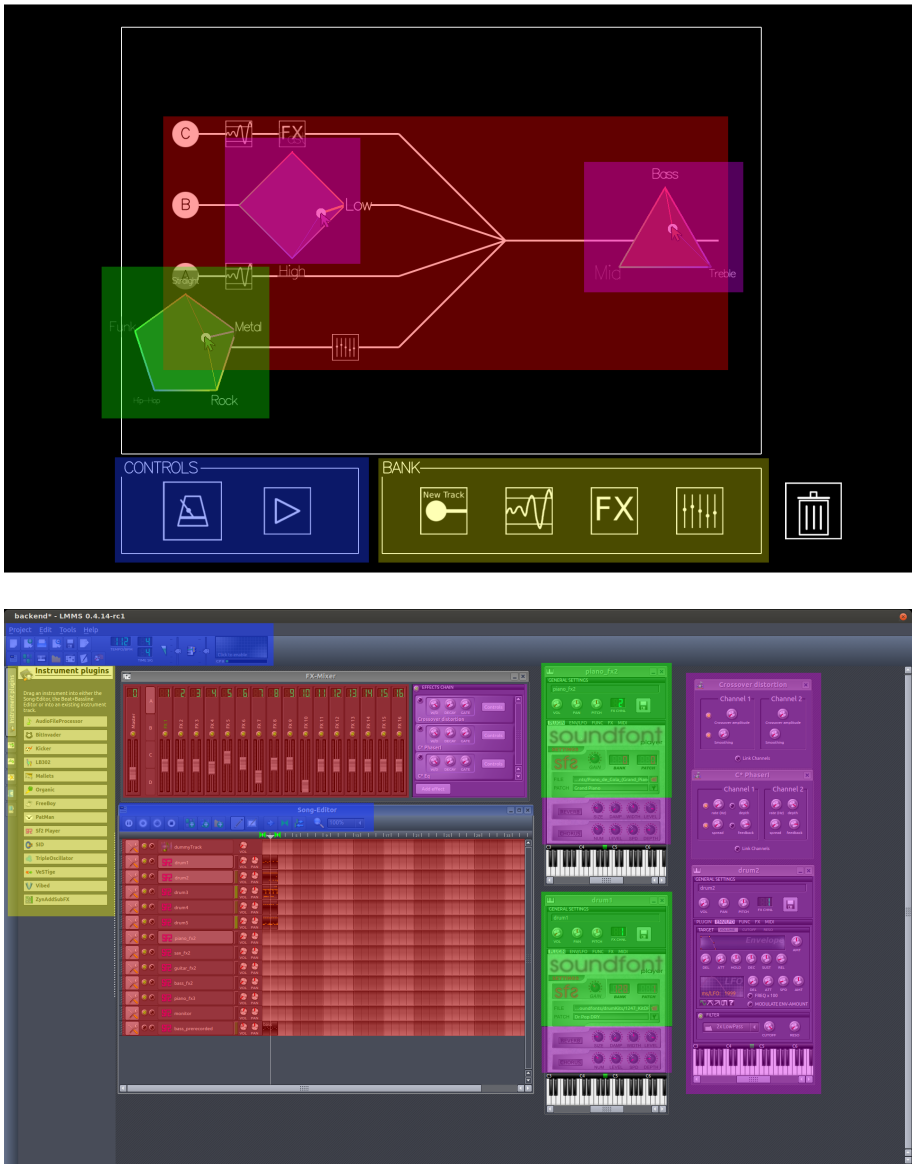

Figure 4.7: These are screen captures of the interface presented in this thesis (top) and the interface of LMMS (bottom) [2]. Highlighted regions of the same color are functionally comparable. Red areas highlight Tracks. Blue areas highlight session controls. Yellow areas highlight banks full of addable features like Effects. Green areas highlight instrument selection and modification areas for Tracks. Purple areas highlight modification areas for Effects. LMMS's sections are separated and include large numbers of complex controls and elements. My system's interface generally contains categorically similar regions, and has fewer and simpler controls and elements in each region.

CHAPTER 5
USER STUDIES

Two different user studies were performed, with two different groups of individuals, to test the audio environment presented in this thesis. The preliminary user study focused on assessing the intuitiveness of the interface and its features. Results of this study were mostly positive, but highlighted specific areas that needed improvement. Revisions to the system were made based on these results, and the back end of the system was optimized to improve its scalability. A second revised user study was then performed with the improved system. This study was broken into a recording phase and an Effects phase, to focus more specifically on these processes. The revised study, unlike the preliminary study, was designed to assess how well users were able to collaborate and make group decisions, and evaluate the behavior and responses of users with different levels of prior experience.

5.1 Study Organization

The user studies for this system were designed for small groups of users. In each group it was required that at least one of the users had some experience with music, and some experience playing instruments. The rest of the users could have any amount of previous experience. A musically experienced user was given a small MIDI keyboard controller with which to record new tracks, but was not given a USB mouse. The other four or five users in the group were given mice, but similarly had no control over instrument input from the MIDI keyboard while recording tracks. This separation of tasks was designed to promote maximum amounts of communication and collaboration between users of the system, though the system could handle other kinds of user configurations. Figure 5.1 shows three users continuing to experiment with the system after the revised user study concluded.

Figure 5.1: This picture was taken immediately after the revised user study. The image shows three of the users making further modifications with the interface, and further experimenting with functionality of PolyButtons.

5.2 Preliminary User Study Design

The first study was designed to determine how intuitive the interface was for new users, and how quickly users were able to understand the functionality of the interface despite very little prior explanation. Users were given very specific tasks to complete as a group. Each of the tasks were strategically designed with underlying goals for specific types of collective and individual user interactions. The rest of this section documents the underlying goals for the study and the strategic instructions given to users in order to accomplish these goals.

For the first study, a single bass guitar track was provided to users at the start. The users were required to add new Tracks on top of the preexisting bass track. The bass track was provided in order to give the users a solid starting place, and lessen potential problems with creative block or bashfulness during Track recording. The

first goal was for users to discover the Drum Track's instrument PolyButton without much direction, and for users to play with the functionality of this button until they understood how it and how PolyButtons in general worked. If needed, the users were given hints about the next step of the process. To accomplish this goal, users were instructed to click the play button and listen to the bass track. Next, they were told to modify and select a drum beat to accompany the bass track.

The next goal was for users to discover not only how to add and record new Tracks to the Track Region, but for users to discover and quickly understand the functionality of the Recycle Bin (at the time called the Trash Can) in relation to Tracks. To achieve this goal, users were told to record a new Track to accompany the bass track and their selected drum beat. Once finished, they were told to record at least three new Tracks, but were allowed to keep only one recorded Track. Additionally, because users were told to choose a single Track, it was hoped that users would learn how to expand the Recycle Bin and drag specific Tracks back to the Track Region from the Recycle Bin.

In the final instruction, users were told to add either a distortion Effect or a modulation Effect (phaser, flanger, chorus, etc.) to this new Track or the Drum Track and modify it with the goal of changing the style of the overall piece. They were told that they must keep only one Effect, but must try both Effects and keep the one they prefer. The goal for this instruction was for users to learn how to add Effects to Tracks, and modify them. Users would hopefully discover the one-user-at-a-time nature of PolyButtons during this process. Similarly, users would need to add at least one Effect to the Recycle Bin to remove it from playback. Users were expected to expand the Recycle Bin to look for old Effects the same way they found removed Tracks, only to find that Effects are not stored in the Recycle Bin. At this time they would hopefully attempt to re-add an Effect to a Track and discover that its settings are saved from the last time it was used on that Track. If users did not discover this on their own, they were made aware of this functionality by the proctor.

5.3 Preliminary User Study Results

This section describes my own observations during the testing periods of the preliminary user study, and provides some of the results and feedback from the exit survey given to each user after the completion of the test period.

5.3.1 Observations

- Users did not find the Drum Track PolyButton on their own. One user requested a larger and more descriptive icon for the PolyButton on the Drum Track.
- The user with the MIDI controller discovered the ability to practice with the instrument without needing to be recording.
- Users found and expanded the Recycle Bin without any help from the proctor.
- Users added new Tracks and recorded them without any help from the proctor.
- After some help from the proctor, users understood that Effect settings were saved, and successfully re-added Effects to a Track.
- Users were confused by the fact that the Drum Track looped continuously despite the fixed length of the recorded Tracks.
- Users talked to each other and collaborated. Often many users experimented until one user figured out the correct way to accomplish some goal.

5.3.2 Exit Survey

This survey was given to users after the all the goals of the session were completed, and the group was happy with their end results. Each user filled out the same survey. The survey included four qualitative written answer questions, and one quantitative user self assessment question where users were expected to place themselves on a relative numeric scale for given categories. Questions were designed to encourage users to comment on the intuitiveness of different features and the overall successes and failures of the test session. The survey also included areas for users to provide ideas for improving the system.

The first question asked users about the intuitiveness of PolyButtons, and to comment on whether they prefer PolyButtons to traditional slider and knob interface elements. Nearly all users found PolyButtons to be intuitive. The second part of the question yielded mixed results. Many users noted that PolyButtons were more simple, and one user noted that they are great for quick experimentation, but a few users also commented on their restrictive nature and lack of full control. One user commented: "More experienced 'musicians' would probably prefer the greater level of control sliders/knobs would give."

The second question asked the users which elements they found the easiest and most difficult to understand. All users found Tracks and the Track Region to be intuitive and easy to understand. "Making a track seemed very easy. I like how [the Tracks] split." Users also found that adding things to the Recycle Bin (then called a Trash Can) was fairly easy and intuitive. The original trash can convention was chosen because of a physical trash can's ability to be opened and closed with a lid, similar to the expansion of the interface element. Many users commented that the Trash Can was confusing because they assumed that it would permanently delete what was added to it. Similarly, users were confused by the fact that they needed to drag a Track to the "trash" just to mute it during playback.

The next question asked users to describe their level of understanding of various system elements and features on a numeric one-to-ten scale. Figure 5.2 provides a graph detailing the average values provided by users. In general, users had a very good understanding of Recording, Tracks, PolyButtons, and multiple mice. Users had less understanding of the Trash Can and Effects. Because of the results of this study, *Recycle Bin* was chosen as the new name for the *Trash Can* in the hope that users would better understand that it was not meant for permanent deletion, but rather meant for temporary removal.

Users were asked to comment on the advantages and disadvantages of having multiple simultaneous users, and whether or not collaboration was enhanced or hindered as a result. User opinions were varied. Some users thought that collaboration was enhanced because there weren't any defined leaders, and other users though that some individuals took charge while others remained more passive. "When par-

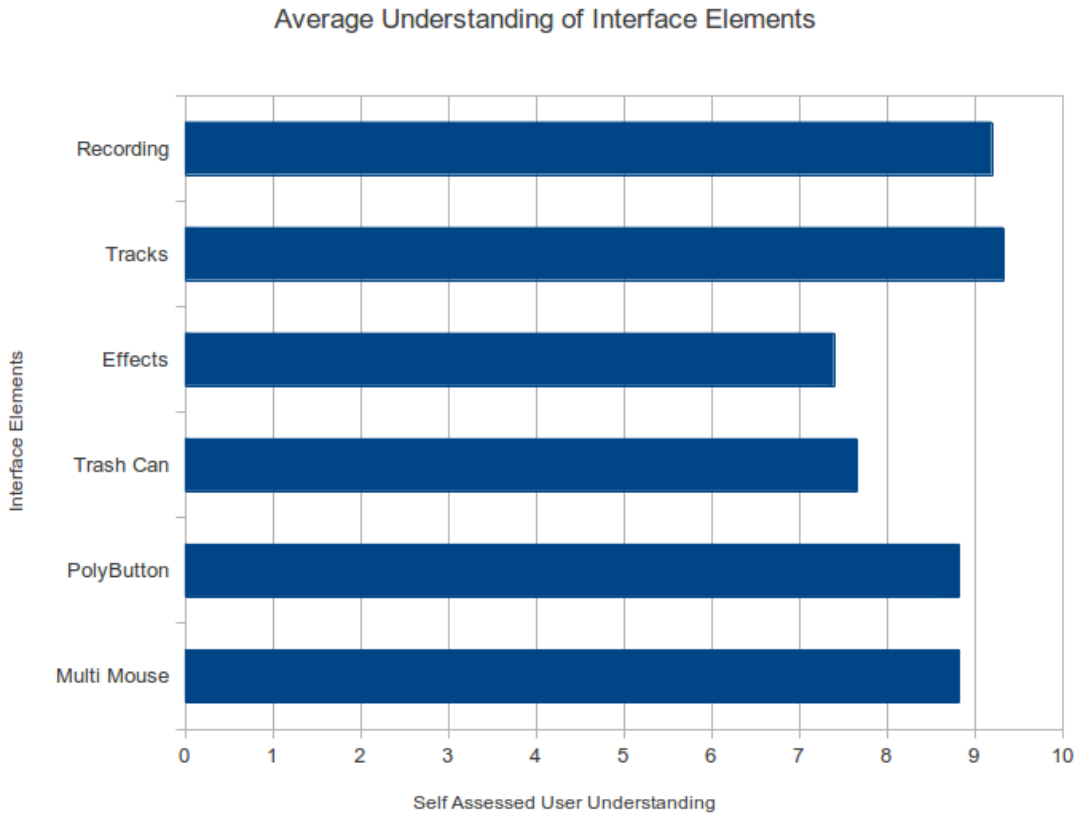

Figure 5.2: This graph depicts the average of all the users' self assessed understandings of particular interface elements after having used the system in the first user study. The scale of understanding ranged from 1 (no understanding) to 10 (understanding well enough to explain to others). This graph shows that, on average, users had a very good understanding of most of the interface elements. Users only had a slightly harder time understanding Effects and the Trash Can.

ticipants collaborate mentally first (i.e. agree on what they want to do), multiple mice work well. When they disagree, one person could undo the work of another." One user commented on the benefits of the simplicity of the input devices. "Because of the interaction being mostly single click, multiple mice on screen was not confusing." Some thought that having multiple users was beneficial because of its convenience, while others questioned the limit of interface features that could be interacted with simultaneously.

The final question asked users to explain how they would change the system

to improve the amount of collaboration between users. Some suggestions included giving users more specific roles, adding a voting mechanism, and removing the single user ownership properties of PolyButtons.

The following is a list of some of the replies users gave in the additional feedback section.

- "liked the simplicity of the design"
- "I felt like I could understand this pretty thoroughly without spending a ton of time figuring it out"
- "Interface almost too 'simple' to justify/need multi-user capability"
- "The Trash Can suggests it's something you want to get rid of entirely... Maybe even change the trash to a recycle bin"
- "[add] ability to save effects in Trash"
- "add a 'tutorial mode' where you have an animated (pre recorded) mouse move"

5.3.3 Preliminary User Study Conclusions

Users overall liked the simplicity of the interface design, and noted that functionality could be understood quickly. All users responded that PolyButtons were intuitive tools, and were easy to use and understand with little to no prior explanation or demonstration. PolyButtons were well liked for the most part, but some users picked up on the parametric restrictions inherent in the design of PolyButtons. Most other elements of the interface did not need prior explanation with the exceptions of the Trash Can and Effects. The Trash Can confused users because they associated it with permanent deletion, and re-adding Effects confused users because they did not respond to being dragged to the Trash Can in the same ways that Tracks did.

5.4 Changes and Optimizations After Preliminary User Study

As a result of the preliminary user study, and the feedback from the study, not only were revisions made to the design of the user study, but changes were made to different aspects of the system itself.

The Trash Can feature was given the name *Recycle Bin* to reflect its metaphoric properties of re-use as opposed to a Trash Can's permanent deletion implications. The sound back end for the system was also optimized in many ways to overcome some of the MIDI limitations faced during the preliminary study. This allowed for greater scalability in terms of the numbers of total record-able Tracks possible in the system, and the numbers of total functional Effects possible in the system. After optimization, it was possible to use four functional record-able Tracks, or use twelve Effects simultaneously, though not at the same time and with the same back end.

5.5 Revised User Study Design

The improved system in this study allowed for the use of more features and tracks. The test was divided into two different phases. The first recording Pphase of the test focused on recording new Tracks and modifying instruments. Users were not given any pre-recorded Tracks, and were not able to use any of the Effects. The second effects phase of the test focused specifically on Effects. Users were given three pre-existing Tracks and a pre-existing drum beat. They were not able or remove or add any Tracks or modify instruments. They were able to add, modify, and remove any Effect to any pre-existing Track. This study included more initial explanations and demonstrations of the interface and its features than the preliminary study. This study was focused less on initial understanding and intuition with interface elements, and was focused more on user collaboration, group behavior, and the opinions of users with different levels of experience.

5.5.1 Recording Phase Goals and Design

The goals for the recording phase were that users understand the basic functionality of Tracks, and understand the processes for modifying Track handle Poly-Buttons, recording new Tracks, adding and removing Tracks, and re-adding Tracks.

Users were expected to record at least a total of six Tracks, a realistic number for composing and experimenting. Each user was also expected to use at least one PolyButton, and use the Recycle Bin with at least one Track.

In this phase users were tasked with creating two pieces. First, they were guided through the rhythm selection process, and then instructed to choose a new rhythm. They were then shown the process of recording a new track, and told to record a new Track. Once recorded, they were shown how to modify the instrument for that track, and how to remove it and add it back to the Track Region using the Recycle Bin. They were then told to record at least three tracks and keep most four tracks to create a piece based on the initial rhythm they chose. This instruction was meant to familiarize users with the editing and recording environment, and how each feature therein functioned.

Users were then instructed to significantly change the rhythm. Once changed they were instructed to change the existing tracks or record new tracks (both preferably) in order to fit the style/genre of the new drum beat. The users were instructed to stop when all users were satisfied with the results. This instruction was given to encourage the maximum amount of track recording, removing, and re-adding, and to demonstrate to users the creative potential of the interface.

5.5.2 Effects Phase Goals and Design

The goals for the Effects phase were that users understand the basic functionality of Effects, and understand the processes for modifying Effects, adding Effects to Tracks, removing Effects from Tracks, and re-adding Effects from the Bank Region back to Tracks. Each user was expected to modify at least one Effect, add one Effect to a Track, and remove one Effect from a Track. At least one user was expected to re-add a previously removed Effect to a Track from the Bank Region. This was not expected of every user because it is potentially confusing, and the other types of interaction are much more common.

In this phase users were instructed to press the play button and listen to the original recording for as long as they needed to get familiar with it. Users were told that they were not be allowed to modify the preexisting tracks by changing their

instruments or using the Recycle Bin with these tracks. Users were then tasked with modifying the existing tracks using Effects so that the piece can be defined by a new genre (as opposed to the starting genre of Jazz). They were then shown how to add, modify, and remove effects from a track. They were also told the properties of re-adding effects to a track. This instruction is designed to familiarize users with the different kinds of Effects available to them, and introduce them to Effect functionality.

Once satisfied, they were be instructed to make the piece as unrecognizable from its original form as possible. This instruction was intended to encourage all users to add, modify, remove, and re-add as many different Effects as possible. It is also meant to demonstrate the varieties of sound modification available, and the breadth of the sound modifications possible with the interface.

5.6 Revised User Study Results

This section describes my own observations during the testing phases of the revised user study, and provides some of the results and feedback from the exit surveys given to each user after the completion of the test period.

5.6.1 Observations

- With an initial demonstration for some of the interface features, users very quickly understood the interface functionality, and needed little other explanation.
- Users occasionally had a hard time keeping track of their own mouse cursor on the display.
- Users frequently and effectively communicated with each other verbally.
- There was some confusion during recording as to the identity of particular tracks. Users' inability to rename or re-identify Tracks may have been the cause.
- Users requested the ability to modify the instrument parameter for the monitored keyboard before recording, rather than just after recording was finished.

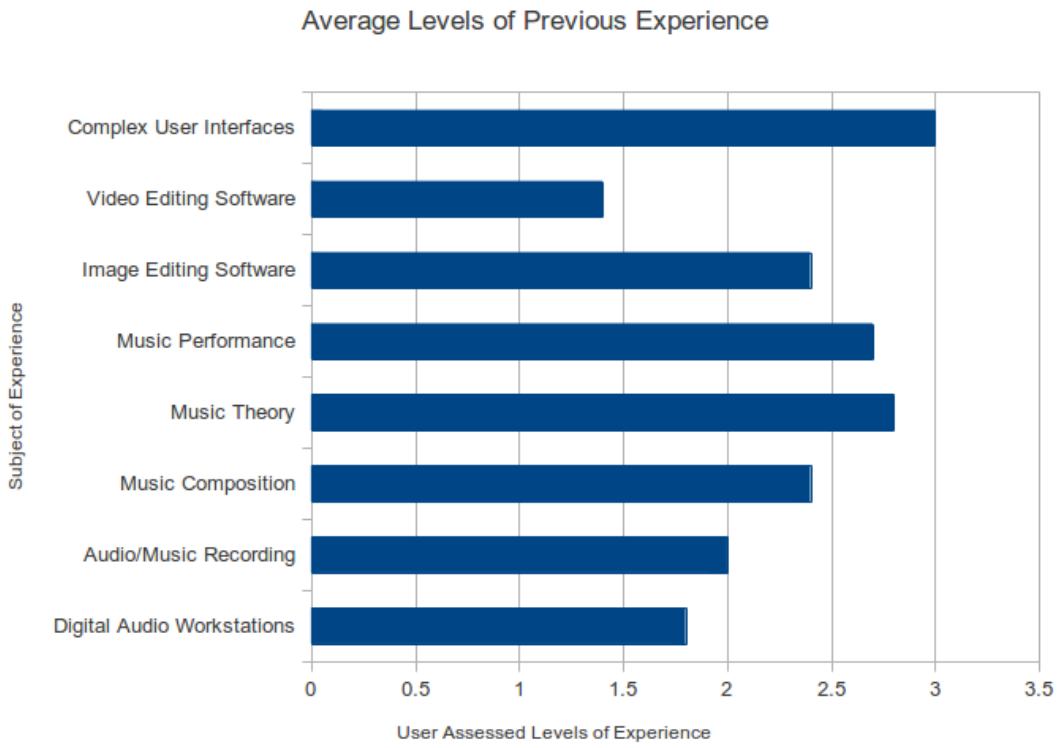

Figure 5.3: This graph depicts the average of all the users' self assessed experience levels in different music and software categories in the second user study. The scale of experience ranged from 1 (no experience) to 5 (highly experienced).

5.6.2 Exit Survey

The exit survey for the second study was divided into two sections. One section asked users about their prior levels of experience in various subjects related to system they used during the study. Users were also asked about specific types of software they had used before, and their software preferences. Each user's survey was uniquely numbered for referencing. The second section of the survey asked users for their feedback on various elements of the interface and system. The feedback was correlated with each user's levels of prior experience.

Feedback Section The first question was focused on the Recording Phase of the test, and asked users to comment on the positives and negatives of the Track recording process, and the process of removing and re-adding Tracks from the Recycle Bin.

Users 2 and 5, the two least experienced users, only commented that the processes and the controls were intuitive. User 3, the user playing the keyboard during recording, and the most experienced user in terms of DAWs and music in general, noted that adding tracks was easy, but wanted the ability to modify the instrument for the monitor before recording. "[I] wish I could record/'improv' with the other instruments as well". User 4, a moderately experienced user, commented that they liked how Tracks preserved their Effect and Instrument parameters. User 1 with similar amounts of experience also liked that Tracks preserved preserved their Effect and Instrument parameters, but desired the ability to dynamically rename or relabel a Track.

The second question was focused on the Effects Phase of the test, and asked users to comment on the positives and negatives of the process of adding, modifying, removing, and re-adding Effects to tracks. The more inexperienced users, 2 and 5, had trouble distinguishing and understanding the modifications that they made to Effects. User 2 explained that the labels alone were not enough to predict and understand the characteristics of different Effects and parameters. However, user 2 commented that the simplicity of the PolyButtons allowed for fast experimentation, implying that prior knowledge about the verbal descriptions of Effects was less necessary because of the understanding that came from rapidly experimenting. User 5 had issues distinguishing their own changes from the group while other modifications were occurring. More experienced users 1 and 3 agreed that understanding Effects enough to predict the outcome of parameter modifications required experimentation. Moderately experienced user 4 wouldn't have understood that Effects could be added to the Master Track without prior explanation.

The next question involved PolyButtons in general, and asked users to comment on the positives and negatives of using them. User feedback was mostly positive, and few negatives were listed. One negative comment came from experienced user 3. "I think it'd be difficult to incorporate more than 4 options without the sound getting messy." This speaks to the scalability of PolyButtons in terms of their number of parameters. All of the users, experienced or not, found PolyButtons easy to use and intuitive. Less experienced user 2 liked that it was possible to

easily try most parameter configurations, even though that required more limited parameter options.

The following question asked users to compare PolyButtons with interface elements they had used in other software, and name some respective advantages and disadvantages. Only more experienced users 1 and 3 provided feedback. User 3 noted the disadvantage of parameter constraint compared to other interface elements, but also noted that PolyButtons were "more fun and inspired creativity from the get-go." User 1, moderately experienced with image editing software, noted the similarity between PolyButtons and two-dimensional color selectors, stating that the latter is good for getting a general idea of a color rather than a precise RGB value, in the same way the PolyButtons are good for assessing audio ideas, rather than producing precise sounds. The latter comment affirms the strategy of two-dimensional slider input, and the former addresses the issues presented by the parametric constraints with PolyButton's unique few-to-many mappings, offering some arguments in its favor.

The last question asked users how collaboration was enhanced through simultaneous use, and what other advantages and disadvantages there were for simultaneous interactivity. Less experienced user 5 commented that simultaneous interaction made it hard to distinguish their own changes. Less experienced user 2 noted that the group probably arrived at a piece of audio faster than a single individual would have. Experienced user 1 noted that achieving goals was easier with a group working together, but also noticed a lack of organization. "Very little self organization happened between users though the potential existed." Experienced user 4 noted issues, but praised the collaborative aspects of the system. "It was difficult to always keep track of my own cursor, but the collaboration is without a doubt the most enjoyable aspect of this system."

Prior Experience Section The first question was quantitative. It asked users to describe their amount of prior experience in various fields, all related in some way to the system used in the study, from 1 to 5. Figure 5.3 provides a graph detailing the average values provided by users. Figure 5.4 provides a graph detailing the levels

of experience of each individual user. This graph also shows the large differences in experience levels between different users in the same test group. The test group, on average, was most experienced with complex user interfaces, but least experienced with specific media software like DAWs and video editing software. The group had a moderate level of experience with image editing software, and with music performance, music theory, and musical composition. Individually, each user in the test group had very different self assessed levels of prior experience. Some users had no experience with music or specific media editing software, While others were highly experienced with music and with editing software like DAWs.

Users were then asked to list the different kinds of audio software they had experience with (if any), which they prefer, and why they prefer it. Only users 1 and 3 listed more than one piece of audio software. The software they listed included Pro Tools, Audacity, Reason, Sibelius, FL Studio, Logic, Reaper, and others. User 1 preferred Audacity for quick edits, likely because of its relative simplicity. User 3 preferred Logic because of its intuitiveness and its ability to easily sync with Sibelius software. User 4 had only used Audacity, user 5 had only used Sound Studio, and user 2 had never used audio software before.

Users were asked to list any specific software they were experienced with that had a complex feature-rich user interface, especially in the fields of digital media or editing. Users 1, 4, and 5, a group with varied experience levels in terms of music and audio software, listed experience with Adobe Photoshop and user 1 listed the entire Adobe Suite. Less experienced users 2 and 5 listed Xcode and Keynote respectively. Very experienced user 3 did not list any other pieces of software, but had listed many pieces of audio software in the previous question.

5.6.3 Revised User Study Conclusions

In general, users felt that PolyButtons were an intuitive and powerful tool regardless of their experience levels. Users with more musical experience and more experience with editing and media software commented about PolyButton's parametric constraints in comparison to other interface elements, but some provided arguments in favor of this. More inexperienced users had more difficulty under-

standing the changes that were made to the sound when Effects were modified, and had a harder time distinguishing their individual modifications. Users with varied experience levels mentioned that the collaborative aspects of simultaneous interaction allowed for fast and easy achievement of goals, but that simultaneous interaction can cause confusion and ambiguity.

5.7 Summary and Conclusions

This chapter presented two different types of user studies performed with the intention of evaluating the success of the system through user observations and written user feedback. The preliminary study focused on intuitiveness, while the revised study focused on collaboration and group behavior, correlating users responses with their individual prior experience levels. The results of the studies affirmed that PolyButtons were intuitive, easy to use, and allowed for fast experimentation. The results also showed that most of the interface elements were not only intuitive, but promoted user collaboration by effectively interfacing with PolyMice. In the time between the two user studies discussed here, the implementation of the system, and specifically the sound back end, was optimized to allow for a greater number of functional interface elements without severe latency issues or MIDI messsage dropping. Those optimizations, along with other important system implementation details, are described in the next chapter.

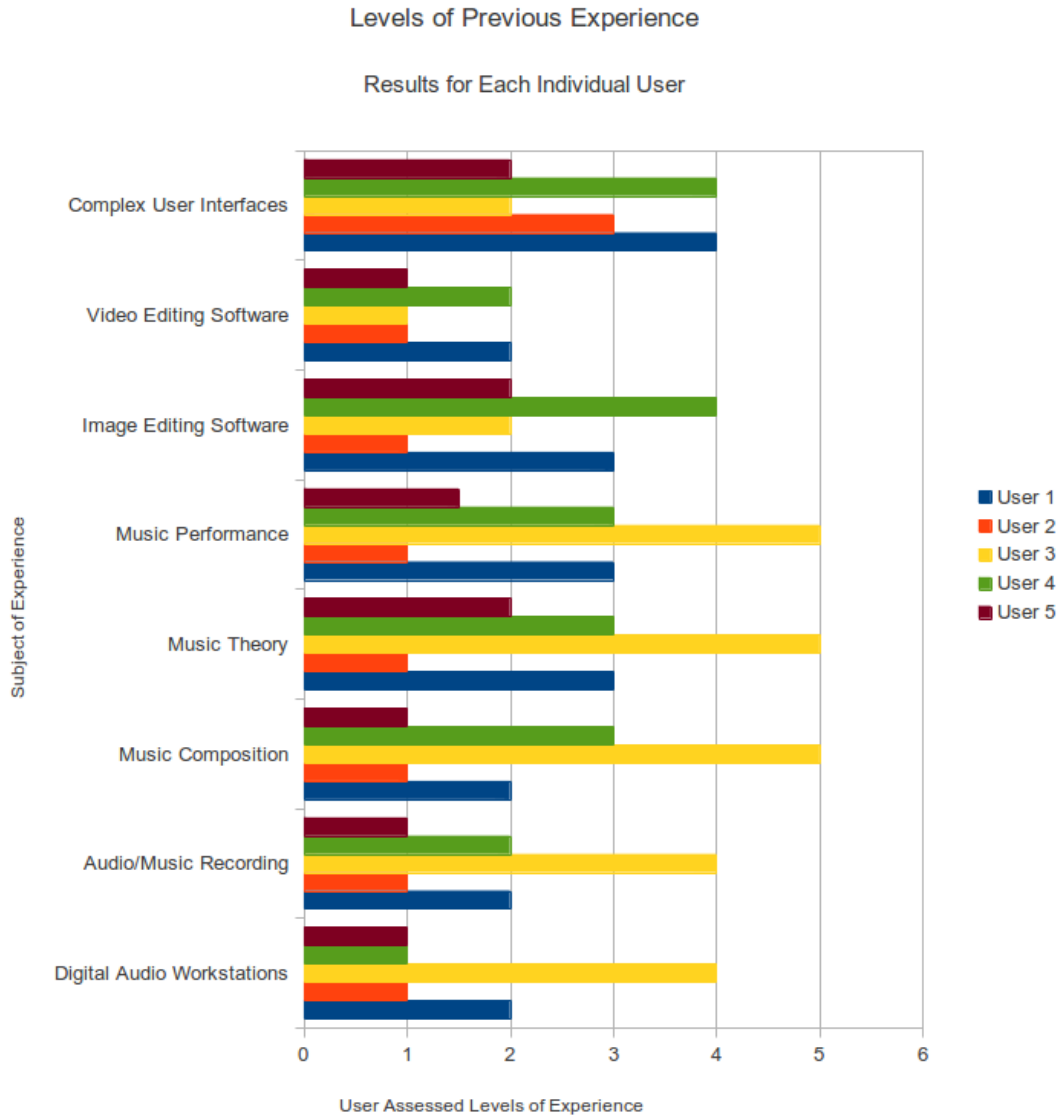

Figure 5.4: This graph depicts each of the user's self assessed experience levels in different music and software categories for one session in the second user study. The scale of experience ranged from 1 (no experience) to 5 (highly experienced). Note the wide range of experience levels in the group.

CHAPTER 6
SYSTEM IMPLEMENTATION

This chapter provides details about the three major aspects of the system's implementation. It describes the PolyMouse interface, a novel example of single display groupware for USB mice. It also describes important elements of the system's interface implementation, and describes the connections to the sound back end, and some of the design and implementation challenges therein. Specific optimizations for the system, especially with regards to the sound back end, are also explained. The system diagram for the PolyMouse interface can be seen in Figure 6.2, and the system diagram for the entire audio environment can be seen in Figure 6.1.

6.1 Simultaneous Mouse Input

The *PolyMouse* software interface has been implemented for Ubuntu Linux systems, and utilizes the `udev` daemon and the ability to make custom rules files. These custom rules are capable of overriding many different functions of kernel behavior. Valentino et al. describe methods for creating custom `udev` rules for devices like USB flash drives, and use the term *intelligent peripheral controller* (IPC) to describe such devices [24]. In this case, the rules file overrides the functionality of specific USB mice based on their *name* attribute as read by the operating system's device manager. Once overridden, a specific USB mouse is given a new custom device file in the `dev` directory. The relative actions of mice and other types of overwritten hardware are written to the new device file.

A mouse reading listener application written in C++ was created to read the new specific mouse device files using `fread`. Because the `fread` system call in Linux is necessarily blocking, Pthreads were used to create a distinct listener thread for each of the PolyMice. Once created each thread listens to its specific device file until the listener application is terminated. Once relative mouse position and state information is read from the device file, it communicates with other applications via shared data log files. When each listener thread writes to a log file, a Pthread mutex

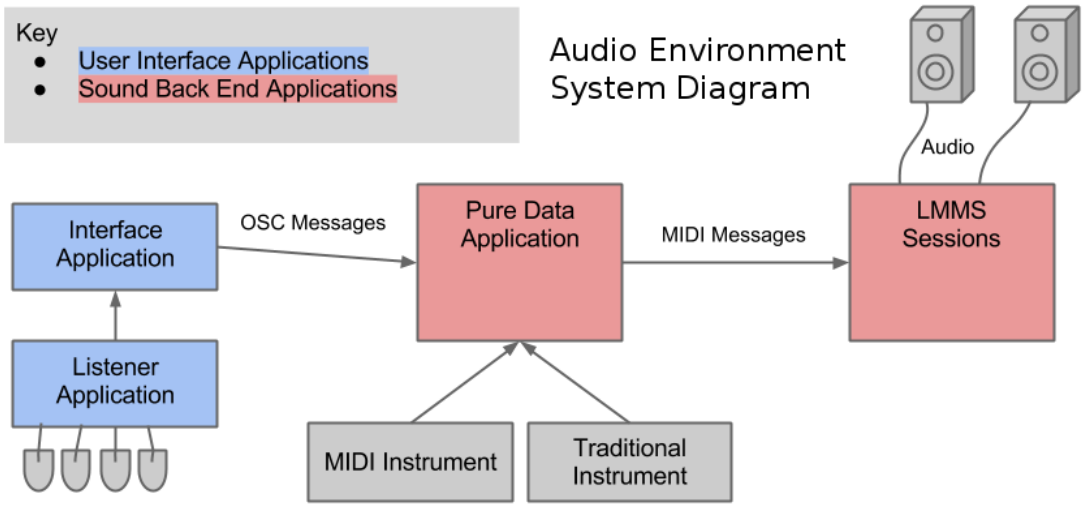

Figure 6.1: This system diagram shows the implementation details for the entire audio environment. It provides distinctions between the front facing interface applications, and sound back end applications. Communication details and individual application responsibilities are also depicted. Notice the possibility to record with traditional instruments as well as MIDI controllers.

is acquired before and released after the write is performed. This ensures that no log file can be written to by multiple threads simultaneously, and log data will not be corrupted.

A shared data logging class handles the specifics of writing to and reading from these log files, and maintains them. Once one thousand log files have been created, the oldest log files are replaced with newer ones, in order to ensure that memory is used efficiently. Similarly, log files can only reach a particular size for the sake of memory efficiency and read and write speeds with log files. This logging mechanism ensures that each piece of input data read from a device is made available to other applications, even though the data isn't permanently available. Different caps on the number of total log files were experimented with, but setting the cap at one thousand log files, with my system, ensured that all mouse data was accessed by the interface application before it was overwritten. Other applications using the PolyMouse interface utilize USB keyboards for simultaneous input as well as

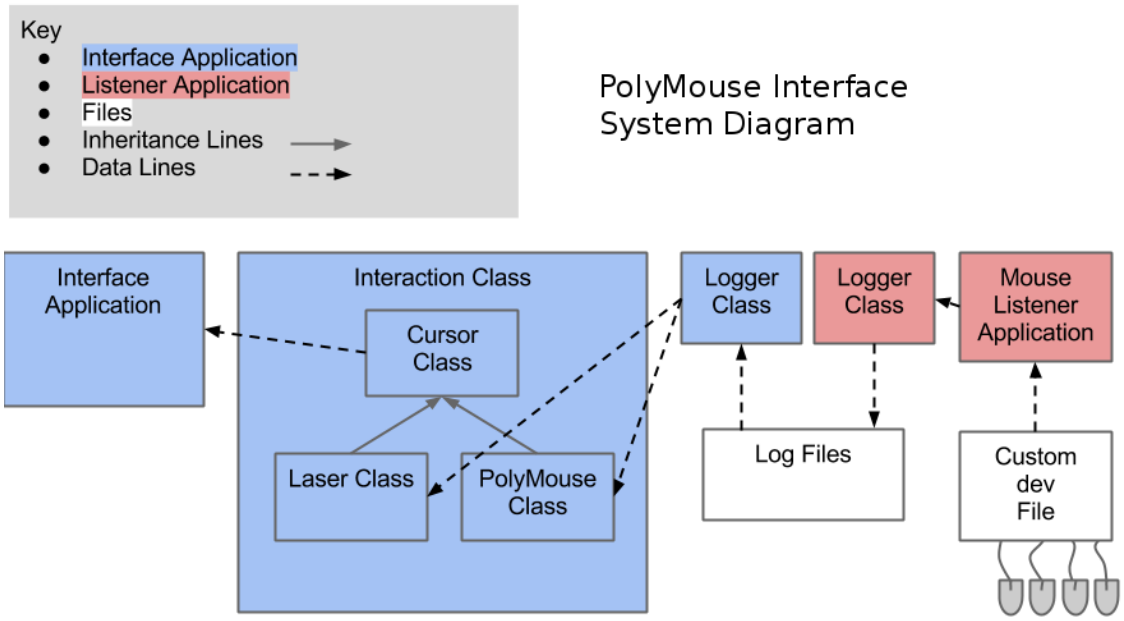

Figure 6.2: This system diagram shows the implementation details for the PolyMouse interface. It provides clear distinctions for the two main applications involved, and uses dashed arrows to represent the transportation of data from the USB mice to the interaction callbacks in the interface application. Notice the lines of inheritence from PolyMouse to Cursor, and the other input device class (Laser) derived from the Cursor class.

mice. The logging class also provides global public variables that store position information about the individual mice. This position data is stored, and can be modified by incoming relative mouse motion data.

Another shared class handles interactions. This *Interaction* class uses the information stored in the global state variables in the logger class. A *Cursor* class is used as a base class for human input media like mice, and has member variables like *up*, *down*, and *position*. A child *PolyMouse* class inherits from the Cursor class, and implements specific methods for gestures like clicking and movement. Such a design scheme is useful because other human input devices can be used the in same system by extending the Cursor class. The current Interaction class stores a specific PolyMouse object for each USB mouse currently in use. The Interaction class also handles the drawing of individual uniquely colored cursors on the display

using OpenGL. When a Cursor object is in a *down* state, its border is made brighter to visually represent its state to other users of the system.

The greatest advantage of the Interaction and Cursor classes is that they allow the main user interface application to operate without any prior information about the specifics of the human input media being used. The main user interface, written in C++, simply registers its callback functions with the Interaction class, and the Interaction class calls the necessary callback functions, passing in a pointer to the specific Cursor object involved. The user interface application therefore only needs to concern itself with Cursor objects, and is much more flexible and extensible as a result of this layer of abstraction.

6.2 Interface Implementation

The design of the interface discussed in Chapter 4 required specific functionality in its implementation. The interface needed to be capable of fast parallel interactions, and it needed to be extensible and flexible enough to handle small frequent changes and handle frequent and sometimes drastic changes in the sound back end. The implementation of the PolyButton, Track, and Track Region classes will be discussed in greater detail.

6.2.1 PolyButtons

The PolyButton class is an extension of the Button class. A PolyButton object contains a Button object for its initial collapsed state, and a *PolyInterface* object for its expanded state. The collapsed Button renders a trivial custom icon image. When clicked, the collapsed Button is hidden, and the PolyInterface is rendered. Once clicked and expanded, the ID of the owner Cursor object is stored in the PolyButton class, ensuring that only it may interact with the PolyInterface, or collapse and release the PolyButton by clicking outside of the PolyInterface region. Background color, taken from the owner Cursor and made slightly darker, is also stored in the PolyButton class.

6.2.2 Tracks

All Tracks are Normal Tracks unless initialized as either the Drum Track or the Master Track, and the rules for these particular Tracks are different than Normal Tracks. Each Track aside from the Master Track contains four PolyButton pointers. The first points to the handle PolyButton, which modifies the instrument of a Normal Track, or the rhythm of the Drum Track. The other three PolyButtons are Effect PolyButtons. The Track stores boolean values to determine whether or not a particular Effect is active, and Tracks are initialized with no active Effects. This implementation is fast because it does not inefficiently delete or reconstruct Effect instances, and it allows for parameter persistence even when an Effect is not active. The Track contains one Effect object for each of the 3 available Effects; distortion, modulation, and equalization.

6.2.3 Track Region

The Track Region class contains a list of pointers to all the Normal Tracks currently included in the audio playback. The Track Region always contains a pointer to the Drum Track and the Master Track. It handles the organization and rendering of Tracks on the display. It provides a vertical arrangement method that appropriately arranges all of the current Normal Tracks and the Drum Track in a vertical pattern with even spacing in the vertical center of the Track Region.

In the event that a PolyButton from the Track Region is released somewhere outside of the Recycle Bin, it will result in the PolyButton floating back to its original location, and return to being fully rendered. The floating animations are handled in separate Pthreads so they can be rendered asynchronously with other interface and Cursor renderings, while remaining within the Track Region's callback method. Furthermore, these animations do not need to occur in a higher level `glutMainLoop` callback where things like Cursor rendering takes place. Similarly implemented animations occur when dragging Tracks out of the Recycle Bin.

6.2.4 Interface Connection With Sound Back End

One way communication from the interface to the audio back end applications occurs with Open Sound Control (OSC) messages. OSC is a standardized data format that provides a wrapper around messages with traditional User Datagram Protocol (UDP) sockets. The interface sends messages over localhost to applications on the same machine, but could send messages to back ends on other networked machines. A callback in the `glutMainLoop` handles the packaging and sending of OSC messages to the back end. Most of the information sent to the back end involves Tracks. It is important for the back end to know which recorded Tracks are active, which of their Effects are active, and to know the parameter values of their handle PolyButton, and Effects PolyButtons.

There are some message types that signal starting and stopping playback and recording sessions, but most OSC messages, packaged in the standardized string format, include information about a particular Track in the Track Region, and include the Track ID, the Track's handle PolyButton parameter values, the Track's active Effects, and each of those Effect PolyButton's parameter values. This tends to quickly amount to a large amount of data, especially when there are many active Tracks. Therefore, checksums are used with each Track, to quickly assess whether or not any of its active Effects or parameter values have changed since the iteration through the `glutMainLoop`. The checksum function adds all of the Track PolyButton's parameter values together. If the Track has not changed, then an OSC message about it will not be sent to the back end. This greatly reduces the amount of messages sent, and the amount of network bandwidth used. The maximum number of messages that can be sent simultaneously is determined by the number of Cursors on screen, each capable of modifying a different Track. Before the optimization, the interface always sent the maximum number of messages, which was instead determined by the (potentially large) number of Tracks in the Track Region.

Because the interface sends out OSC messages, a well known standard for communication with existing audio applications, the interface is very flexible, and a back end can be created using existing tools and audio applications relatively quickly. It can also be modified easily and frequently, without having to modify or

extend the interface.

6.3 Sound Back End Implementation

The back end created for this system was created using an extension of Pure Data called Pd-extended, and Linux MultiMedia Studio (LMMS). Pd receives and parses the OSC messages, handles recording with MIDI, and uses MIDI message to control custom sessions of LMMS.

6.3.1 Pure Data

The main back end application was created using the Pure Data visual programming environment. Pure Data was chosen over Max/MSP because it runs in Ubuntu Linux, and is free and open sourced. Pure Data objects will be notated with brackets, as such: `[object]`. The application first uses the `[route]` and `[unpack]` objects to parse incoming OSC messages from the front end, and these one way messages are intercepted, without a handshake, using the Pd-extended object `[udpreceive]`.

The application interfaces with a USB MIDI controller keyboard. When the back end is signaled to start recording, the `[seq]` and `[midiparse]` objects are used to record and playback sequences of MIDI notes as `.mid` files in the `/tmp` directory. The application is also capable of interfacing with traditional instruments like an electric guitar, or any acoustic instrument recorded with a traditional microphone using the `[fiddle~]` object to convert audio input into MIDI messages. This object is capable of taking direct audio input, performing Fourier spectral analysis on that input, and quickly outputting in MIDI note format its best guesses as to the pitch of the notes played in the audio input. The `[fiddle~]` object in practice can only decipher one note at a time with any consistency. Also, the `[fiddle~]` object struggled to decipher single notes being played in quick succession. The sound back end was tested with an electric guitar, but because of the resulting latency, inaccuracy, and inability to read more than one note at a time, a MIDI controller was used as a more stable alternative.

The Pure Data application does not produce any sound itself. Instead, it

sends MIDI messages to a custom LMMS session, which produces sound. LMMS was chosen because of its extensive effect and virtual instrument capabilities. These capabilities allowed for a broader set of prototype options and greater prototype flexibility. During playback, the Pd application uses the `[noteout]` object to send MIDI note messages to LMMS, and uses the `[ctlout]` object to control various one-to-one sliders and knobs in LMMS. Because MIDI messages are not able to dynamically control things in LMMS like the number of tracks in a session, etc., there is a hard-coded limit to the number of *record-able* Tracks for the system at any given time. MIDI messages also have technical limitations described by Rothstein [5]. MIDI messages travel slowly compared to other types of network packets, and are processed serially. This can cause latency issues in systems that use many simultaneous MIDI messages called *MIDI lag*. In practice, I found that no more than fifteen simultaneous MIDI messages could be sent reliably without MIDI lag problems.

Certain optimizations were made as a response to the limitations in the MIDI standard. A `[change]` object monitors each `[ctlout]` MIDI message, and ensures that no message is sent unless its value is different than the last one sent. The MIDI files for the Drum Track are also stored directly in the LMMS session, to reduce the MIDI note message congestion. Despite this, there are still system limitations due to the MIDI messages. In practice, if approximately 25 or more MIDI connections are active on the machine, recording latencies become very high. Therefore, the current system is limited to four record-able tracks.

Because Drum Tracks were added directly to the LMMS sessions, it became necessary to control the playback function of the LMMS application. But, MIDI messages cannot control playback in LMMS. Therefore, in order to control playback in LMMS from Pure Data, the `[shell]` object is used to send a command-line message to a system shell. The shell message invokes the command-line application xdotool, an X11 automation tool, and sends a virtual space bar key press signal to the window system after setting LMMS as the active window. The space bar is a keyboard shortcut in LMMS that starts and stops playback of the LMMS session. The `[shell]` object is also used to send a command-line message to delete all of

the recorded MIDI files in the `/tmp` when the system is reset.

6.3.2 Linux MultiMedia Studio

The LMMS sessions created for this back end contain one virtual instrument track for each parameter of a Track's instrument PolyButton, for a total of four record-able Tracks in the interface. Free virtual instruments were downloaded for this system in the SoundFont `sf2` format. If each Track in the interface has an instrument PolyButton with four parameters, this amounts to sixteen virtual instrument tracks in the LMMS session. LMMS tracks providing sound for the same interface Track share the same Effects bus, thereby reducing the total number of LMMS (LADSPA) effects, and MIDI controller connections from Pd needed. The same is true for the Drum Track and its 5 rhythms stored as tracks in the LMMS session. When an instrument (or rhythm) PolyButton is modified and its parameter values are changed, the volume parameters of each corresponding LMMS track is changed. Specific rhythms were created that all follow the same time signature, and can all be played concurrently and in sync with each other. Effect PolyButtons control various parameters in a specific LMMS (LADSPA) effect for a Track.

6.4 System Limitations

The system used in the first user study was unoptimized and significantly limited in the numbers of Tracks and Effects that were possible in the back end. Despite this, the interface did not need to change in any way to accommodate this limitation. The back end was optimized for the second user study in ways mentioned previously. This optimization was able to increase the number of record-able Tracks and Effects before problems with recording latency became significant. But, even after implementing the system optimizations, limitations still existed.

The second user study was broken into two phases, each phase using a slightly different sound back end with different MIDI connections. The first allowed for four record-able Tracks with fully functional handle (instrument or rhythm) PolyButtons and no Effects PolyButtons. The second involved prerecorded Tracks and non-functional handle PolyButtons, but fully functional Effects PolyButtons for each of

the Tracks.

Because LMMS does not support MIDI message control of the tempo for session, I could not make a back end that functionally connected to the metronome element of the interface. This element, designed as a PolyButton with only two parameters, is functional in the interface, but does not control the tempo of the resulting sound.

6.5 Summary and Future Work

This chapter has discussed the implementation of the PolyMouse interface, some elements of the interface for the audio environment, and the implementation details for the sound back end. Included were descriptions of many of the system's limitations, and the optimizations that were designed to improve upon these limits.

If I were to continue working on the system, I would no longer use MIDI messages to control an LMMS session because of the limitations within the MIDI standard. I would also choose not to use LMMS sessions for audio playback as it places limits on the number of record-able Track possible in the system. Rather, I would choose to create a more extensive Pure Data application with the ability to produce audio, or create a custom audio production application in a traditional programming language like C++. In either case, it would not be necessary to alter the interface of the system. I would also more extensively test the scalability of the interface elements, as well as the scalability of PolyMice.

CHAPTER 7
CONCLUSION AND DISCUSSION

This thesis has documented the design and implementation of a new type of collaborative digital audio environment for musical experimentation and exploration rather than polished digital audio production. Such software gives a broad range of individuals a simple interface with a gentle learning curve for a system that can be used as both a compositional scratch pad and an audio exploration and learning platform.

The system was designed for multiple simultaneous users via multiple USB mice and multiple mouse cursors on a single display. Many interface elements were designed for collaboration between the simultaneous users. The interface was designed to be simple enough for users with no musical or audio software experience, but powerful enough to allow users to experiment with and explore a wide range of audio possibilities in interactive time. The system's interface and the back end audio applications for the system were created to be independent of each other, flexible, and extensible. Two user studies, specifically designed for groups with varied levels of experience, aided in the evaluation of different aspects of the system.

7.1 Presented Contributions

This thesis has presented the PolyMouse interface, a novel implementation of *single display groupware* by creating multiple custom cursors on a single display, and controlling each cursor with a specific USB mouse. This multi-user environment aspect encourages collaboration between users, and allows less experienced users to learn from those with more experience.

This thesis has presented PolyButton interface elements. These multiparametric two dimensional sliders use the vertices of polygonal shapes to represent sound parameters that are categorically similar. They allow users to make complex changes very easily and quickly, favoring user experimentation rather than precise decision making. Users in two separate users studies found these interface elements

to be very intuitive, and were able to quickly explore sound possibilities to learn and to accomplish goals.

This thesis presents the results of a preliminary, and revised user study, each designed to test different aspects of the audio environment. Users verbally communicated and collaborated with each other throughout each of the studies, and commented that elements of the interface design like multi-user interaction and PolyButtons aided them collaboratively. Users liked the simplicity of the design and found the interface and its elements easy to use and understand quickly.

7.2 Discussion

The audio environment created for and presented in this thesis includes a wide range of users in its target demographic, but has attempted to fill a more specific categorical gap in the digital audio software industry. The results of users studies and feedback about this audio environment indicate its success in bringing together groups with different experience levels to creatively explore audio production and musical composition. The simple and powerful PolyButton interface elements presented in this thesis promoted collaboration in a multi-user sound environment, but could easily become an effective collaborative tool for other types of software.

References

[1] Avid Technology, Inc. (2013) Avid | pro tools family. [Online]. Available: http://www.avid.com/US/products/family/pro-tools [Jun. 15, 2013].

[2] T. Doerffel. (2013, Jun.) *LMMS - Linux MultiMedia Studio*. [Online]. Available: http://lmms.sourceforge.net/home.php [Jun. 15, 2013].

[3] D. Franz, *Producing in the Home Studio with Pro Tools*. Boston, MA: Berklee Press Publications, 2001.

[4] Audacity Team. (2013) Audacity: Free audio editor and recorder. [Online]. Available: http://audacity.sourceforge.net/ [Jun. 15, 2013].

[5] J. Rothstein, *MIDI: A Comprehensive Introduction*. Madison, WI: A-R Editions, Inc., 1995, vol. 7.

[6] D. Timis and D. G. Willenbrink, "Method and system for editing digital audio information with music-like parameters," U.S. Patent 5,792,971, Aug. 11, 1998.

[7] Ableton AG. (2013) What is live? learn more about ableton's music making software | ableton. [Online]. Available: https://www.ableton.com/en/live/ [Jun. 15, 2013].

[8] K. Robinson, *Ableton Live 8 and Suite 8: Create, Produce, Perform*. Burlington, MA: Focal Press, 2009.

[9] S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner, "The reactable: Exploring the synergy between live music performance and tabletop tangible interfaces," in *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, Baton Rouge, LA, 2007, pp. 139–146.

[10] T. Blaine and T. Perkis, "The jam-o-drum interactive music system: A study in interaction design," in *Proceedings of the 3rd Conference on Designing*

Interactive Systems: Processes, Practices, Methods, and Techniques, New York, NY, 2000, pp. 165–173.

[11] D. Diakopoulos and A. Kapur, *Argos: An Open Source Application for Building Multi-Touch Musical Interfaces.* Ann Arbor, MI: MPublishing, 2010.

[12] M. Puckette *et al.*, "Pure data: Another integrated computer music environment," in *Proceedings of the Second Intercollege Computer Music Concerts*, Tachikawa, Japan, 1996, pp. 37–41.

[13] Pd-community. (2013) Pure Data - PD Community Site. [Online]. Available: http://puredata.info/ [Jun. 17, 2013].

[14] Cycling '74. (2013) Max is powerful software. « Cycling 74. [Online]. Available: http://cycling74.com/products/max/ [Jun. 17, 2013].

[15] E. A. Bier and S. Freeman, "MMM: A user interface architecture for shared editors on a single screen," in *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '91, New York, NY, 1991, pp. 79–86. [Online]. Available: http://doi.acm.org/10.1145/120782.120791

[16] J. Stewart, B. B. Bederson, and A. Druin, "Single display groupware: A model for co-present collaboration," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '99, New York, NY, 1999, pp. 286–293. [Online]. Available: http://doi.acm.org/10.1145/302979.303064

[17] U. Pawar, J. Pal, and K. Toyama, "Multiple mice for computers in education in developing countries," in *International Conference on Information and Communication Technologies and Development*, Bangalore, India, 2006, pp. 64–71.

[18] U. S. Pawar, J. Pal, R. Gupta, and K. Toyama, "Multiple mice for retention tasks in disadvantaged schools," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07, New York, NY, 2007, pp. 1581–1590. [Online]. Available: http://doi.acm.org/10.1145/1240624.1240864

[19] P. Barr, R. Biddle, and J. Noble, "A taxonomy of user-interface metaphors," in *Proceedings of the SIGCHI-NZ Symposium on Computer-Human Interaction*, Hamilton, New Zealand, 2002, pp. 25–30.

[20] L. M. Reeves *et al.*, "Guidelines for multimodal user interface design," *Communications of the ACM*, vol. 47, no. 1, pp. 57–59, 2004.

[21] A. Hunt and R. Kirk, "Mapping strategies for musical performance," *Trends in Gestural Control of Music*, vol. 21, 2000.

[22] A. Hunt and M. M. Wanderley, "Mapping performer parameters to synthesis engines," *Organised Sound*, vol. 7, no. 2, pp. 97–108, 2002.

[23] H.-C. Steiner, "Building your own instrument with Pd," in *Proceedings of the 1st International Pd Conference*, Graz, Austria, 2005.

[24] G. Valentino and S. Zammit, "Design and implementation of an intelligent usb peripheral controller," in *3rd National ICT Conference, WICT*, Msida, Malta, 2010.