Building Mobile Instruments for Improvised Musical Performance

Damon Holzborn

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Musical Arts
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2013

Abstract

Building Mobile Instruments for Improvised Musical Performance

Damon Holzborn

This paper explores an approach to building electronic musical instruments for use in improvised music that I have found to be particularly effective for developing flexible, dynamic, and versatile instruments well adapted to the improvised context, and a resultant set of suites of solo improvised character pieces. The lessons learned from this research can be useful beyond the scope of this particular instrument design philosophy. In Part I, I present the foundations of my approach to instrument design, based on my past experience and the technological environment in which electronic music has developed. I discuss the values that guide me in the creation of instruments for use in improvised performance, and describe the development tools iRTcmix and Nikl, and Dixey, an instrument I have created with those tools and hardware devices using the Apple iOS operating system. Part II discusses the musical issues related to the creation of *Character Weekend*, a set of solo recordings produced with the tools described in Part I.

Table of Contents

List of Charts, Graphs and Illustrations

Acknowledgements

I'd like to thank my advisor George Lewis for the inspiration that he has provided me for over 20 years, and Brad Garton and Douglas Repetto at the Computer Music Center, who exposed me to so many new ideas and opportunities during my time at Columbia University. I'd also like to thank my friend and long-time collaborator Hans Fjellestad who is always available to debate ideas old and new and never hesitates to challenge me.

Dedication

To my parents, Owen and Nancy, whose support has meant the world to me. And to Betsy, for her patience and encouragement.

Foreword

I had my first experience with a synthesizer when I was a teenager in the 1980s. I had the misfortune of coming into the world of electronic musical instruments just after the digital era of synthesizer design had been established. My first synth, the Roland Alpha Juno 1, was typical in that the front panel of the instrument, rather than being covered in the knobs and sliders that encouraged exploration in its predecessors, had been reduced to little more than a piano-style keyboard, which was not even velocity sensitive. The instrument was programmable, but the tiny screen, with its pages and pages of menus, wasn't particularly friendly to experimentation.

A few years later, I was exposed to an instrument from an earlier era. The Roland System 100 was a semi-modular system from the 1970s. As with full modular systems, the instrument didn't do anything until you connected the various modules together in a "patch," thus finishing the job of instrument design that the engineers at Roland had started. The modular design allowed for plenty of flexibility in creating new sounds, but was high maintenance. If you wanted a different sound, a lot of plugging and unplugging of cables was required first.

Thus began the first of many experiments with complex electronic music systems. The next step, after I discovered and dedicated myself to free improvisation, was with guitar and electronics. These had a different set of capabilities from the modular synth, to be sure, but at least the piles of cable were familiar. I eventually traded in the real cables for virtual ones when I converted to a laptop-based system using the MaxMSP programming environment.

A curiosity to create and explore drove this quest, but so did a dissatisfaction with the environments I had created for myself. I finally realized that my problem was not one of degree but of direction. I was building large systems to create as many options as possible in order to have maximum expressive power. This, I eventually realized, was the wrong approach. The strategy I came to embrace was one designed to strip away rather than pile on.

Part I of this paper discusses my new approach to instrument building. I start with what brought me to this point, sharing details of past attempts at producing satisfying instruments. I describe antecedents in the industry, inspirational and otherwise. After describing the values and strategy that I employ in instrument building, I introduce the instrument development environment that I'm currently using and provide an example of an instrument that that I developed within it.

In Part II, I present my technique of developing timbre-based works that take advantage of my current approach to instrument building, and *Character Weekend*, a resultant work. *Character Weekend* is a set of short solo pieces for electronics. The idea for this piece came to me when I was in front of a class of undergraduates teaching Robert Schumann's *Carnaval* suite of character pieces. I realized that the solo character piece would be a perfect medium in which to explore my current instrumentarium. With no stitching together of takes, overdubbing, or other studio assemblage techniques, the instruments would be raw and exposed. Although I didn't take the "character" aspect of these pieces as literally as Schumann did, a series of short works, each of which explores one idea, was an exciting challenge to the expressiveness of my new instruments.

Part I: Instrument Design

Introduction

I have tested different strategies over the course of more than 20 years as an improvisor and instrument builder. My early explorations in developing systems for electronic musical improvisation started with the electric guitar. I assembled stomp boxes, rack effects, and various controller pedals into large systems that attempted to balance the musical goals of flexibility and variety with the more practical concern of playability. Since my preferred style of playing favors timbre and texture over pitch and rhythm, the level of success I felt I achieved was limited. Since both hands were busy with the grid of the guitar, much of my available dexterity was tied up in an interface that was designed for music that favored pitch. It was difficult for the system to provide continuous control of timbre. The manipulation of the dynamic properties of various effect processors was largely relegated to foot control.

Dissatisfaction with that strategy eventually led me to abandon the guitar entirely. I started by incorporating hardware digital samplers into my existing system, but by 2004 I had converted to a laptop-based strategy. Rather than having to adapt my playing to the instrument, I was now able to use the flexibility of blank-slate systems such as Cycling 74's Max programming environment to pursue my preferred playing style. I combined my own custom Max patches with commercial software such as digital effect plug-ins. While this strategy proved fruitful for a number of years, it accumulated a level of complexity with which I eventually became uncomfortable. Presented with the unlimited range of possibilities available to the laptop musician, I found myself creating a large, complex system. An array of objects for producing sound, looping, and processing were combined into one unified patch, called Enchilada, which I used and developed from 2004 until 2011. The goal was to have as many options as possible to allow me to adapt to any improvised context. In time, I came to think of Enchilada as a kind of cruise ship. It had a lot of power but it wasn't as nimble as I would have liked, with much less performative mobility than I envisioned.

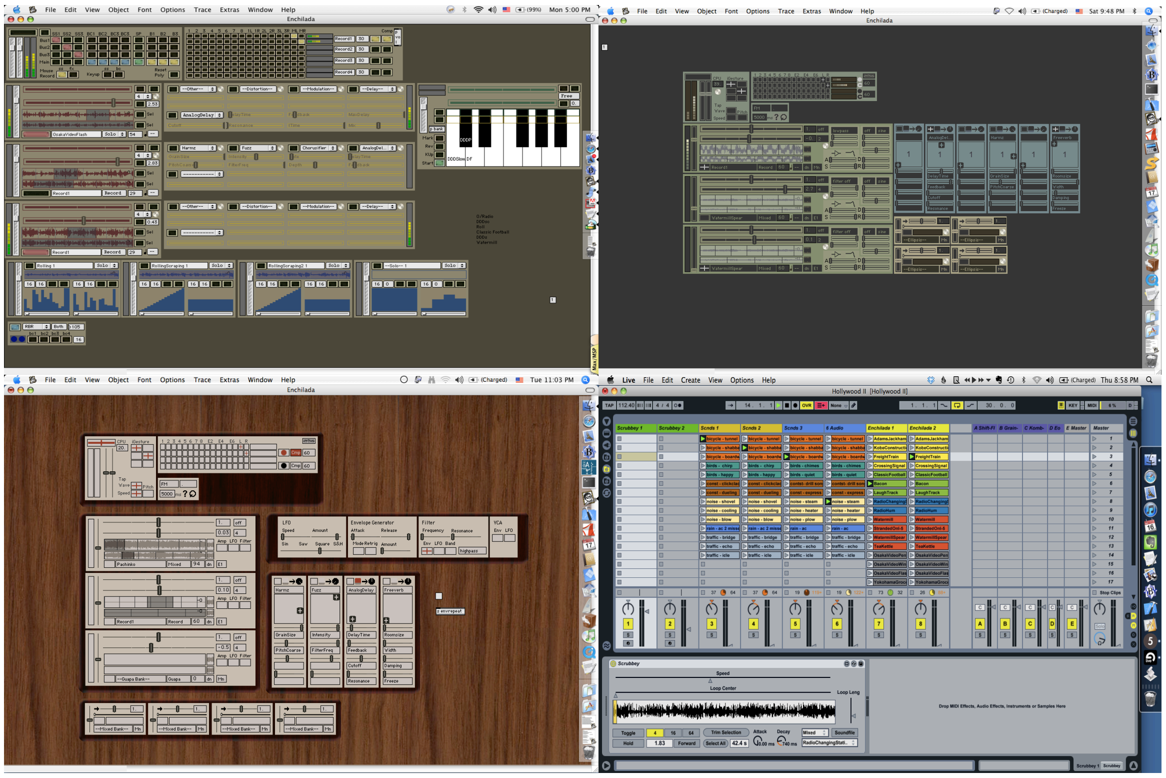

Figure 1 - Enchilada - four views of my primary performance system as it evolved over the years.

In 2010, I began to change my design and performance strategy, phasing out the use of Enchilada. Instead of unwieldy cruise ships with as many features as possible, I started to work toward building speedboats, expressive instruments with a small set of features that could be easily accessed without the cognitive and temporal barriers that arose when switching among a complex set of states, as with the Enchilada model of interaction with the instrument. What I lose in variety in this approach is compensated for by both a greater degree of nuance, and by having a multitude of these much simpler instruments available. Just as a woodwind player might switch from saxophone to clarinet to flute, I can switch from one instrument to another. However, at any one moment, my attention need only be focused on one simple set of abilities. The design philosophy behind this new strategy, and a case study of a resulting suite of character pieces, is the focus of this paper.

Origin Myth

I received a Korg Monotron as a gift for Christmas in 2010. My design philosophy had already started to move towards a simpler approach, and my encounter with the Monotron cemented it. I had read about this simple little synth and seen YouTube clips of others playing it, but this was my first opportunity to play one. From what I had seen, heard, and imagined, it looked to be merely something that I would pull out once in a while to kill a few minutes, or just to have a little fun. In an industry that has spent the past few decades packing as many features as possible into its gear, the Monotron may appear to be nothing more that a toy, hardly to be confused with a "real" instrument. However, I, along with many others, quickly discovered that the Monotron was considerably more useful than expected. As Peter Kirn points out, "If the first blush of monotron was 'toy,' those of you who picked them up and made music with them proved they were more" (Kirn 2013). The powerful simplicity of the Monotron family of instruments became important, not only in my own music-making, but also as a portable and tangible reminder of the lessons of simplicity that I was already starting to apply to my own instrument building.

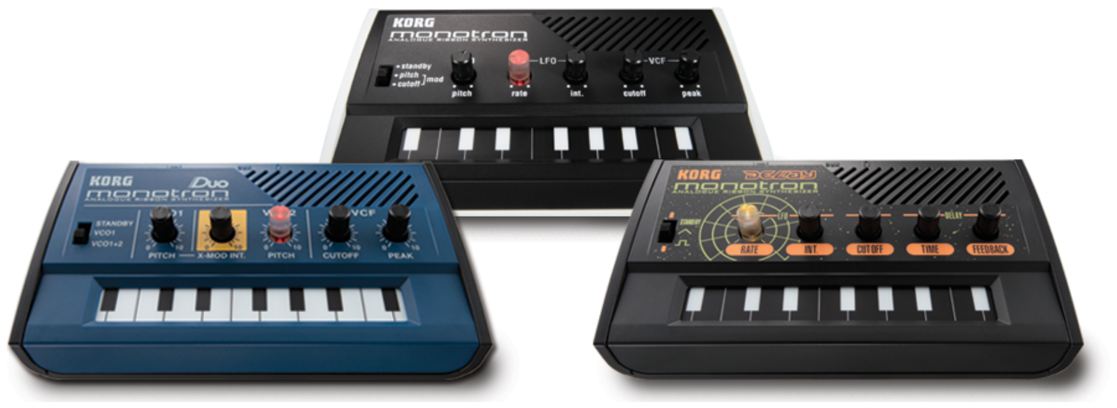

Figure 2 - Korg Monotron family.

The original Monotron is an "Analogue Ribbon Synthesizer" that fits in the palm of your hand. It is a monophonic instrument with a single oscillator, one low-pass filter with resonance control, one sawtooth-wave LFO (low frequency oscillator) to modulate either the pitch or the filter frequency, and a touch-sensitive ribbon that functions as its "keyboard." The sound is projected through a built-in speaker or the headphone jack, and it has an audio input for processing external sounds through its filter (Korg 2013).

This is a paltry feature set compared to most of the instrument's larger analog cousins, not to mention the hordes of workstation-style digital keyboards. Nonetheless, the Monotron embodies many values that I have come to appreciate in my work as an improvisor and instrument builder.

For one thing, it is a "complete" instrument. Nothing more than two AAA batteries are required for a musical performance. This is, perhaps surprisingly, something that distinguishes the Monotron from most of its larger and ostensibly more capable peers. A synthesizer, like an electric guitar, is only one part of a system that comprises an electronic musical instrument. Both need at least one other piece of equipment, an amplifier, in order to be heard. In addition, many synthesizers do not include an interface for musically interacting with the device. It is expected that the user will have some sort of controller, in most cases communicating to the synthesizer via MIDI. Of course, these days a "synthesizer" is as likely to be software in a laptop (a "software synth") as a physical object. So the chain could end up looking something like this:

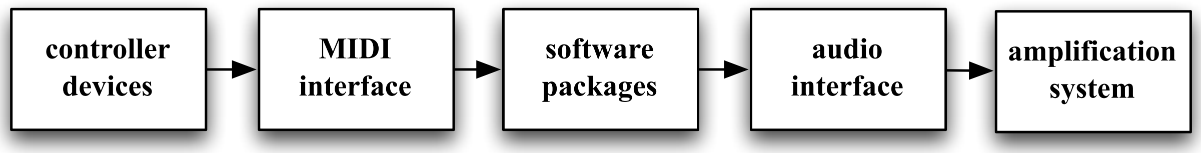

Figure 3 - Example laptop performance system.

In contrast, Korg has boiled down the Monotron's feature set to just about the minimum required for a musical performance. Control over rhythm, pitch, timbre, and dynamics are all afforded through the instrument's five small knobs, one switch, one ribbon controller, and one volume roller, all of which add up to a kind of access to the instrument that most "systems" lack. In the same way that someone could pick up a guitar when the mood strikes, a Monotron might sit on an end table, to be picked up in a moment of boredom or inspiration. This contributes to a sense of personal connection to the Monotron as an instrument that a hard drive full of software lacks. Guitarists are famous for their relationship to their guitars, even going so far as to name their favorites, such as B.B. King's string of "Lucilles." How many laptop musicians do you suppose have named their laptops? While I doubt we are going to find a musician dedicating his or her life to the Monotron in the same way that one might to the trumpet, for example, this property of accessibility is oddly compelling.

The extreme constraints of the Monotron family allow for a great deal of performative mobility. In a more complicated system, though one may have the entire world's collection of soft synths installed, ostensibly affording a wider variety of techniques and a far greater number of sounds, the complex interface required to realize this set of possibilities makes it more difficult to navigate in performance situations. Particularly in a live improvised context, where the unexpected can arise at moment's notice, many capabilities will go unused if, to migrate from one state to another, one has to open a window, click a tab, flip a switch, and move a few knobs first. By the time that's been done, the moment has passed. Limits can be liberating.

Models

> *Suddenly, one day, it seemed clear to me that full flowering of music is frustrated by our instruments ... In their range, their tone, what they can render, our instruments are chained fast and their hundred chains must also bind the composer.*
> -- Ferruccio Busoni (1907, quoted in Holmes 2002)

> *Our musical alphabet must be enriched. We need new instruments very badly ... I have always felt the need of new mediums of expression in my own work. I refuse to submit myself only to sounds that have already been heard. What I am looking for are new technical mediums which can lend themselves to every expression of thought and can keep up with thought.*
> -- Edgard Varèse (1915, quoted in Morgan 1991)

I believe the use of noise to make music will increase until we reach a music produced through the aid of electrical instruments which will make available for musical purposes any and all sounds that can be heard.
-- John Cage (1937)

Despite a century's worth of ideas like those above, "new" sounds have not been the main goal of the most popular electronic instruments. The Theremin, invented in 1919, not long after Busoni wrote the words above, had a sound that would most often be compared to a singer or violin. It gained its early fame through traditional music performances, such as those by virtuoso Clara Rockmore playing the Western music repertoire (Oliver 2012). The Hammond organ was developed to imitate a real organ, down to details such as drawbars labeled with pipe lengths. Though it didn't actually sound that much like a pipe organ, it often served in the same contexts, becoming a popular church instrument in the 1950s (Pinch and Bijsterveld 2003). The first experiments with the electric guitar were intended to amplify a quiet instrument, rather than create a new sound world. Both the organ and the electric guitar would later be used in new ways by creative musicians, but the original development was driven by existing needs, not in order to "enrich our musical alphabet."

Of the two major analog synthesizer pioneers of the 1960s, only Don Buchla remained firmly committed to the uniqueness of the synthesizer as a new instrument, for which new playing techniques should be developed and a separation maintained from what had come before. To this day, he refuses to develop a traditional piano-style keyboard for his synthesizers (Pinch and Bijsterveld 2003). Robert Moog had no such qualms; the keyboard he developed for his early modular synthesizers was used by Walter (later Wendy) Carlos to make the first big hit of the synthesizer era, "Switched On Bach" (Carlos 1968), using this new instrument to imitate 200-year-old music. Shortly thereafter, Moog released the first portable synthesizer, the keyboard-dominated Minimoog (Pinch and Trocco 2002), and the concept of "synthesizer" became almost inextricably associated with keyboard instruments.

As the technology moved from analog to digital in the 1980s, the keyboard's domination of the synthesizer became complete. Performance interface design for early digital synthesizers was almost entirely limited to the keyboard. Both the Fairlight CMI ("Computer Musical Instrument") and the Synclavier, early digital audio and performance workstations, were built around the keyboard, with the latter even including "clavier" in its name (Wishart 1996). As the price of digital technology dropped enough to bring it into the mainstream, most keyboard-based synthesizers would have little more than a pitch bend wheel and modulation wheel to provide performance-time continuous control over the sound of the instrument. The extreme flexibility of the early synthesizers in the control of every parameter was lost, and pitch became nearly the sole focus of performance control. As samplers (or playback-only "ROMplers", or read-only memory players) came to dominate the market, the use of synthesizers to imitate other instruments directly, in the form of digitized recordings of those instruments, further limited any pretense of the new in much of the industry. In this period, the image of a synthesizer player evoked a person surrounded by racks of keyboards. In fact, you are likely to find "keyboards" listed as the instrument of the synthesizer player (The Cars 1981, Depeche Mode 1981).

This brief summary of this period of synthesizer development, while capturing the foundation of the modern perception of what a synthesizer is, only tells part of the story. Modular systems designed by Serge used touch sensitive keyboards similar to the innovative designs of Buchla. Moog also experimented with touch-sensitive keyboards in the 1980s (Fine 1993), but this was never a mass-produced item. Boutique manufacturers continue to produce innovative approaches to touch control with products like the Haken Continuum (Haken Audio 2013) and the Madrona Labs Soundplane A (Madrona Labs 2013). As the underlying technology has come to be accessible to the DIY (do-it-yourself) crowd, individuals are now freer than ever to experiment with custom designs. But despite a resurgence in interest in analog synthesizers that has produced new hardware synths that bring back the knobs from an earlier era, the bulk of the music technology industry is still focused on emulation.

Values

> *A convention is a cultural constraint, one that has evolved over time. Conventions are not arbitrary: they evolve, they require a community of practice. They are slow to be adopted and, once adopted, slow to go away. So although the word implies voluntary choice, the reality is that they are real constraints on our behavior. Use them with respect. Violate them only with great risk.*
> -- Donald A. Norman (Norman 1999)

The values that go into the design of electronic musical instruments for use in live improvisation are strongly linked to the musical values of the instrument's creator(s) (Nissenbaum 2001).

Creating instruments for one's own use gives a builder-composer-performer the opportunity to embody those values at the most basic level of the practice of electronic music performance. Music has grown and changed along with the development of instruments and the invention of playing styles (Ryan 1991). Claiming responsibility over the development of the means of sound production provides power to take control of all sides of the equation. As both an artist and an engineer, the work is divided into two parts. There are often tasks on the engineering side that concern technical details that distract one from the concerns of the artistic side. Even the simplest interface design contains a complex set of compromises. A firm understanding of underlying principles will guide the designer to balance sometimes contradictory needs. By maintaining an awareness of the values important to the task, the designer develops a guide to aid both the production of instruments and the artistic process.

My instrument building strategy exhibits a fundamentally problem-based approach, in which I start with a musical problem and design the interface in order to solve that problem. This may be distinguished from a solution-based approach, in which the interface comes first and the parameters to be controlled come later. Prominent examples of the solution-based approach include gaming hardware such as the Nintendo Wiimote and the Microsoft Kinect, used as non-traditional musical interfaces for which builders have started with a catalog of modes of interface to the novel hardware, and subsequently sought musical parameters to attach to the gestures. In this approach, the resulting sound is framed as secondary to the interaction; thus, the importance of the link between action and result is weakened (Paine 2009). In my problem-based approach, the primary guiding issue in determining how to design the interface is rooted in the sound of the instrument. I develop a specific set of parameters available to sculpt the sound and must design the proper interface from which to control them.

The concept of expression is important to a musician in any culture, even while the term means something different to each individual. Tim Blackwell sums up what it means in improvised music:

> Expressive qualities are high level descriptions of the music, including attributes such as event density, average loudness and pitch. The participants interact by either trying to match the expressive quality of the musical environment, as they perceive it, or by attempting to change it in some way. (Blackwell 2003)

The details may vary from musician to musician and situation to situation, but some version of this principle forms the basis for generating a set of values from which to work. If the goals allow the performer to follow musical ideas and other musicians, then an instrument can be said to be expressive (Arfib, Couturier, and Kessous 2005).

In building instruments for improvised music, there are a number of concerns one can recognize that will apply to a wide range of personal styles. First, it is important that the interface support immediacy of action. No amount of brilliant sound sculpting can overcome an interface that does not afford quick and easy access to its performance techniques. This is related to what, in the design world, from a technical standpoint, would be called usability. Just as expressiveness needs to be considered as the basis for the artistic side of the process, the practical concerns of interface need to underlie each part of the technical conception of the instrument. A real human, with limited attention and dexterity, will have to interact with the system in real-time. Any feature that divides the attention or breaks the concentration of the performer must be considered carefully.

Moreover, the performer must be able to learn and master the instrument. The context in which the instrument will be used will determine the level of difficulty the instrument should exhibit, as will the amount of time that the performer may devote to mastering the instrument. How far the interface diverges from familiar models will be an important determinant of the level of effort required for mastery. As synthesizer performer Bob Ostertag points out, this

> is closely related to the issue of virtuosity, by which I mean what happens when someone acquires such facility with an instrument or paintbrush, or with anything physically manipulable, that an intelligence and creativity is actually written into the artist's muscles and bones and blood and skin and hair. It stops residing only in the brain and goes into the fingers and muscles and arms and legs. (Ostertag 2002)

What Ostertag describes is one of the more difficult values to adapt to instruments that are built in software: physicality. The lack of a fixed and physical form in software instruments presents several challenges. One challenge arises from the distinction between the response and the responsiveness of an instrument (Ryan 1991). Here, "response" refers to the aural characteristics, the types of sounds an instrument is able to create. This applies equally well to the physical and virtual. Responsiveness, on the other hand, is related to the physical feedback generated by the instrument, and is often lacking in an instrument that exists in software.

An acoustic instrument's physical properties push back against the player's technique in ways that suggest, if not completely define, the limits of control, providing a source of tension that can be used by the performer in creative ways. A guitarist not only has the sound emerging from the guitar to guide the fingers, but also physical sensations, such as the feel of the frets and the resistance of the strings. Furthermore, the sound produced by traditional instruments results from a complex relationship among the control parameters, not a simple one-to-one mapping of parameter to control as is often seen in electronic instruments (Rasamimanana, Kaiser, and Bevilacqua 2009). These physical properties allow for a very nuanced level of control for the experienced acoustic performer.

An electronic instrument does not exhibit the same type of resistance, so the builder must devise the appropriate level of depth in order to anticipate and realize experimental and exploratory capabilities (Wessel and Wright 2002). If it is too unpredictable, the performer loses control. An instrument that lacks depth, however, may not provide the challenge necessary to create a dynamic performance.

The affordance of virtuosity is one of the fundamental challenges of the electronic musician. Muscle memory is a powerful aid to any physical task, and can only be obtained through repetition. An improvisor must develop a deep level of comfort with the tools in order to successfully navigate a performance. In order for the performer to develop skilled playing technique the builder must take care to monitor the rate of change and the level of novelty (Gurevich et al. 2012). In order for an instrument to exhibit an acceptable amount of expressive depth, the builder must strike a balance between a level of sophistication that invites exploration and the predictability that allows for consistency and the effective expression of intention. Once fundamental concerns of practicality are met, the level of flexibility the instrument can provide will vary according to the musical values of the creator as well as the role the instrument is to fulfill.

Even if one cannot, or at least does not, attempt to recreate the physical properties of traditional instruments, there are other lessons to be learned from physical instruments, such as their relative stability as fixed in physical form. The Theremin, particularly interesting in this context as a "physical" instrument that one does not actually touch to play, demonstrates this feature well:

> ...the Theremin was a conceptually complete instrument that did not undergo a constant series of revisions, redesigns and "upgrades." One could devote years to learning to play it without worrying that all that hard work would be made useless every 6 months by an "upgrade" that changed everything. (Ostertag 2002)

If an instrument is changed beneath one's fingers every few months (or weeks, or days), the ability to develop proficiency will be to some extent frustrated. Performers rely on the human capacity for muscle memory gained through years of hard work and repetitive exercises on their instrument. A certain minimum level of stability is thus to be valued when developing an instrument. One great advantage electronic instruments have is an essentially unlimited capacity to grow and improve. It is easy to add a few lines of code or a new software package in response to new demands or to make up for perceived deficiencies based on experience gained through time spent with the instrument. This advantage must be balanced, however, with careful consideration of the resulting physical demands made of the performer.

There are also extra-musical considerations to assess. Performer movement affects more than just the sound generating capacity of the instrument; concern with the physical presence of the performer is necessary to provide the visual cues that audiences have come to expect from instrumental performance. Music making is a physical act and some level of movement is natural during the process. Physicality communicates intention to the audience, even those gestures that are purely theatrical, acknowledging that "musicians use gesture both as a means to engage the production of sound on an instrument and as an expression of an inner intentionality" (Paine 2009). Visual gestures exhibit intentionality that empathetically connect the performer, the sound, and the audience (Katz 2010). The understanding of the intent of the actions increases the audience's response to that action (Herndon 2010). Conversely, an interface that ignores the physical can alienate an audience, leading to the ubiquitous comparison of laptop performance to checking email. Addressing the performance aspect of an instrument serves to adapt it to the social demands that exist in the presentation of live music (Bahn et al. 2001).

In addition to the broadly generalizable values that I have discussed thus far, there is a further set of concerns that, while neither idiosyncratic nor intended as a list of rules, prescriptions, or proscriptions, are significant in how they relate to the way in which I personally make music. Taken together, they can be seen as a set of principles that guide my process of designing instruments for use in live improvised music.

There is always a battle between simplicity and complexity, and the temptation to make exceptions is great. The easiest way to avoid the temptation for a layered interface is not to have that feature available at all. Instead of finding ways to add another feature or control, one is encouraged to find creative solutions while working within a simpler, more manageable environment. Thus, in my work, the presence of multiple states and hidden interface elements--pop-up windows, tabs, or some other method of displaying additional controls that won't fit in a single interface--are to be avoided as much as possible.

Traditional computers, whether desktop or laptop, have never been easy to use as musical instruments. Even though the multi-touch capabilities of modern devices carry much more potential for effective control than earlier, pointer-based interfaces, QWERTY keyboards, mice, trackpads, and GUIs, any changing of state of an interface takes time--time that is not always available in a real performance situation. The elimination of state changes avoids potential pausing points in performance; there will be no interruption while changing from one state to the next, or checking to verify the current state. The resulting instrument will therefore be quick to adapt to changing circumstances and find itself at home in more contexts (Bahn et al. 2001).

My interfaces also must support a continuous, timbre-based approach to music making, rather than the lattice and note-based approach that most of the instruments coming from major electronic music manufacturers are designed to produce. Rather than an interface based around a large grid of keys, controls capable of a continuous range are dominant. This doesn't prescribe any specific means of controls; knobs, sliders, expression pedals, touch screen controls and any number of physical or virtual interface elements are capable of providing continuous control. It does, however, virtually eliminate the many popular keyboard-based devices and grid controllers (such as those commonly used to interface with Ableton Live).

Simplicity

> *Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.*
> Antoine de Saint-Exupery, *Wind, Sand, and Stars* (1939)

There has been much research supporting the benefits of using simplicity as a strategy to improve design. Keeping user interfaces clean and carefully grouping elements keeps the user's attention focused and minimizes distractions (Novella 2012). Recent work has shown that simplicity can facilitate expressive actions and found an alliance between constraint and the development of style (Gurevich, Marquez-Borbon, and Stapleton 2012). Simplifying the user interface and freeing the artist from being overwhelmed by too many choices can be useful in generating novelty and guiding exploration toward the discovery of new possibilities and transformations, ultimately becoming a source for creativity and inspiration (Magnusson 2010).

It is important to recognize that simplicity is a technique--a means to reach design goals, not the goal in itself. It is easy to see how some of the values previously discussed--avoiding hidden interface elements, immediacy, and learnability--map well to a strategy of simplicity. It may be less obvious, however, how other principles are served by this strategy. Flexibility may seem at odds with simplicity; it implies addition rather than the "taking away" that Antoine de Saint-Exupery encourages in this section's opening quotation. The capabilities of a music performance system are the result of the combination of instrument and performer, not merely a list of abilities of the instrument alone. The usefulness of an instrument is ultimately determined through the results the performer is able to generate with the instrument, and the possibilities of dexterity and attention available to a performer will be poorly served by an overly complex interface. By careful management of the complexity of the user interface, the details that are presented to the performer can receive greater attention and focus may be better maintained.

Development Environment: iRTcmix and Nikl

There has been much attention paid to the application of simplicity in design in recent years, often connected to the success of the iOS operating system for the iPhone, iPad, and iPod Touch, and the app ecosystem that has developed around it. The iPhone, with a nearly unmarred glass surface, set a standard for stripping down an interface to the bare essentials. It's a phone without a "phone" button. App developers have followed Apple's example and many have produced software with spartan interfaces, radical constraints, and innovative modes of user interaction. The makers of iOS apps, such as the drawing app Paper, the list app Clear, and the music sequencing app Figure, have all attracted attention through novel applications of these techniques.

The iPad has proven to be a satisfying platform on which to develop an instrument. Interaction is natural, physical and easily incorporated into physical practice (Wang, Oh, and Lieber 2011). My primary instrument building environment for this platform is based on the RTcmix digital signal processing and sound synthesis language (Garton 2013). Embeddable and command line versions of this language are available for Mac OS and Windows, along with versions that work within software platforms such as MaxMSP, Pd and SuperCollider, and interfaces for the Perl and Python programming languages. The version to which I have turned my attention is iRTcmix, a system for using RTcmix to develop iOS apps. In the following section, I illustrate my instrument building practice, informed by the philosophy described previously, through my instrument building tool Nikl, and through Dixey, an FM synthesis instrument built using Nikl and used in the composition *Character Weekend*.

iRTcmix has been developed over the course of the last few years by Brad Garton and myself. The basis of the system is the RTcmix library, compiled for use in an iOS app. This includes the many instruments that are part of the library, as well as the Minc scripting language parser, used to process the "scores" that give the instructions to RTcmix for sound production. iRTcmix also includes two convenience classes, written in Objective-C, that provide the glue between an author's app and the RTcmix library. The first, RTcmixPlayer, contains the basic commands necessary to communicate with RTcmix. The second, RTcmixScore, is a convenience class to streamline the process of sending Minc scores to the RTcmix parser. The library makes it possible to include sound generating functionality in an iOS app with just a few lines of code. No deep understanding of digital signal processing code is required--only a basic grasp of the iOS development APIs and the Xcode environment. The system is designed with the inexperienced programmer in mind, and many example projects are available to demonstrate all the features available. iRTcmix may be used to build apps for use solely by the author or for distribution in the iOS App Store.

I have developed Nikl on top of the iRTcmix foundation. Nikl is an RTcmix instrument development app that uses XML template files that contain both the description of the user interface and the Minc scores for producing sound. A debug mode is available to display error messages, with troubleshooting of templates to be done within the app. In the case of hard crashes, error messages may be viewed in third-party apps available in the App Store, such as System Status. Editing of the XML template files may be done in the user's favorite iOS text editor and sent to Nikl with the "Open In..." feature. Editing may also be done on a desktop computer and synced to the iOS device, using a service such as Dropbox, or sent via email. These methods also make it easy to share templates with other Nikl users.

Nikl itself is a ground-up redesign of a previous effort called Smoothey. Like Nikl, Smoothey enabled a user to develop instruments that included both the Minc scores and the user interface elements to connect to them. After using Smoothey for development and performance over the course of many months, I ultimately decided to scrap it and start over from scratch in order to simplify the instrument building process. The differences between the two apps illustrate a refinement of my instrument development process.

Both the development process and the feature set were simplified in this redesign of the system. In Nikl, the user interface is designed through the editing of a text file. Smoothey contained a GUI editor built into the app that allowed the user to manipulate the user interface items directly on screen by touch, dragging and stretching elements through the touch screen interface. This may seem to be easier than editing a text file, and for some users it may be. However, because years of manipulating text files for web development and programming has made me comfortable in the text editor environment, GUI manipulation in Smoothey became an undesirable time sink. Precise placement of each element was time consuming, and any change to the interface demanded further time wasted in rearranging all the elements. The time spent pushing pixels was a much greater portion of the development time than I was comfortable with. This approach emerged in part from my experience with using RTcmix from within MaxMSP. The short scores needed to create an instrument in RTcmix were much easier to read and edit when returning to them days, weeks or months later than the mass of graphical objects and wires in MaxMSP.

I realized if I simplified the interface design capabilities I could create an XML format that could be easily human-readable, allowing all editing to be performed using a text editor of the author's choice. For example, here is a template that produces a simple beep:

```
<?xml version="1.0" encoding="utf-8"?>
<nikl>

 <!-- descriptive data that appears in the file listing -->
<meta>
      <name>Hello Beep</name>
      <author>Damon Holzborn</author>
      <description>Beep!</description>
</meta>

<!-- one or more scores that the instrument will have access to -->
<scores>
      <setupscore> // this score runs when the template is loaded
            print_off()
      </setupscore>

      <onscore> // this score is defined by the builder and invoked through the
 interface
            WAVETABLE(0, 1, 28000, 440) // plays a one second long sine wave
 beep
      </onscore>
</scores>

 <!-- instructions for the user interface of the instrument -->
<interface>
      <zone1-4>
            <row>
                  <type>push</type>
                  <name>beep</name>
                  <label>Beep</label>
                  <onscore>onscore</onscore>
                  <color>blue</color>
            </row>
      </zone1-4>
</interface>

</nikl>
```

In this short example above, there are three sections to the template format. The first is the metadata. The name of the template, the author, and a short description may be included, to be presented in the Nikl template listing. The second section contains one or more Minc scores. There is a pre-defined "setupscore" that will be run when the template loads. The author may include anything that will need to be configured before play starts (e.g. effects routing), anything that needs to be run only once (e.g. tables for LFOs or envelopes), and various setup variables. In addition to the setupscore, any number of additional scores may be included to be invoked through the user interface. The final section contains the user interface elements, including all information necessary to communicate with RTcmix. Scores may be selected to run when the performer touches or releases an interface element and/or RTcmix variables may be changed, including "pfield" variables that exercise real-time control over the parameters of an executing score. The ranges of the values can be set for continuous elements such as sliders, along with a curve along which the slider will traverse the range. The display parameters of a color and a name for the element is also set in the template.

A simplification of the layout capabilities is required to support this minimal XML format. In Smoothey, the layout of an interface element was entirely arbitrary: any element could be placed anywhere on the screen with any dimensions. In Nikl, the screen is divided into quadrants. Up to four zones of the screen may be addressed independently. Within each area, however, the space is evenly divided into either rows or columns. If there are four rows in a zone, each element expands to fit the width of the zone and has a height of 1/4 of the zone height.

This regimentation of the layout does limit its flexibility, but in return, produces a valuable trade-off that borrows the concept from interactive software development environments:

> These are languages which allow the making of incremental changes in the software, each change taking only as much time as it takes the programmer to type it. Such methods allow direct manipulation of the composer's model in a loop with the composer's ears in the middle. The time it takes to cycle this loop is a critical part of the discovery process. If a particular path seems difficult, rapid feedback on one's hypothesis can make

the difference between attempting the path or not. (Ryan 1991)

Moving items around the interface is simply a matter of re-ordering a few lines of text in the XML file, thus streamlining the development process. An element of improvisation may thus be brought to the act of instrument building. Since the effort to create new interface elements in Nikl is trivial, with less time spent developing and more time spent testing, experimentation is encouraged. Planned features do not always work out and one often discovers new ideas during the testing process. Testing early in the development process produces more iterations of the building-testing feedback loop to generate a better performing instrument (Arfib et al. 2005).

The Instrument: Dixey

I will use Dixey, built in Nikl and used in *Character Weekend*, to provide an example of a realization of my instrument-building philosophy. Dixey is a two-operator FM synthesizer inspired by examples of successful minimal instruments like the Monotron family. It is a simple polyphonic instrument with an LFO for pitch or filter frequency modulation, optional distortion, and amplitude envelope control. The features of the synthesizer are controlled with interface elements presented on the iPad's touch screen, as well as by physically tilting the device, making use of the built-in accelerometer that measures its orientation. Nikl is not the first environment in which Dixey has appeared; the first version was built using Smoothey. A comparison of the two versions will illustrate how Nikl exemplifies my simplification strategy for instrument building.

By streamlining the interface development process, unnecessary elements are discouraged and a well thought-out interface results. Comparing the layouts of Dixey (figure 4) produced in each app, it is evident that the Smoothey version has more on-screen controls, which necessitates that some of them become smaller. There are, on average, larger touch targets in Nikl and modifications of some features to simplify the interface. For example, from using Dixey in performance, I realized that I rarely modified the amplitude envelope attack time. I decided that a fixed attack time was satisfactory, so I removed the attack time slider. The turbo mode (distortion) was changed from a continuous slider to a toggle button because I didn't find the level of nuance available with the slider to be very useful. In both cases, experience taught me that the elements were taking up space and attention unnecessarily, and that the interface could therefore be improved through the removal of controls and the simplification of the layout. Either of these changes could have been made in Smoothey, of course, but they never were. The regimentation of the template format in Dixey encouraged greater scrutiny of the necessity of each element, and made experimentation with the layout easier. The result was a more playable interface and a more satisfying instrument.

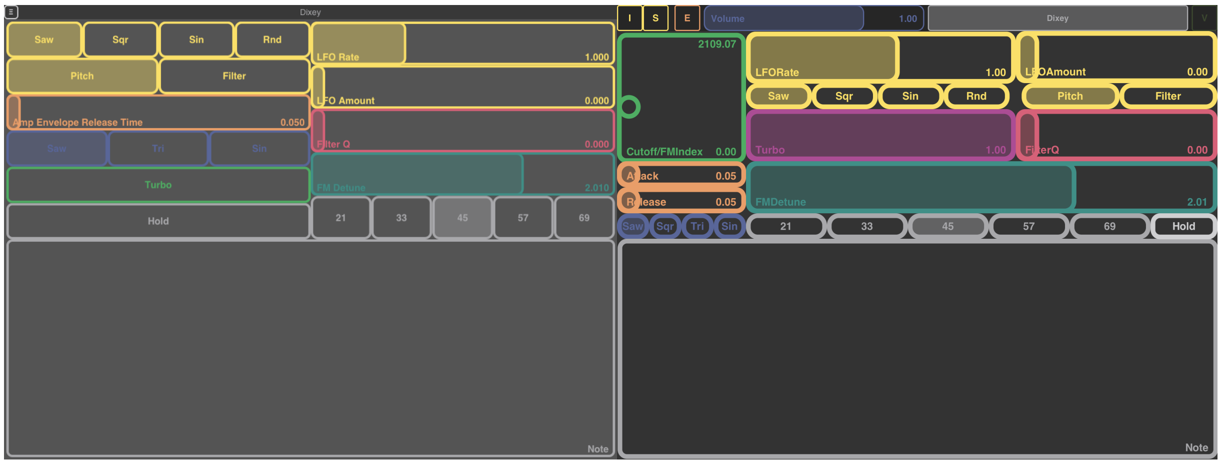

Figure 4 - Dixey built in Nikl (left); Dixey built in Smoothey.

Another difference between the interfaces produced by Smoothey and Nikl is how the iPad's built-in accelerometer data is displayed. The tilt of the device on the X and Y axes is reported at all times and this data is routed to RTcmix variables to control features of the instrument, in a similar manner to the on-screen touch controls. In Smoothey, this data is displayed in a two-dimensional slider, seen in green in the upper left corner of the screen. This serves as a display but also has the ability to convert to manual control. A double tap of the element will toggle between accelerometer and manual performance modes. This is an example of a feature that seemed great when I invented it but ended up being unnecessary in practice. I rarely, if ever, toggled the control to manual during performance in any of my templates. When developing Nikl, this feature was removed, creating less wasted space when using the accelerometer. In Nikl, the accelerometer data is not displayed graphically.

Smoothey also had the capability of creating a tabbed interface. The performer could toggle among an arbitrary number of screens full of interface items. This is another feature I removed when creating Dixey. As noted before, attention to managing states of the instrument is distracting and inefficient in performance, and removing these hidden interface elements allows the performer to focus complete attention to the musical performance. The goal of statelessness or near-statelessness challenges the builder "to be able to produce complex results without accumulating information from multiple gestures" (Wessel and Wright 2002).

It was in this context that I created the polished version of Dixey that was used in *Character Weekend*. This version is based on a two-operator FM synthesis instrument available in RTcmix. The multi-touch control that covers the bottom half of the screen acts as the note-triggering mechanism. The pitch of the carrier wave is controlled on the X-axis, spanning three continuous octaves, while the Y-axis controls the frequency of a lowpass filter independently for each note. Above the note control, there is a hold button to sustain notes and a radio button to select the base octave. In addition to the individual note filters, there is a global lowpass filter through which all notes pass. The Y-axis of the accelerometer controls the global lowpass filter frequency while the X-axis controls FM depth. The global filter resonance and oscillator modulator wave frequency are both controlled by a slider. There are sliders to control the LFO depth and speed, a radio button to toggle the assignment of the LFO to modulate either pitch or global filter frequency, and a radio button to select the LFO waveform. The final slider controls the amplitude envelope release time.

This economical interface provides a great deal of nuance to a skilled performer. It is designed to make it easy to operate several parameters at once. The gestures required to control the accelerometer add a degree of physicality to the performance, and though the accelerometer only adds two axes of control, it contributes greatly to the expressiveness of the instrument. The instrument is handheld rather than desk-bound, and by careful selection of assigned parameters, satisfying timbral control is possible by tilting the device, leaving both hands free (or really one and a thumb since the device must be held) to manipulate other parameters. Thus, instead of dedicating a whole hand to one parameter, as one would have to do with traditional hardware that uses physical knobs, a technique can easily be developed to control four, six, or even more parameters at once, resulting in a greater degree of nuance.

Part II: Character Weekend

Materials

> *I consider the creation of a specific electronic music instrument as being part of the compositional process ... The way a sound is created and controlled has such an influence on its musical character that one can say that the method of translating the performer's gesture into sound is part of the compositional method. Composing a piece implies building special instruments to perform it as well. The inventor role is thus an integral part of composing ... I cannot see a personal involvement in the technical functionality of the instruments and performance as separate from the work of composing, so simply consider me a composer.*
> -- Michel Waisvisz (Krefeld and Waisvisz 1990)

> *From the final quarter of the twentieth century, it now seems clear that the central watershed in changing our view of what constitutes music has more to do with the invention of sound recording and then sound processing and synthesis than with any specific development within the language of music itself.*
> -- Trevor Wishart (1996)

The design values of Part I describe a method for guiding the effort of instrument building that can be applied to a wide range of aesthetic values. Besides these values, a view of desired aesthetic results are necessary to guide the building process. *Character Weekend* demonstrates my personal compositional goals on a number of levels. The work is representative of my aesthetic values and uses performance strategies I have developed for creating material. In addition, it is an attempt to deal with the issues of recording improvisation and distributing music through the Internet.

The resources available to the digital musician are vast. Sound can be created through any waveform that mathematics can describe or that can be collected with a microphone. These sounds can be further transformed through another unmanageably large set of potential processes. Developing a set of principles that can be applied to a variety of possible methods of creating electronic sounds is better adapted to this problem of plenitude than is an attempt to catalogue and categorize. Timbre does not exist on a single dimension, as does pitch, nor are there any perceivable nodes in each of its independent dimensions (Wishart 1996). By describing approaches to sculpting timbre that can be deployed independently of source or technique, we may avoid a purely mechanical approach to this problem that will inevitably become obsolete either with the advent of new technology or a change of focus to previously unexplored processes or interfaces.

The basis of my work in the exploration of timbre requires a means for organizing material on a continuum. Timbre is the result of the interaction of many interrelated parameters. Features such as envelope, spectral energy, grain, noise characteristics, inharmonicity and various morphological characteristics all contribute to the timbral qualities of a sound (Wishart 1996). There is no universal way to describe timbre, nor are there particularly effective methods to notate it (Krefeld and Waisvisz 1990). Attempts to classify sound into objects necessarily results in arbitrary lines of demarcation drawn between categories.

Derek Bailey defined improvising as the search for material which is endlessly transformable (Fischer 2012). In electronic music, one is not limited by the physics of the instrument. Devising techniques to transform the sound in the technological context thus becomes the primary responsibility of the improvisor. Missing the physical responsiveness of a physical instrument, the focus is on the aural response of the instrument and how one manipulates and explores these characteristics in performance. The technical sound-generating capabilities of a given instrument become of secondary concern to the transformation processes that the performer develops.

Through this focus on transformation, the performer builds strategies for navigating timbre space that are not beholden to a particular technology or instrument. In *Character Weekend,* the various instruments used are unified by a common strategy of attention to the moment and active manipulation of fine details of the sound's development. Small variations can be made in the sound to create movement and variety. Multi-leveled timbres are developed to provide the complexity necessary to reward multiple listenings.

The improvisor must pay close attention to the context in which the sound appears. Michel Waisvisz says:

> Through experience I know that the meaning of even the smallest independently perceived fraction of sound entirely depends on its context. Every change creates a new way of interpreting the previous state of affairs. A particular sound is imbued with value by the sounds that precede it and, once more, by those that follow it. (Waisvisz 1999)

This context will guide further transformations of the sound. Thus, the process of developing material feeds back on itself to provide guidance for the performance. New directions to explore will also be suggested, even in the case of the solo performance, since each moment is unique.

The needs of the moment will supply the improvisor with inspiration guiding the exploration of new directions.

The improvisor is freed from the reliance on technology of the moment with this strategy. A personal style will develop that is not dependent on one individual instrument or system. If one compares *Character Weekend* with my 2004 solo release *Adams & Bancroft* (Holzborn 2004), one finds similarities in style that transcend differences in technology. Improvised pieces such at *Summit* and *O/Radio* display a strong relation to my more recent work, although they were recorded while in my Enchilada "cruise ship" phase rather than my current "speed boat" strategy. Despite the resources I had at the time at the time for layering, recycling, and building a bigger sound, there is still a clear focus on a sound as a whole rather than streams of independent processes. With whatever set of tools I use, I still possess the "ability to create music through gestures that have a direct relationship with what [I] want to express" (Waisvisz 1999).

The timbre-focused music that I aim to create requires the development of new criteria for exploring sound. We can develop a method to determine "what might be the effect of ordering sounds in one way rather than another, and what might be fruitful avenues for exploration" (Wishart 1996). Since sound is all around us, one way to start is to note the effects of environmental sound. It is useful to note that in nature, sounds are almost always dynamic, rarely perfectly constant or cyclical. Constant sounds have only relatively recently become commonplace. The Industrial Revolution generated movements dedicated to the irradiation of the noise that machines brought with them (Bijsterveld 2008). Constant buzzes and hums are annoying in ways that less predictable sounds are not.

Unpredictability implies agency. Humans are well equipped to recognize agency; the very survival of our ancestors depended on it, and we prefer it when things appear to exhibit these features. Phenomena that we now know have no actual agency, such as the wind in the trees or thunder in the distance, are capable of stirring greater emotion than the strictly inanimate. Even simple geometric shapes, given the proper animated display, will evoke attribution of agency (Barrett and Johnson 2003).

Acoustic instruments display related properties. Most acoustic instruments afford continuous control, and much of musical expression is dependent on this property (Wessel and Wright 2002). Moreover, since acoustic instruments follow physical principles, they will not produce strictly continuous tones. Even the most skilled performer of a wind or bowed instrument produces a tone with fine variations, and this property is pleasing. No listener is fooled for long by even the best sampled instruments; no matter how skillfully it is created, we sense the synthetic nature. Humans are too sensitive to the minute differences that are produced by a human performer with an acoustic instrument (Smith 1991).

This observation forms a foundation to direct the effort of exploring timbre space. Our innate preference for unpredictability provides a strategy that can be applied to any method of sound generation. Change itself becomes a guiding principle, and techniques for producing sounds must accommodate this need. This highlights the unsuitability of most commercial electronic musical instruments. Their predilection for imitating the instruments of the past results in a focus on pitch. They often lack well-developed capabilities to manipulate timbre in expressive ways that most acoustic instruments possess.

Even if we are not interested in breaking timbre into objects in order to develop an organizational strategy analogous to a theory of harmony, we may still borrow ideas from more traditional areas of music theory. The balance of tension and release that is developed in tonal harmony is a useful concept to borrow when developing a language in which to manipulate sound. Even when not using fixed objects, we can develop sets of features that will serve one function or another. Once these broad ideas are recognized, they can be used to guide the development of a composition. These fundamental observations can be used as a starting point for any number of practical ways of generating sound. Since perceptual judgements tend to be relative (Wessel 1979), we can apply these principles as appropriate within the context of the possibilities created by the instruments being used.

A basic feature of sound that may produce emotional tension is a complex, noise-like, spectrum. As the sound approaches white noise, however, it will start to have the opposite effect. White noise generators, after all, are sometimes used as a sleep aid. That concept of noisiness functions as a tension inducer with other parameters as well. Rapid modulation of dynamics, pitch or timbre can all be used to increase emotional tension. Any process that is irregular and reduces the level of predictability can also be useful. Related to that, sharp note attacks can also increase the overall tension even in the absence of predictability. Even in more stable environments, we can identify features that can contribute to building tension by looking for features that might create discomfort. Higher pitches, which are associated in the body with a greater vocal cord tension, are symbolically linked to greater emotional tension, as are louder sounds (Bijsterveld 2008).

To find ways to generate release, we look to the opposite of these features. A simple spectrum will sound more settled than a noisy one. This could be a less complex sound, with fewer elements to the spectrum, or something even more predictable, such as the harmonic series. A sound that is more stable, in as many parameters as possible, will serve to release tension. Quiet sounds, low pitches, and slow attacks will also contribute to creating a sense of calm.

These techniques for creating tension and release can also be helpful in finding ways to root the musical work. In the absence of a natural base, such as the concept of the tonic in tonal music, one still may wish to provide some feature that can be used as a perceptual arrival point. The patterns of tension and release can help to generate that sense of arrival, even in the absence of well-established patterns like the dominant-tonic relationship. Repetition may be employed to reinforce this.

A method to generate structural movement may also be borrowed from traditional harmonic concepts. William Hsu suggests:

> With the decreased importance of pitch, timbre and the rate of change of timbral parameters become more important structural elements. Gestures whose features evolved slowly are perceived very differently from gestures whose characteristics undergo abrupt

and rapid modifications. (Hsu 2005)

A modulation of a sort can be created though the grouping of timbral and dynamic characteristics of the sound and using the rate and amount of change to create local stable regions. Distance can be implied through the number of morphological differences between groups.

In *Character Weekend*, the sound generation techniques are varied. Three different Monotron analog synthesizers are all featured in the first suite, *Character Weekend 01*. The second suite, *Character Weekend 02*, uses Dixey, the instrument I described in Part I, for all of the pieces. For the third suite, I used the commercial synthesis app iPolysix, that models its namesake synthesizer from Korg, controlled by an interface I designed in the MIDI control app Lemur.

The fact that all three instrument families used in the work share a similar design pattern aided in the creation of a unified voice throughout. The simple interface used in all the instruments allows detailed control of the sound generation. This encourages a strong focus on just a few parameters and their interaction, rather than high-level oversight of automated processes. The instruments are further united though common means of timbral control. All use resonant low-pass filters and LFOs to modulate pitch, amplitude or filter frequency. These characteristics provide a consistency that grounds the sound despite varying methods of sound generation. Regardless of the specifics of the tone generation method used, I am able to manipulate the sound through a common set of filtering, processing and modulation techniques. Another feature that all of these instruments share is a simple oscillator at the start of the processing chain -- in some cases monophonic and in others polyphonic. Since my primary attention is on the sculpting of the timbre, techniques to add to the complexity of the waveform are vital.

Solo

Once one takes formal structure as a synonym for the beauty of patterns, cold reasoning, a law-abiding mind, and dogmatic thinking, and one interprets physicality as wild emotion, instinct, sensuality, eroticism, and feelable matter, then a historic intellectual conflict is perpetuated! The balance between structure and physicality is the most intriguing one I can imagine because one attempts to weigh out two highly contradictory but crucial entities. The composer who can handle these extremities is bound to create a lively piece of music.
-- Michel Waisvisz (Krefeld and Waisvisz 1990)

In improvised music in particular, there is a tension between the formal aspects of the "composition" that is produced in real time and the physical demands of the process used to realize it. It can be viewed as a battle, as the performer struggles with the physical demands of the instrument in order to generate a physical result of an intellectual intention. Part of the job of the improvisor is to come up with interesting battles, working within the limits of the instruments and the ability of the performer. The constraints within which the performer chooses to work have a great influence on the outcome of these battles. Coming up with intelligent constraints that strike a good balance between what is physically possible and what is theoretically desired comprises much of the work of preparation by the improvising musician. The improvisor must also be prepared to adapt to new circumstances not explicitly planned for as they arise. These responsibilities are multiplied when one builds the instruments as well as plays them. If the work in question is to be solo, this preparation is all the more important since the performer must come up with all material without the aid of another player with whom to trade ideas.

This battle is less readily apparent in a recorded work than it is in a live presentation. The visual witness to the physical act makes the struggle more apparent. In the *Character Weekend* series of suites, I aimed to expose this struggle even in the absence of an audience. It is common in electronic music, as it was in much of my previous work, for the tools used in the creation of a piece, even one that is performed "live," to contain much more than just the sounds that can be attributed to the direct agency of the performer. Techniques such as interactive processes, looping, and layering prerecorded elements can all be used to make the sound world created seem bigger or more complex than would seem possible with a solo performer.

In contrast, in *Character Weekend*, all of the sound that emerges from the instruments is the direct result of an act of the performer. No looping, overdubbing, or other automated processes are acting on the sound of the performance. In the creation of any given piece in the suite, I have only one simple instrument to use. There are no settings to reconfigure the instrument and no new presets to jump to. The situation is similar to that of a woodwind player who doubles on several instruments. There may be a choice, but only one will be used at a time. If while playing the saxophone, the player wishes the timbre of the flute, the saxophone must be set down and the new instrument picked up.

By keeping my materials stripped down I sought to expose some of the struggle and tension that is present in a live improvisation. The only editing I performed was to select the start and stop points in a given recording; there was no non-linear editing involved, eliminating the possibility of editing out a momentary flub in an otherwise interesting performance. Hence, while I lacked a live audience to apply external pressure, I managed to retain some of the urgency which is helpful in maintaining focus in performance. By keeping the pieces short, and attempting to limit the materials of each piece to just one idea to explore, I added further constraints to focus my efforts. By keeping the recordings raw, I believe that both the dynamism of the gestures and the struggle of the performance remain evident.

Even so, the focus that is imposed on the improvisor when presented with the pressure of a live audience is lost in this scenario. There is, however, something gained in exchange. One of the things I most enjoy during the instrument building process is the time immediately following the creation or modification of an instrument during which I get to explore new possibilities without inhibition. There is no audience, no pressure, and no requirement to play anything that makes sense for any predefined length of time. I just get to play, in every sense of the word. If I were to record these sessions and listen back to them, I might not find anything coherent enough to warrant presenting them to other listeners, but the goal of these sessions is not to create a coherent piece of music, but rather to explore the capabilities of the new instrument.

In these moments I'm free to start and stop ideas without the pressure to create a finished work and to try out whatever I think of next. Most importantly, I can push the limits of the instrument, and my playing ability, in a context free from consequence. This allows me to discover new sounds and new playing techniques. Experimentation with new techniques can only be approached carefully in a live context, due to the unpredictable results that may ensue.

In addition to the new discoveries one makes in this free play, one also is able to work on the very edge of control without fear of failure. Some of this freedom is retained when recording in the studio. Experiments that end in failure can be left on the cutting room floor. When they work, however, they have the potential to yield a result that is unlikely to come from a live performance. The potential to create something unique thus helps justify these frozen improvisations.

Recording Improvisation

> *Perhaps the debate over recording improvised music keeps rearing its head because, unlike other recorded music, there is no apparent economic justification for it.*
> -- Derek Bailey (1992)

The concept of recording is often a troubling one for improvisors. An improvised performance is never the same twice, while a recording is exactly the same every time. Repeated listenings may still be rewarding, and lead to the discovery of previously unnoticed details, but a recording remains fixed in place to a degree far beyond even the most fastidious performances of notated music. A recording has come to represent an "ideal" performance for many genres of music. Modern recordings are typically digitally created from multiple takes and do not document any single performance, but rather a perfected assemblage of many. In the case of notated music, that ideal is generated from the score and what the performer or producer finds to be the best realization of that score.

In the case of improvised music there is no such document from which to generate an ideal, and the ideal of perfection in improvisation is much harder to determine. Improvisation is a journey that a performer or performers take with an audience. Perfecting every moment is secondary to embarking on an interesting journey. Unfortunate moments pass quickly in the live context and can be forgotten as the performance continues. A recording is like a journey preserved in photographs: anything undesirable will be the same each time. That a listener has difficulty ignoring the repeated mistake raises the issue of editing. What type of editing should be allowed? A small snip just to remove an errant sound? Complete reassembly, transforming the performance into a new composition? Somewhere in between? If editing is allowed, what relationship does the recorded product have to the original?

Over the years I've made many releases of improvised performances, but I've had difficulty philosophically with the process. The questions above are difficult to settle in any satisfactory way. For a long time, I resigned myself to releasing CDs as "documents." This consigned the recorded product to a lesser status; a necessary evil, used to disseminate my work to a wider audience than I could possibly reach in live performance, or as a marketing tool used for self-promotion.

Digital distribution via the Internet has long since surpassed the sales of physical CDs. Releases are sold through the large digital retailers, most notably iTunes, though most releases are still the virtual versions of a physical release. The expense and effort of releasing a 60+ minute CD meant that the release of a recording would be a relatively rare event for me. The opportunity for digital distribution, however, has allowed me to approach the question in another way. Some of the immediacy of live performance, if not the uniqueness, can be regained with a more frequent release schedule of shorter recordings. Thus, though many performers of improvised music may be most satisfied working in a live context, virtues may be still found in releasing recorded works.

The ability to release digital recordings absent a physical counterpart has streamlined the distribution process. Instead of fewer long recordings that are each the culmination of an extended period of development, shorter releases may be made more frequently. Without the arbitrary time constraints that came with fixed physical media such as CDs, releases can be of any length. A live performance is what the performer is doing now; a CD is what was done last year. The new, shorter release schedule provides access to what was done last week, or even earlier today.

These mini-releases can then be distributed for free through a service such as SoundCloud, or put up for sale using a web site such as Bandcamp. With instant global distribution available, geographic, cultural and class barriers fade (Katz 2010), providing direct access to a global audience with which one can build a direct relationship. George Gershwin acknowledged that the composer "has been helped a great deal by the mechanical reproduction of music ... Music is written to be heard, and any instrument that tends to help it be heard more frequently and by great numbers is advantageous to the person who writes it" (Katz 2010). It cannot be overstated how much the Internet has multiplied this advantage.

A live concert has the advantage of immediacy. A recording session has the advantage of reflection. One can regain a bit of the lost connection to the audience while taking advantage of an editing process that selects the best moments for distribution. *Character Weekend*, with its short suites of small pieces, is perfectly suited to this strategy. I can hold performance sessions frequently, and release recordings of them as soon as a satisfying whole is ready.

Conclusion

Nikl, or its predecessor Smoothey, have been my primary performance and compositional tools since early 2011. The instruments I have built have been satisfying in performance. I have found them to be both expressive and flexible. In addition to the solo work demonstrated in Character Weekend, I have performed in various ensembles ranging from two to twelve performers, and have been pleased with their suitability in a wide range of situations.

Furthermore, Nikl has proven to be a productive development environment in which to work. The production and modification of new instruments is easy and fast. The simple template XML file format reduces the amount of time between idea and execution and encourages experimentation. Working instruments are realized quickly, allowing for a greater focus on the iterative improvement phase.

The frequent record-and-release strategy that I developed for the distribution of *Character Weekend*, has also proved satisfactory. It has encouraged creativity, and I have enjoyed the quick feedback that is possible with the compressed release schedule. I plan to continue the *Character Weekend* series, as well as developing longer form solo compositions. In addition, I am working with other musicians to release ensemble work in the same manner.

Although Nikl has been in active use in my performance practice for some time, it is still a work in progress. Development is proceeding on the layout engine to allow for more flexibility in the placement of user interface elements. In addition there are some new elements I would like to add--most notably a note slider, a rethinking of the idea of the "keyboard" for touch screen use that borrows from the touch sliders of Don Buchla. I also will soon add the ability to import samples. As excited as I am about the instruments I've built with Nikl, I do miss the ability to use the large library of field recordings I have made over the last decade.

The final step is to make the tools I've been developing available to the public. Nikl still needs further refinement before it is ready to be submitted to the iOS App Store. New templates are in development for use as examples for those who will use Nikl as a building tool, and as instruments for the less technically inclined. Although I prefer to edit the templates in a text editor, not all will be as comfortable editing XML files, so I will need to develop GUI tools in order to expand the potential audience. Finally, extensive documentation and tutorials will be necessary.

The iRTcmix is also nearing release to the general public. The API is stable and the last few bugs in the library are being addressed. I have developed an extensive set of tutorial apps, each of which is designed to each address a single feature of the iRTcmix system, in order to be as friendly as possible to inexperienced developers. Once the documentation is complete, the library will be ready for release.

Works Consulted

Arfib, Daniel, Jean-Michel Couturier, and Loïc Kessous. "Expressiveness and digital musical instrument design." *Journal of New Music Research* 34.1 (2005): 125–136.

Bahn, Curtis, Tomie Hahn, and Dan Trueman. "Physicality and feedback: a focus on the body in the performance of electronic music." *Proceedings of the International Computer Music Conference* (2001): 44–51.

Bailey, Derek. *Improvisation: its nature and practice in music.* New York: Da Capo Press, 1992.

Barrett, Justin L., and Amanda Hankes Johnson. "The role of control in attributing intentional agency to inanimate objects." *Journal of Cognition and Culture* 3.3 (2003): 208–217.

Bijsterveld, Karin. *Mechanical sound: Technology, culture, and public problems of noise in the twentieth century.* Cambridge: MIT Press, 2008.

Blackwell, T. M. "Swarm music: improvised music with multi-swarms," *Proceedings of the Symposium on Artificial Intelligence and Creativity in Arts and Science* (2003): 41–49.

Carlos, Walter. *Switched-On Bach*. Columbia, 1968. LP.

Cage, John. "Credo." *Silence: Lectures and Writings.* Middletown: Wesleyan University Press, 1961. (Originally published in 1937.)

Cars, The. *Shake It Up*. Elektra, 1981. LP.

Depeche Mode. *Speak & Spell.* Mute, 1981. LP.

Fine, Larry. "In the Shop with Bob Moog: A Personal Account." *Acoustic & Digital Piano Buyer* (1993). Available at http://www.pianobuyer.com/articles/moog.html. Accessed 18 May 2013.

Fischer, Tobias. "15 Questions to Nate Wooley." *Tokafi* (2012). Available at http://www.tokafi.com/15questions/15-questions-nate-wooley/. Accessed 21 May 2013.

Gurevich, Michael, Adnan Marquez-Borbon, and Paul Stapleton. "Playing with constraints: stylistic variation with a simple electronic instrument." *Computer Music Journal* 36.1 (2012): 23–41.

Hakken Audio. "Haken Audio Continuum Fingerboard." Haken Audio (2013). Available at http://www.hakenaudio.com/Continuum. Accessed 26 September 2013.

Herndon, Holly. "Embodiment in Electronic Music Performance." MA diss., Mills College, 2010.

Holmes, Thom. *Electronic and Experimental Music: Pioneers in Technology and Composition*. New York: Routledge, 2002.

Holzborn, Damon. "Adams & Bancroft." Accretions, 2004. CD.

Hsu, William. "Using timbre in a computer-based improvisation system." *Proceedings of the ICMC* (2005): 5–9.

Katz, Mark. *Capturing Sound: How Technology Has Changed Music.* Berkeley: University of California Press, 2010.

Kirn, Peter. "Hands On with Korg's <$150 Volcas, Plus Sounds from Its Creator [Direct Audio Samples]." *Create Digital Music* (2013). Available at http://createdigitalmusic.com/2013/05/hands-on-with-korgs/. Accessed 6 May 2013.

Korg. "Monotron Series." Korg USA (2013). Available at http://www.korg.com/monotrons. Accessed 26 April 2013.

Krefeld, Volker, and Michel Waisvisz. "The hand in the web: an interview with Michel Waisvisz." *Computer Music Journal* 14.2 (1990): 28–33.

Madrona Labs. "Hardware." Madrona Labs (2013). Available at http://madronalabs.com/hardware. Accessed 26 September 2013.

Magnusson, Thor. "Designing constraints: composing and performing with digital musical systems." *Computer Music Journal* 34.4 (2010): 62–73.

Morgan, Robert P. *Twentieth-Century Music: A History of Musical Style in Modern Europe and America*. New York: W.W. Norton, 1991.

Novella, Steven. "More Inattentional Blindness." *NeuroLogica Blog* (2012). Available at http://theness.com/neurologicablog/index.php/more-inattentional-blindness/. Accessed 28 April 2013.

Nissenbaum, Helen. "How computer systems embody values." *Computer,* March 2008: 118–120.

Oliver La Rosa, Jaime E. "Theremin in the press: construing 'electrical music'." Unpublished article (2012).

Paine, Garth. "Gesture and morphology in laptop music performance." In *The Oxford Handbook of Computer Music*, ed. Roger Dean. Oxford and New York: Oxford University Press, 2009, 214–232.

Pinch, Trevor J., and Frank Trocco. *Analog days: The Invention and Impact of the Moog Synthesizer.* Cambridge: Harvard University Press, 2002.

Pinch, Trevor J., and Karin Bijsterveld. "' Should one applaud?' Breaches and boundaries in the reception of new technology in music." *Technology and Culture* 44.3 (2003): 536–559.

Rasamimanana, Nicolas, Florian Kaiser, and Frederic Bevilacqua. "Perspectives on gesture–sound relationships informed from acoustic instrument studies." *Organised Sound* 14.2 (2009): 208–216.

Garton, Brad, et al. "RTcmix - An Open-Source, Digital Signal Processing and Sound Synthesis Language." (2013). Available at: http://rtcmix.org. Accessed 16 May 2013.

Ryan, Joel. "Some remarks on musical instrument design at STEIM." *Contemporary Music Review* 6.1 (1991): 3–17.

Smith, Julius O. "Viewpoints on the history of digital synthesis." *Proceedings of the International Computer Music Conference* (1991): 1–10.

de Saint-Exupéry, Antoine. *Wind, Sand, and Stars.* New York: Houghton Mifflin Harcourt, 1992. (Originally published in 1939.)

Waisvisz, Michel. "Riding the sphinx--lines about 'live'." *Contemporary Music Review* 18.3 (1999): 119–126.

Wang, Ge, Jieun Oh, and Tom Lieber. "Designing for the iPad: Magic Fiddle." *Proceedings of* *the International Conference on New Interfaces for Musical Expression* (2011): 197–202.

Wessel, David. "Timbre space as a musical control structure." Computer Music Journal 3.2 (1979): 45–52.

Wessel, David, and Matthew Wright. "Problems and prospects for intimate musical control of computers." *Computer Music Journal* 26.3 (2002): 11–22.

Wishart, Trevor. *On Sonic Art.* Amsterdam: Harwood Academic Publishers, 1996.

Appendix A: RTcmix Resources on the Internet

Official RTcmix Website:
http://rtcmix.org

RTcmix-discuss - a discussion list for RTcmix:
http://music.columbia.edu/mailman/listinfo/rtcmix-discuss

Twitter:
http://twitter.com/rtcmix
@rtcmix

Appendix B: Dixey Nikl Template

```
<?xml version="1.0" encoding="utf-8"?>
<nikl>

<meta>
   <name>Dixey</name>
   <author>cnco</author>
   <description>FM synth.</description>
</meta>

<scores>

   <setupscore>

      // ***************** SUNDRY SETUP VARS *****************
      print_on(3)
      control_rate(3000)
      dur = 45000

      soundingNotes = {}
      for (i = 0; i < 11; i += 1)
      {
         soundingNotes[i] = 0
      }
      touchedNotes = {}
      for (i = 0; i < 11; i += 1)
      {
         touchedNotes[i] = 0
      }

      TurboPreamp = pfield_TurboPreamp

      lfoRoutePitch = (pfield_LFORoute - 1) * -1
      lfoRouteFilter = pfield_LFORoute

      // ***************** FM Depth (accel X) *****************

      fmDepthClipped = makefilter(pfield_accelX, "clip", -0.1, 0.9)
      fmDepthShifted = fmDepthClipped + .1
      fmDepthSquared = fmDepthShifted * fmDepthShifted
      fmDepthRanged = makefilter(fmDepthSquared, "fitrange", 0, 10)
      fmDepth = makefilter(fmDepthRanged, "smooth", 60)

      // ***************** Global Filter Freq (accel Y) *****************

      globalFilterFreqClipped = makefilter(pfield_accelY, "clip", -.9, .4)
      globalFilterFreqScaled = (globalFilterFreqClipped + .9) * .765
      globalFilterFreqSquared = globalFilterFreqScaled * globalFilterFreqScaled
      globalFilterFreqRanged = makefilter(globalFilterFreqSquared, "fitrange", 40,
12000)
```

```
      globalFilterFreqSmoothed = makefilter(globalFilterFreqRanged, "smooth", 60)

      // ***************** Note XY Controller *****************

      notePoly = {}
      for (i = 0; i < 11; i += 1)
      {
         notePoly[i] = makefilter(pfield_PolyNoteX[i], "smooth", 60)
      }
      filterPoly = {}
      for (i = 0; i < 11; i += 1)
      {
         filterPoly[i] = makefilter(pfield_PolyNoteY[i], "smooth", 60)
      }
      volumePolyClipped = {}
      volumePolyScaled = {}
      volumePolySquared = {}
      volumePoly = {}
      for (i = 0; i < 11; i += 1)
      {
         volumePolyClipped[i] = makefilter(pfield_PolyNoteArea[i], "clip", .25, .6)
         volumePolyScaled[i] = (volumePolyClipped[i] + .1) * 1.38
         volumePolySquared[i] = volumePolyScaled[i] * volumePolyScaled[i]
         volumePoly[i] = makefilter(volumePolySquared[i], "smooth", 50)
      }

      // ***************** NOTE *****************

      noteRangeLow = makeconverter(pfield_NoteRange + 4.09, "cpspch")
      noteRangeHigh = makeconverter(pfield_NoteRange + 7.09, "cpspch")
      noteRange = noteRangeHigh - noteRangeLow

      // ***************** WAVETABLES *****************

      saw = 0 square = 1 tri = 2 sine = 3 noise = 4
      sawAA1 = 5 sawAA2 = 6
      triAA1 = 7 triAA2 = 8

      waveTables = {}
      waveTables[saw] = maketable("wave", "interp", 1000, "saw")
      waveTables[square] = maketable("wave", "interp", 1000, "square")
      waveTables[tri] = maketable("wave", "interp", 1000, "tri")
      waveTables[sine] = maketable("wave", "interp", 1000, "sine")
      waveTables[noise] = maketable("random", 1000, "linear", -1, 1)

      waveTables[sawAA1] = maketable("wave3", "interp", 1000, 1,1/1,0, 2,1/2,0,
3,1/3,0, 4,1/4,0, 5,1/5,0, 6,1/6,0, 7,1/7,0, 8,1/8,0, 9,1/9,0, 10,1/10,0, 11,1/11,0,
12,1/12,0, 13,1/13,0, 14,1/14,0, 15,1/15,0, 16,1/16,0, 17,1/17,0, 18,1/18,0,
19,1/19,0, 20,1/20,0, 21,1/21,0, 22,1/22,0, 23,1/23,0, 24,1/24,0)

      waveTables[sawAA2] = maketable("wave3", "interp", 1000, 1,1/1,0, 2,1/2,0,
3,1/3,0, 4,1/4,0, 5,1/5,0, 6,1/6,0, 7,1/7,0, 8,1/8,0)
```

```
      waveTables[triAA1] = maketable("wave", "interp", 1000, "tri")
      waveTables[triAA2] = maketable("wave3", "interp", 1000, 1,1/1,0, 3,1/9,0,
5,1/25,0, 7,1/49,0)

      sawAAMap = { 0, 5, 5, 5, 6 }
      triAAMap = { 2, 2, 2, 7, 8 }

      // ***************** LFO *****************

      sawMapTable = maketable("literal", "nonorm", 4, 1, 0, 0, 0)
      squareMapTable = maketable("literal", "nonorm", 4, 0, 1, 0, 0)
      sineMapTable = maketable("literal", "nonorm", 4, 0, 0, 1, 0)
      noiseMapTable = maketable("literal", "nonorm", 4, 0, 0, 0, 1)

      sawMap = makefilter(pfield_LFOWaveSelect, "map", sawMapTable, 0, 3)
      squareMap = makefilter(pfield_LFOWaveSelect, "map", squareMapTable, 0, 3)
      sineMap = makefilter(pfield_LFOWaveSelect, "map", sineMapTable, 0, 3)
      noiseMap = makefilter(pfield_LFOWaveSelect, "map", noiseMapTable, 0, 3)

      // ***************** NOTE BUSSES *****************

      waveOutBus = {}
      waveOutBus[0] = "aux 1 out"
      waveOutBus[1] = "aux 2 out"
      waveOutBus[2] = "aux 3 out"
      waveOutBus[3] = "aux 4 out"
      waveOutBus[4] = "aux 5 out"
      waveOutBus[5] = "aux 6 out"
      waveOutBus[6] = "aux 7 out"
      waveOutBus[7] = "aux 8 out"
      waveOutBus[8] = "aux 9 out"
      waveOutBus[9] = "aux 10 out"
      waveOutBus[10] = "aux 11 out"

      moogInBus = {}
      moogInBus[0] = "aux 1 in"
      moogInBus[1] = "aux 2 in"
      moogInBus[2] = "aux 3 in"
      moogInBus[3] = "aux 4 in"
      moogInBus[4] = "aux 5 in"
      moogInBus[5] = "aux 6 in"
      moogInBus[6] = "aux 7 in"
      moogInBus[7] = "aux 8 in"
      moogInBus[8] = "aux 9 in"
      moogInBus[9] = "aux 10 in"
      moogInBus[10] = "aux 11 in"

      for (i = 0; i < 11; i += 1)
      {
         bus_config("MOOGVCF", moogInBus[i], "aux 16 out")
         /*   (outsk, insk, dur, amp mult, inputchan, pan, bypass, freq, q) */
         MOOGVCF(0, 0, dur, 1, 0, 0, 0, filterPoly[i], 0.01)
      }
```

```
		// ***************** GLOBAL DISTORTION *****************

		turboPreampInverse = (TurboPreamp - 1) * -1
		bus_config("DISTORT", "aux 16 in", "aux 17 out")
		/* (outsk, insk, dur, amp, disttype, preamp, lowpassfreq[, inputchan, pan
{default: 0.5}, bypass]) */
		DISTORT(0, 0, dur, (turboPreampInverse * turboPreampInverse * 2) + 1, 1,
(TurboPreamp * TurboPreamp * 100) + 1, 0)

		// ***************** GLOBAL FILTER *****************

		filterlfoSaw = makeLFO(waveTables[saw], pfield_LFOFreq, 0, pfield_LFOAmount)
		filterlfoSquare = makeLFO(waveTables[square], pfield_LFOFreq, 0,
pfield_LFOAmount)
		filterlfoNoise = makeLFO(waveTables[noise], pfield_LFOFreq / 500, 0,
pfield_LFOAmount)
		filterlfoSine = makeLFO(waveTables[sine], pfield_LFOFreq , 0, pfield_LFOAmount)
		filterlfoCombined = ((filterlfoSaw * sawMap) + (filterlfoSquare * squareMap) +
(filterlfoNoise * noiseMap) + (filterlfoSine * sineMap) + 1) / 2

		globalFilterLFOed = globalFilterFreqSmoothed - (globalFilterFreqSmoothed *
filterlfoCombined * lfoRouteFilter)
		globalFilterFreq = makefilter(globalFilterLFOed, "clip", 20, 20000) // protect
from going out of range

		bus_config("MOOGVCF", "aux 17 in", "out 0 - 1")
		//  (outsk, insk, dur, amp mult, inputchan, pan, bypass, freq, q)
		MOOGVCF(0, 0, dur, .7, 0, 0.5, 0, globalFilterFreq, pfield_GlobalFilterQ)

	</setupscore>

	<onscore>

		alreadySounding = bus_exists(Touch)

		if (alreadySounding != 1)
		{
			lfoAmountScaled = pfield_LFOAmount * 4
			lfoFreq = pfield_LFOFreq * (random() / 10 + 1)
			lfoSaw = makeLFO(waveTables[saw], lfoFreq, 0, lfoAmountScaled)
			lfoSquare = makeLFO(waveTables[square], lfoFreq, 0, lfoAmountScaled)
			lfoNoise = makeLFO(waveTables[noise], lfoFreq / 500, 0, lfoAmountScaled)
			lfoSine = makeLFO(waveTables[sine], lfoFreq , 0, lfoAmountScaled)
			lfoCombined = (lfoSaw * sawMap) + (lfoSquare * squareMap) + (lfoNoise *
noiseMap) + (lfoSine * sineMap)
			noteScaled = (notePoly[Touch] * noteRange) + noteRangeLow

			if (var_OscWaveSelect == 0) { carWaveIndex = sawAAMap[var_NoteRange] }
			else if (var_OscWaveSelect == 1) { carWaveIndex = triAAMap[var_NoteRange] }
			else { carWaveIndex = sine }


			ampEnv = makeconnection("pfbus", Touch, 0.0)
			bus_link(Touch)
```

```
          wiggleAmp = 6000 * ampEnv * (volumePoly[Touch])
          wiggleCarFreq = noteScaled + (noteScaled * (lfoCombined / 4) *
lfoRoutePitch)
          wiggleModDepth = 2
          wiggleFilterType = 0
          wiggleSteepness = 1
          wiggleBalance = 0
          wiggleCarWave = waveTables[carWaveIndex]
          wiggleModWave = waveTables[sine]
          wiggleModFreq = noteScaled * pfield_FMDetune
          wiggleFMDepth = fmDepth
          wiggleLowPassFreq = 22000
          wigglePan = .5

          bus_config("WIGGLE", waveOutBus[Touch])
          WIGGLE(0, dur, wiggleAmp, wiggleCarFreq, wiggleModDepth, wiggleFilterType,
wiggleSteepness, wiggleBalance, wiggleCarWave, wiggleModWave, wiggleModFreq,
wiggleFMDepth, wiggleLowPassFreq, wigglePan)
      }

      envAttack = .05
      ampEnvStart = maketable("line", "dynamic", 1000, 0,"curval", envAttack,1)
      PFSCHED(0, envAttack, Touch, ampEnvStart)
      soundingNotes[Touch] = 1
      touchedNotes[Touch] = 1

   </onscore>

   <offscore>

      alreadySounding = bus_exists(Touch)

      if (alreadySounding && var_Hold != 1)
      {
          ampEnvEnd = maketable("line", "dynamic", 1000, 0,"curval",
var_AmpEnvRelease,0)
          PFSCHED(0, var_AmpEnvRelease, Touch, ampEnvEnd, 1)
          soundingNotes[Touch] = 0
      }
      touchedNotes[Touch] = 0

   </offscore>

   <releasescore>

      if (!var_Hold)
      {
          for (i = 0; i < 16; i += 1)
          {
             if (soundingNotes[i] && !touchedNotes[i])
             {
                ampEnvEnd = maketable("line", "dynamic", 1000, 0,"curval",
var_AmpEnvRelease,0)
```

```
                PFSCHED(0, var_AmpEnvRelease, i, ampEnvEnd, 1)
                soundingNotes[i] = 0
            }
        }
    }

    </releasescore>

</scores>

<interface>

    <accelerometer>medium</accelerometer>

    <zone1>

        <row>
            <type>radio</type>
            <name>LFOWaveSelect</name>
            <label>Saw, Sqr, Sin, Rnd</label>
            <range>4/0</range>
            <color>yellow</color>
            <parameters>pfield</parameters>
        </row>

        <row>
            <type>radio</type>
            <name>LFORoute</name>
            <label>Pitch, Filter</label>
            <range>2/0</range>
            <color>yellow</color>
            <parameters>pfield</parameters>
        </row>

        <row>
            <type>slider</type>
            <name>AmpEnvRelease</name>
            <label>Amp Envelope Release Time</label>
            <range>.05/4/1/.05</range>
            <color>orange</color>
            <parameters>var</parameters>
        </row>

    <row>
            <type>radio</type>
            <name>OscWaveSelect</name>
            <label>Saw, Tri, Sin</label>
            <range>3/0</range>
            <color>blue</color>
            <parameters>var</parameters>
        </row>

        <row>
            <type>toggle</type>
```

```
      <name>TurboPreamp</name>
      <label>Turbo</label>
      <range>0/1/0/0</range>
      <color>green</color>
      <parameters>pfield</parameters>
   </row>

   <row>
      <type>toggle</type>
      <name>Hold</name>
      <label>Hold</label>
      <onscore>releasescore</onscore>
      <color>gray</color>
      <parameters>var</parameters>
   </row>

</zone1>

<zone2>

   <row>
      <type>slider</type>
      <name>LFOFreq</name>
      <label>LFO Rate</label>
      <range>.02/160/2/1</range>
      <color>yellow</color>
      <parameters>pfield</parameters>
   </row>

   <row>
      <type>slider</type>
      <name>LFOAmount</name>
      <label>LFO Amount</label>
      <range>0/1/0/0</range>
      <color>yellow</color>
      <parameters>pfield</parameters>
   </row>

   <row>
      <type>slider</type>
      <name>GlobalFilterQ</name>
      <label>Filter Q</label>
      <range>0/.9/1/0</range>
      <color>red</color>
      <parameters>pfield</parameters>
   </row>

   <row>
      <type>slider</type>
      <name>FMDetune</name>
      <label>FM Detune</label>
      <range>.25/4/1/2.01</range>
      <color>teal</color>
      <parameters>pfield</parameters>
```

```
        </row>

        <row>
            <type>radio</type>
            <name>NoteRange</name>
            <label>21, 33, 45, 57, 69</label>
            <range>5/2</range>
            <color>gray</color>
            <parameters>both</parameters>
        </row>

    </zone2>

    <zone3-4>

        <row>
            <type>xypoly</type>
            <name>PolyNote</name>
            <label>safd</label>
            <onscore>onscore</onscore>
            <offscore>offscore</offscore>
            <rangex>0/1/1/0</rangex>
            <rangey>20/14000/2/0</rangey>
            <color>gray</color>
            <parameters>pfield</parameters>

        </row>

    </zone3-4>

</interface>

</nikl>
```