California Institute of the Arts

Touchpoint: Effects Processing as a Method of Synthesis and a Performance Practice

by

Nicholas Khan Suda

A thesis submitted in partial fulfillment for
the degree of Master of Fine Arts

in the
Herb Alpert School of Music
Music Technology: Interaction, Intelligence & Design

Supervisory Committee

Owen Vallis, Ph.D

Mentor

Dr. Ajay Kapur

Committee

Jordan Hochenbaum, Ph.D

Committee

Abstract

This thesis suggests that effects processing has developed a fundamental role in contemporary music styles. It has become both integral and widespread enough to the point where it could be considered a method of synthesis unto itself. And yet, possibly because of the nature of their interfaces, effects processors have yet to be appropriately embodied by themselves in live performances of music.

Creative effects processing can often be very elaborate, requiring the use of many different devices in a long chain, and so the process is mostly pre-prepared in execution. If we think of creative effects processing as a mode of synthesis, then what are its basic building blocks? The increasing use of this kind of sound design in the compositional process has produced a problematic representation of such music in a live concert context.

Live concerts imply the real-time performance of instruments as a complete portrait of the music being performed. So, how can we wrench the post-processing element away from being the additional responsibility of a sound engineer, invisibly managing knobs, faders, and pre-recorded cues? Can this important facet of the concert be represented as a musical instrument? Can it be performed in real-time amongst everything else onstage?

What sort of interface befits this kind of performance? Does the style of music itself change by adding this instrument into an ensemble? Furthermore, how did we get here? What are some devices from the past that have culminated toward this idea of a performative effects processor? What are the shortcomings of the use of these earlier devices onstage? Why are they "insufficient?"

In this thesis is presented a new musical instrument called *Touchpoint*, which exists as several pieces of software. Some of this software runs on a computer, which is responsible for dynamically arranging the use of effects into on-the-fly chains, acting as the "effect-based synthesizer." Other software runs on a tablet device, which acts as the interface of the instrument. The touchscreen of the tablet becomes to the "effect-based synthesizer" what the fretboard is to the guitar; what the mouthpiece is to the trumpet; what the keyboard is to the piano.

In order to corral the complexity of dynamically chaining many smaller parts together as an act that is as immediate as playing a musical instrument can be, each of the available parts within this system is programmed as a piece of smaller software code that executes digitally on the host computer. Why is this model - a software engine divorced from a software interface - advantageous over a single piece of "closed" hardware? What sort of technological and historical landmarks in computer music software have led to this design style?

The technical construction of this instrument is then described in further detail for several benefits. A broader simplification about signal processing that was discovered in *Touchpoint*'s development is one aspect. The technical justification behind thinking about the use of effects processors as a form of electronic music synthesis is another element. Several intriguing use cases that reveal the instrument's versatility are described thereafter, which is aided by an explanation of the design of each of the individual processors. Furthermore, contemplating the disconnection between interface and engine reveals the highly flexible placement of the instrument within an ensemble of musicians, which is a component that is arguably unique to *Touchpoint*.

The author then reflects on several concerts featuring *Touchpoint* that have tried some of these diverse configurations out, discussing lessons learned, and inspirations for future design changes in service of these lessons. The general plans for the future of the instrument are discussed in a conclusion, which lays the groundwork for a long-term relationship in developing and improving *Touchpoint*.

Acknowledgments

I would like to thank Ajay Kapur for being the real deal. He is a true-to-form opportunity creator, empowerer, mentor, and believer. He finally got me to learn real text programming in a way that was tailor-made for my experience level, after I had been saying that I wanted to learn my whole life without doing so. He proved to me that straddling between the academic, artistic, and commercial worlds is possible, and that it's a really cool position to be in. He saw that I had a momentum, and convinced me to stop slowing it down and to come to CalArts - less I risk losing it - and pushed me harder than I thought possible.

I would like to thank my friends and advisors Owen Vallis and Jordan Hochenbaum for being the first people I met who showed me that I'm not the only one that this kind of stuff is important to. Kindred, modest, humble spirits who insist on letting the person's work speak for the person. I don't think I'll ever be quite as even-tempered as you two, but I hope that I can be as accomplished and helpful. Thanks for giving me classes to teach, thanks for sending me to Berlin, and thanks for helping me define and attain the skillset - and the portfolio - that I always wanted to have.

Thanks to Dr. B for turning another 18 year-old Aphex Twin fan like me into an aspiring academic and exposing me to the history of academic computer music at so many angles. Copying large swaths of your lectures really makes it sound like I know what I'm talking about to some people. Your live remix of "*Synapse*" with a Launchpad, a Wii Remote and a few instances of *Stutter Edit* at the MIT Media Lab for Barry Vercoe's retirement was a formative moment for me. I will keep pushing forward thanks to the spark you lit for me.

Thank you to Max Zagler, Egbert Jürgens, André Estermann, Christiane Küstermann, Franziska Hollingworth, Mike Gao, Vadim Zavalishin, and every other co-worker and friend that I met in

Berlin during my time at Native Instruments. Your help and advice was invaluable and life-changing.

Thanks to Colin Honigman and Chris Knollmeyer for getting to know my instrument, and performing in a *Touchpoint Series* concert with me. I use both of you as role models for patience and hard work to aspire to.

I would like to thank Benn Jordan and Richard Devine for their influence and friendship, starting barely in my teenage years up to the present day. It's strange to have your famous idols also be acquaintances as you're growing up, and I don't think too many other people have had quite as profound an influence on my sensibilities as these folks.

I would like to thank my life-long best friend John Platter, and my fiancée Amber Lepley, for their love, and their patience, and their investment. Here's to many more years. I love you.

Shout out to William Basinski for writing *The Disintegration Loops* series, and Takagi Masakatsu for *Journal for People*. These two records were the soundtracks of writing this document, especially in the Newhall location of the Santa Clarita Public Library, which also deserves some recognition as Thesis Central.

Last, and most importantly, thank you to my parents, who never stopped buying me the toys that make noise when I was little, who let me make racket as loud and for as long as I wanted, without ever stopping me, and who have provided an unreasonable amount of support and belief and opportunity for me to live my life working from the weird impetus that has motivated my entire path so far.

Contents

List of Figures

List of Tables

Chapter 1

Introduction: The Music of Effects Processing

(For the purposes of this document, "effects processing" is defined as the objectification of a sound source through various electronic modification techniques, via a secondary process. This can describe techniques with constantly streaming input and output, as well as techniques that capture a duration of incoming sound before manipulation, whether in advance or on-the-fly.)

This chapter describes the increasingly important role of creative effects processing in contemporary music. The computerization of music recording is identified as the key accelerator of the relationship between effects processing and the sound of contemporary music. Virtualization, which is an attribute of computerization, is also discussed for its disembodying effect on musical instruments, and the problem that it creates in live performances of music.

A series of examples in which the creative application of effects processing influenced the creation of a new style of music are detailed. Then, a particular musical genre - where the application of automated effects processing has become so elaborate that it describes the genre as a whole - is examined. Altogether, it is posited that computer-based music production has obfuscated the nature of live performance in an as-yet unsolved way. Consequently, the need to re-consider effects processing as a musical instrument-like facet of concert music is then suggested.

An argument is made that a system of effects processors could be made "performative" in order to more accurately classify the effects processor as a musical instrument. Formal methods of musical instrument classification are incorporated, in order to discuss the additional ways in which classifying a processor as a musical instrument is problematic.

A survey of existing effects processing interfaces is offered, including a discussion of their shortcomings as musical instruments. In reflecting back on the effects of computerization, a potential way forward in thinking about how to re-embody the effects processor as a new kind of musical instrument is suggested.

1.1 The computerization of music recording... and performance

"*But the next Beck or Jimi Hendrix may be sitting in a bedroom with a computer and a guitar at this very moment, creating sounds that will be sprung on the world in the next year or two. "What's going to happen," [...] "is that in three or four years we'll wake up one day and say, 'Hey, wait a minute, I just realized that everything in the Top 10 was done inside a computer.'"* "

- "Technology Puts the Recording Studio on a Hard Drive", New York Times, 1999 (Hearst)

While the electro-acoustic manipulation of musical material is as old as the advent of audio recording itself, the creative use of signal processors in music - in composition, in recording, and in performance - has drastically accelerated as a result of music recording's computerization. Devices that cannot - kinesthetically speaking - be performed like a musical instruments have been used in concerts for much of the 20th century. Big boxes that contain waveform generators, reel-to-reel tape players, ring modulators, tape loops, echoes, and delays were feasibly (if impractically) used in concert. These devices are treated as an array of machines whose routing was somewhat inflexibly pre-arranged. Their controls are manipulated onstage in a musical way.

The nature of recording electronic music of the time - of capturing pre-prepared performances with these tools, and editing their recordings on magnetic tape - was not so disconnected from the offerings of live concert recitation that nothing could be gained from some sort of onstage performance. The kinds of interactions with other musical instruments that can be performed inside of or alongside this electro-acoustically manipulated content are rich and musically conversational.

At the turn of the 21st century, the audio recording process shifted from magnetic tape to audio files on the hard disk drive of a personal computer. At the same time, the processing power of these computers became sufficient enough that most - if not all - recording tools became virtualized. From the aforementioned signal generators and processors that augmented musical performances creatively, to musical instruments themselves, all musical material could eventually be convincingly replaced with automated instructions given to software emulations of physical devices. All music that could be described as traditional instrumental performances could be emulated digitally.

It is now not only easier and faster to create complex musical moments - moments that can even appear to be entirely acoustic - but it is also trouble-free and nearly instantaneous to render, record and edit it in very complex ways. Thanks to the advent of parametric automation and non-linear editing, music production became much more *offline* in nature. Thus, the creative use of DSP (Digital Signal Processing) has accelerated both in the studio and in concert, despite a lack of a physical representation of the essential facets of the music in concert recital.

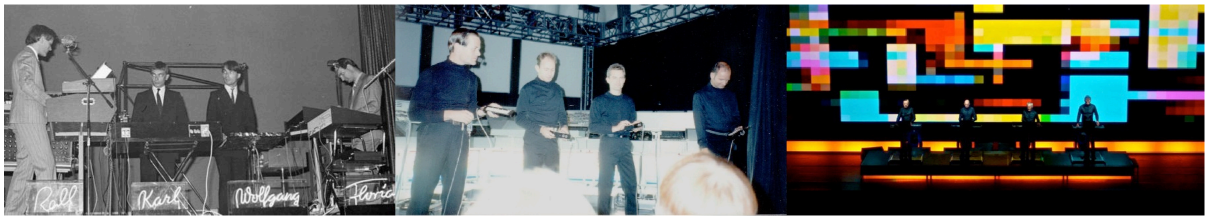


Figure 1. *Kraftwerk*'s stage setup in the 1970s, 1990s and 2010s

In Figure 1. *Kraftwerk*'s stage setup in the 1970s, 1990s and 2010s, the growing complexity of creative signal processing in electronic music that has happened as a result of computerization - as well as the ever-shrinking, digital transformation of the physical footprint of musical equipment - is documented in the changes in the onstage rig of one electronic music band over a forty-year period. What started in the early days of the 1970s as an array of keyboard synthesizers, tape delays, and acoustic instruments like flutes, shifted into an entire recording studio's worth of rack-mounted signal processing gear on wheels by the 1990s. By the 2010s, both the instruments and the signal processors lived onstage as programs running on laptop computers, visible only to the performing musicians. Where once live performances used to define the nature of a recording, now the process of recording dictates the content of a live performance.

1.2 Effects processors and new musical genres

Much of the popular music of the 20th century is significantly defined by the creative use of signal processors. In many cases, many new genres of music came to be defined even more by their use of certain effects processes, more so than the style of the instrumental performances.

For example, from the spring-reverberated surf rock of the 1950s, to the plate-reverberated progressive rock of the 1970s, to the digitally reverberated (and gated) ballads of the 1980s, there are many genres of rock music which are at least partially defined by their use of state-of-the-art reverberation technology. But the importance of electronic reverberation is but one of the many sounds of creative effects processing that has penetrated mainstream music culture throughout recent history.

The creative manipulation of a delayed looping-tape playback is one of the most crucial elements of Jamaican dub music of the 1970s, distinguishing it from the reggae music that preceded it. The *carrier and modulator*-driven techniques of the *vocoder* and the *talk box* have entered the public consciousness several times since the 1970s, from Peter Frampton to Roger & Zapp, Kraftwerk to Daft Punk. This same sort of idea recently reared its head again in the 2000s, with the Auto-Tune sound made popular by Cher in her song "Believe," and then later by R&B artists T-Pain and Akon. Turntables and samplers dominated the sounds of the 1990s. The record-scratched turntable drum fills of bands like Incubus and the sampler solos of artists like DJ Shadow introduced the idea of the performative manipulation of pre-recorded sound playback into the lexicon of musical performance techniques.

The innovations of computer-based recording in the 21st century were manifest in computer programs called DAWs (Digital Audio Workstations). The processor power of personal computers of the 2000s was now sufficient enough to handle real-time, high-quality digital audio streaming, sound synthesis, and signal processing. As a result, new software companies created an entirely new industry based on the emulation of musical equipment to be used inside the DAW.

These products, such as the ones seen in Figure 2, came to be known as "plug-ins," due to the modular nature of their inclusion within the software that they work within. Once the emulation of plug-ins reached a certain point of sonic indistinguishability from their physical counterparts, this removed the necessity for many devices, be they musical instruments or effects processors, to even exist physically. Because the emulations are just instances of software code, they can be instantiated into infinity, in any combination desired.

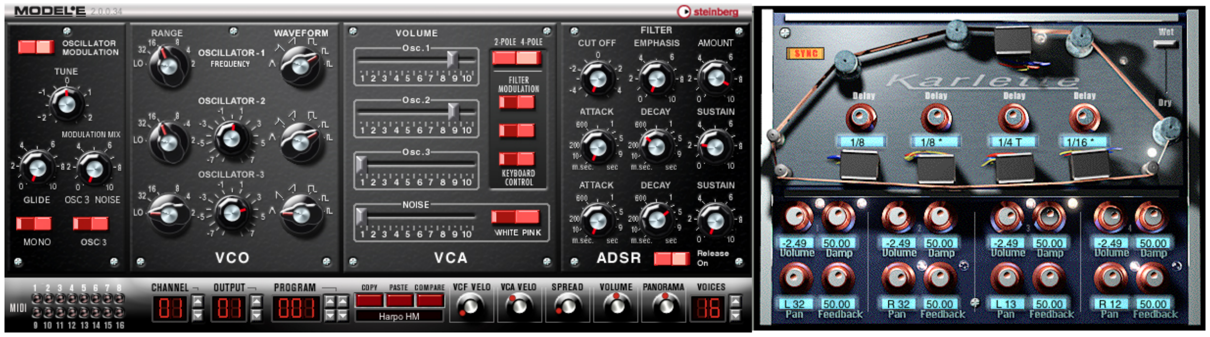


Figure 2. Steinberg's *Model-E*, left, and *Karlette*, right (1996)

Rather than physically existing on a slice of magnetic tape, the recording process now lives in hard disk storage, which is modifiable to great precision. In addition to being heard, each recording was now also perceived visually, as a waveform on the computer's display. Each recording could be iterated into new audio files, without destructively degrading the original recording. Features such as timeline-based automation and the ability to infinitely micro-edit and modify a single audio recording to the heart's content invited the creative use of signal processors to become much more elaborate than before.

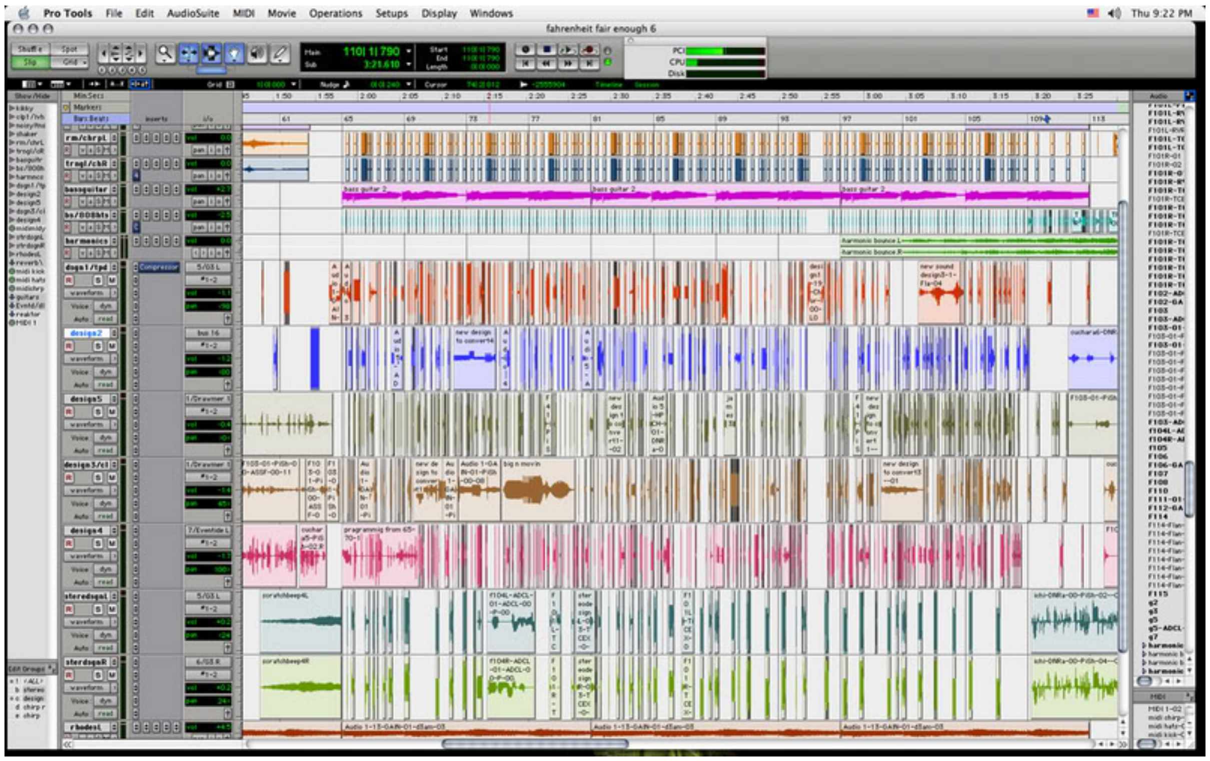

Pro Tools File Edit AudioSuite MIDI Movie Operations Setups Display Windows
fahrenheit fair enough 6
Thu 9:22 PM

Figure 3. Pro Tools session, "Fahrenheit Fair Enough" by Telefon Tel Aviv (2001)

This emergent culture of infinitely-precise audio editing and the potentially endless opportunity for iterative sound design cumulated in the glitchy, "post-digital" (Cascone) style of IDM (Intelligent Dance Music) in the 1990s and 2000s. As demonstrated in Figure 3, artists such as Aphex Twin, Squarepusher, and Telefon Tel Aviv placed destructive processing and re-processing of the same material, offline and via the personal computer, as the central mechanic of composition. Though experiencing music is an art form that occurs over time, in the case of IDM, the act of recording it can now be thought of more like putting together an abstract, pointillist, mixed-media collage made of variously processed bits and pieces of unrecognizable samples, rather than painting a realistic oil portrait made of long, instrumental performances.

In some respects, this style is so transformative to its original source material that the source itself is totally irrelevant. A certain sound's "starting place" could have been a blast of white noise just as much as it could have been a trumpet performance recorded in-studio by a performer. In this music, the domain of live performance is pushed to a total impossibility, since nothing about the composition was realized performatively. It is the music of virtuosic sound design and computer editing, rather than of virtuosic instrumental performance.

IDM is an extreme example of how the DAW - and the infinite signal-processing potential it allows for - has changed music composition. However, no style of mainstream popular music has survived the switch to computer-based recording without somehow incorporating the tools of computer-based music. Whether through pitch correction, drum sample replacement, using a virtual piano or algorithmic bass player, or framing the hook of a song around a signal processing and editing-based technique, some (or many) facets of DSP has augmented popular instrumental music moving forward.

Now, elaborate computer-based techniques are everywhere in commercial music. This includes sounds like the *varispeed* record drop effect, which has become incredibly widespread in modern dance pop like Black Eyed Peas, Lady Gaga, and Ke$ha ("Tik Tok"). The buffer-based *glitch edit* effect - which involves the rapid repetition of a moment of audio so small that it would be impossible to cut with magnetic tape - permeated pop music production of the 2000s, from boy bands like N*SYNC ("Pop") to the rock group Linkin Park ("In the End"). These techniques are also found in popular musics outside of the Western world, from Indian and Middle Eastern to African and Far East Asian styles. They are one of the core characteristics of contemporary popular music worldwide.

Consequently, popular music, which is typically presented to large concert audiences as ensemble music, has developed a growing reliance upon virtual technologies that have no physical form. In an era where so much popular music is defined by computer-based music production techniques that do not involve real-time performance, a lot of music is presented somehow reconstructed, mimed, or re-imagined in the live concert context (de Moraes; Sanneh). Elements which cannot be overtly pinpointed to an instrumental origin are blanket categorized as "magic," (Schloss) or glossed-over in high-budget production spectacles, like that seen in Figure 4.

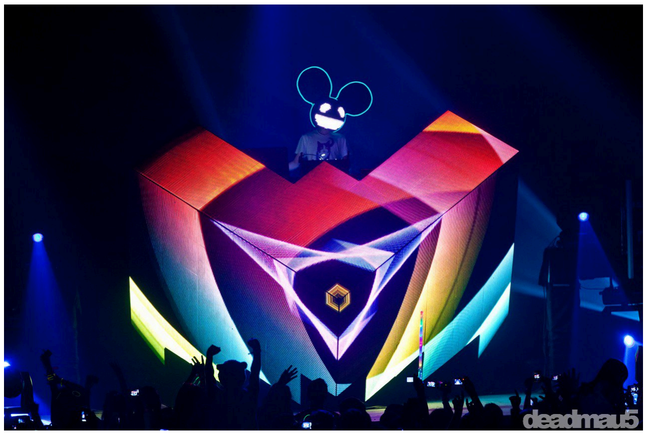


Figure 4. Deadmau5 in concert (2013)

The increasing role of immersive visuals is apparent. Again, the images of Kraftwerk in Figure 1 are referenced. Where there was nothing more than each band member's name in neon lights in the 1970s, there are now four screens of synchronized animations by the 1990s. Kraftwerk's modern day performances now issue 3D glasses to every member of the audience. These performances are dominated by an immersive visuals experience that is operated by one of the four core band members as his sole responsibility. As the physical act of reciting this music has become progressively more disembodied, the importance of accompanying visual content has become louder.

Given the evolution of live concert performances away from instrumental performance and closer to pure spectacle, there is therefore the desire to come back to performance somehow. The musical vocabulary of creative effects processing - which is at the heart of computerized music production - has matured to the point where many distinct sounds can be described and attributed to certain specific devices or techniques. Therefore, one could suggest that creative effects processing could be placed at the center of a new instrumental performance, within its own form factor that can start and stop with a gesture, like any other musical instrument can do.

Rather than increasing the responsibilities of the front-of-house engineer - as has traditionally been the delegation of post-processing effects - a musician performs the effects processing, either in an ensemble, or as a solo instrumentalist. However, classifying a collection of effects processors as a traditional musical instrument is taxonomically problematic.

1.3 Fitting into the taxonomy of musical instruments

In ethnomusicology, the *Hornbostel-Sachs system* describes categories of all musical instruments by the element that produces acoustic fluctuations. (See Figure 5.) *Aerophones*, like flutes and trumpets, require the manipulation of a performer's breath to drive a chamber of wind. Percussion instruments, which are most often *membranophones* (like bass or snare drums) or *idiophones* (like wood blocks or triangles), require striking, stretching, and/or bending. *Chordophones* have strings that must somehow be disturbed, through strumming, plucking, electromagnetic resonation, bowing, or transduction. (von Hornbostel and Sachs)

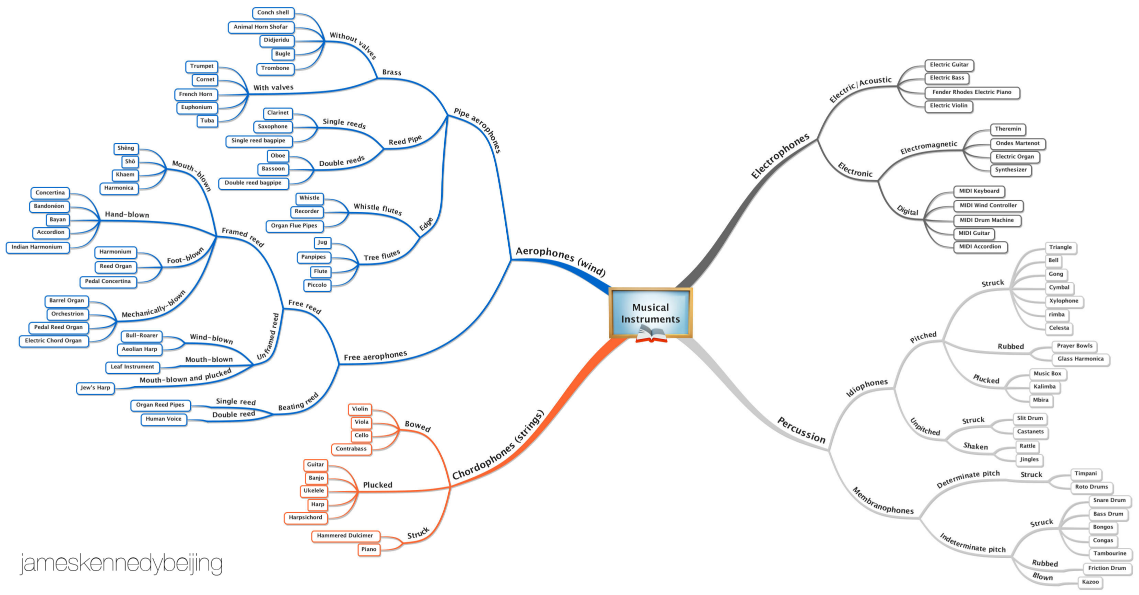


Figure 5. Tree visualization of the *Hornbostel-Sachs* instrument classification system

Essentially, any electric or electronic device used for musical purposes is appended to the *Hornbostel-Sachs system* as an *electrophone*, bound only by the necessity of electricity to operate a loudspeaker transduction process for sonification. This covers oscillator-based, synthesis-driven instruments, like synthesizers, the electric organ, and the theremin, where no physical components are moving in the sonification process. In these instruments, the throttling of an input current by components and chipsets produces waveforms that are amplified to an output.

Electrophones can also describe playback-driven systems, like samplers, turntables and CD players. These devices manipulate the playback of a pre-recorded audio signal. (Personal computers can function in either of these capacities, or also as a hybrid.) Effects processors do not use pre-recorded audio material in the same way that playback systems do, nor do they generate periodic signals given control input, like synthesizers.

Both *synthesizer* and *playback device*-type *electrophones* have mature interfaces that have allowed users to expressively manipulate a sonic result in real-time. Be they the vinyl record and the line mixer of the turntable, the piano keyboard of the synthesizer, or the floating hands of the theremin, traditionally defined electrophones behave like musical instruments that can be dynamically actuated. They typically have well-defined interface features, like keyboards or trigger pads. However, because most effects processors are not designed to be used in a performative, real-time fashion. Their control sets are designed to be as parametrically explicit as possible, so that the user can precisely prepare the result of individual moments in time. Effects processors tend to suffer from what has been described by sound artist Mark Trayle as "the cockpit control problem": Every single possible parameter is broken out for conditional adjustment as a basic knob or button (Trayle).

Be they a reverb unit, a delay, a flanger, a phaser, a distortion unit, a looper, or any others, interaction with effects processors is much more fragmented and static than the manipulation of a musical instrument, because they have no gestural mechanic of actuation. They *augment* an incoming signal that is passively passed to them in a *send-and-return* fashion, usually by a mixer of some description. There is no blowing into a mouthpiece; no stroke of an ivory key; no strike of a drumhead to make them do something. Their input is bypassable in a binary fashion, and their output is controlled only by controlling the mix of signal being sent to and from them.

1.4 Differentiating from precedent signal processors

As mentioned 1.1 and 1.2, the shortcomings of the "cockpit control"-style interface of the prototypical signal processor has not stopped their manifestation into many diverse form factors which have attempted to re-frame them more effectively as such throughout recent history. What follows is an overview of several notable examples of objects or environments that have been used to perform effects processing in a live concert context.

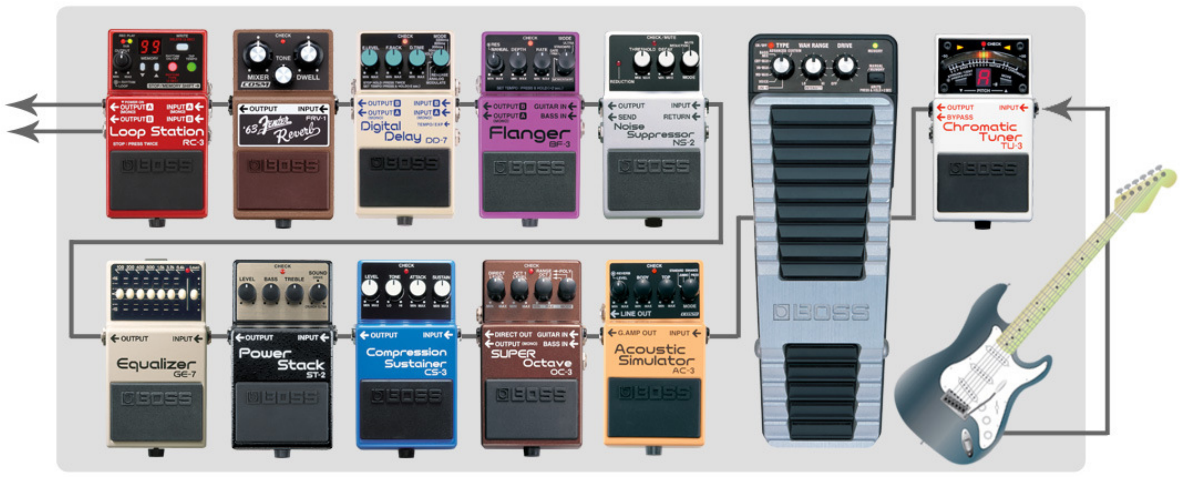


Figure 6. Signal path of a guitar effects pedal board

Guitar pedal effects boards such as the one depicted in Figure 6 have evolved out of many guitarists' tendency to use different tones from individual effects pedals at different times throughout their sets, be that from song to song or even from section to section of a song. Given the popularity of the guitar, effects pedal boards are a very common example of effects processing as an augmentative instrument. The pedal board has seamlessly incorporated itself into the contemporary guitarist's performance technique. Activation and de-activation of individual processors is binary, performed with the musician's feet. Parametric modification is awkward, but possible, requiring the performer to come to their knees to turn the tiny knobs of each pedal. Therefore, more often than not, each processor is simply treated as a tone modifier on a panel of sonic switches.

The cumulative result of turning different combinations of effects pedals on and off can be used as an avenue of expression; as an array of static *sound templates*. But, on the fly re-ordering of the way that these pedals are connected in serial - which may or may not have dramatic results, depending on the pedal - is not possible within an uninterrupted performance. This, in addition to the aforementioned awkwardness of turning knobs on the floor as an avenue of expression. Signal flow is a permanent line, flowing from the first connected effect to the last. Parallel connections are possible, but routing becomes even more complicated to physically manage in this case.

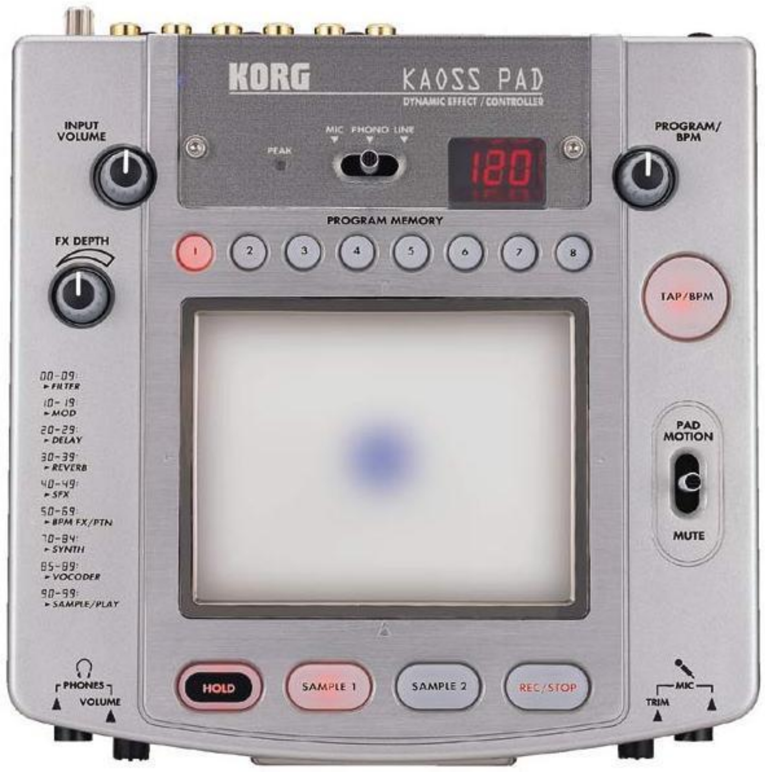


Figure 7. KORG ***KAOSS Pad KP2*** **(2002)**

In the late 1990s, synthesizer company KORG launched a product line called *KAOSS* that appears to have been designed to address many of the above-identified issues of the guitar pedal as a performative effects processor. Originally marketed to DJs, the *KAOSS* line (such as the model shown in Figure 7) became popular with many audiences over the first decade of the 21st century, including gear-happy experimental guitarists like Jonny Greenwood of Radiohead (McNamee), and performing laptop musicians like breakcore artist The Flashbulb.

A *KAOSS Pad* is a bank of digital effects mapped two-dimensionally as a finger-driven, guitar pedal-esque touchscreen. A typical *KAOSS* model includes sequenced-arpeggiator synthesizers, buffer-based processors, sweepable filters, and waveshaping distortion effects. Versus the light switch-style guitar pedal, gestural control is now three-dimensional: activation is managed by an attack-release envelope around the touch of the screen, and two effect-specific parameters are mapped to the X- and Y-dimensions of the interface.

These revisions are a significant improvement in the interest of embodied performability. This form factor has even gained some traction in commercial concert music. Most notably, Matt

Bellamy of the progressive alternative rock band Muse uses an electric guitar with the *KAOSS*-style design of a two dimensional touchscreen embedded into the body of the guitar to control various effects processing parameters in his guitar rig.

However, although one *KAOSS Pad* can swap the kind of effect that it is controlling, the issue of using one single guitar pedal versus a board of pedals is re-introduced in this context. One could chain several *KAOSS Pad*s together if they pleased, but, being so-called "digital multi-effects processors," they are quite expensive. Most of the *KAOSS* line is quite a bit larger than the standard guitar pedal footprint. Also, the *KAOSS Pad*'s cryptic three-character display would obscure which device is responsible for which chosen effect. One could automate changing which effect is mapped to which pad, but that would require an external solution of some sort.

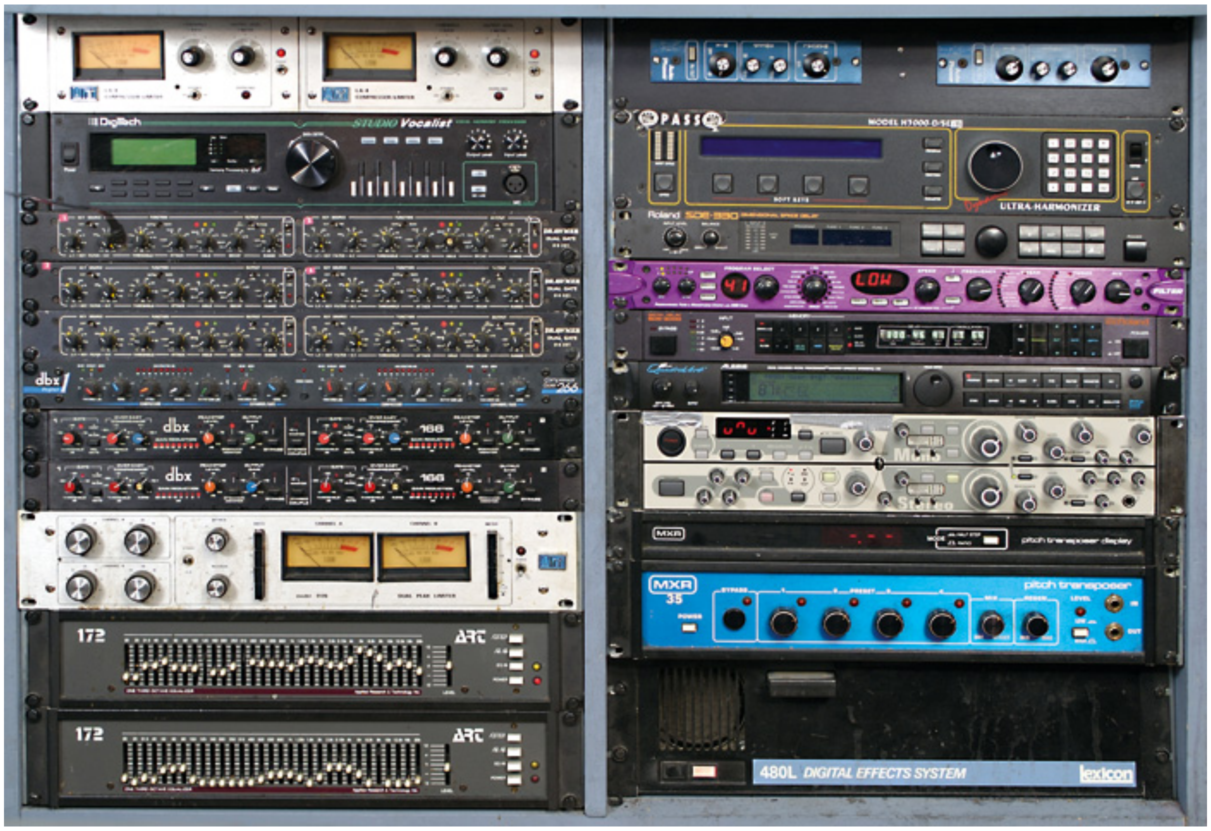


Figure 8. Two full racks of recording studio rack-mounted effects processors

Rack-mounted studio gear - like the two big racks shown in Figure 8 - offers probably the most explicit level of parametric control of all the presented examples, but their front-facing design and per-device interface complexity does not imply a communicative stage performance. Managing their activation and de-activation is complicated, usually requiring a routing device like a patch bay (with patch cables) or a mixer. In a hypothetical stage situation, an audience faces the back of a rack, seeing maybe only the connections to a routing hub. The performer themselves is hard to see, let alone the gestures of their performance.

The mechanic of processor actuation is specific to the device: some have *active/bypass* switches; some have *wet/dry* controls; some, like tape delay units, cannot be cleanly actuated; they must first output the audio that was last run through the unit. Moreover, most of these units are meant to be *augmentative* without being *performative*. For example, while a lot of music has used the typically-rack mounted compressor/limiter/expander or the equalizer creatively as part of the composition, their control sets do not obviously correlate in mapping to a performance gesture.

The use of rack mounted recording studio processing equipment is typically not performed at the "note level" (Wanderley and Orio) - which can be considered an additional parameter for classifying a traditional instrument. "Note level" is defined as one of the three strata that electronic instruments are capable of operating at. Whereas traditional instruments can only perform at the note level, electronic instruments, for example, can act at the "score level", as conductors of form, launching new sections of a piece.

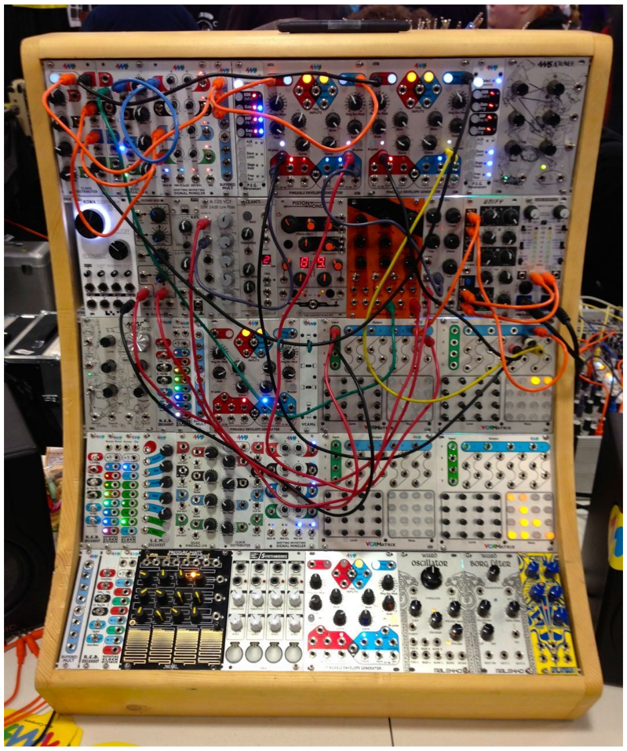


Figure 9. A typical Eurorack modular synthesizer setup

Modular synthesizers (like the one shown in Figure 9) present the actuation dilemma of rack-mounted gear in another way. We may see the act of connecting and disconnecting modules, or the input gestures placed into the system by a peripheral or a step sequencer. However, the routing responsible for the musical result - and the implication of which devices are involved in creating that result - is opaque, and often very complicated. The necessary complexity of building a functional patch becomes the mechanic of performance, binding the performer's ability to respond to moments: quick changes to the music in energy and motivation, stemming from the response by the musician him/herself and by the audience.

Modular synthesizer performances tend to be *evolutionary* in nature; it is not very easy to completely undo one specific contributing aspect of a moment and modify it in a way that easily allows for new, different moments quickly. Unless they are being driven by a traditional keyboard peripheral to serve as a "*synth sound*", modular synthesizers tend to be featured in sets, rather than as ensemble members.

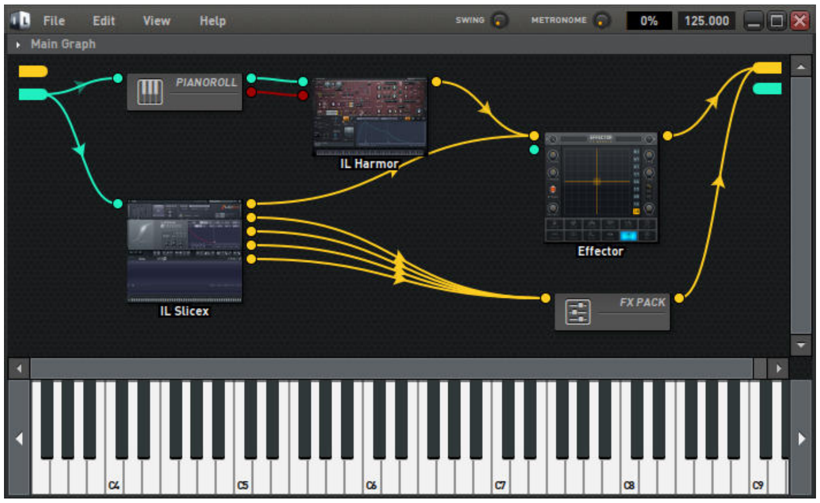


Figure 10. ***Minihost Modular*** **by Image-Line (2014)**

Some dynamic solutions have been implemented as software, as well. Modular plug-in hosts - such as Image-Line's *Minihost Modular* (Figure 10), and various standalone VST/AU host applications - objectify plug-in instruments and processors as though they were the modules of a hardware modular synthesizer. Audio output is patched from each module, and control signals are routed from one master keyboard and set of controls.

Although faster routing flexibility is possible, the issues associated with a communicative hardware modular synthesis performance are compounded with the issues associated with working with hidden, virtual devices on the computer, from the audience's perspective.

Furthermore, the workflow paradigm of these kinds of applications will be excessively beholden to the metaphors of working within the MIDI controller-centric paradigm of the DAW plug-in, since DAW plug-ins are the objects within this environment. Control messaging of individual devices is dependent upon a single MIDI keyboard; the input range can be divided, but ultimately, the environment still functions as a large, single-sound keyboard instrument.

1.5 Summary

In this chapter, it has been identified that effects processing has attained a compositional importance in popular music. This "promotion" can be pinpointed to the switch to computer-based audio recording, which greatly accelerated its use in music making. However, as the use of aggressively computerized effects processing has become an ordinary technique in popular music production, many of the most important elements of a pop song have their origins in being a sequenced cue that was not created performatively. Therefore, important pieces of the music are lost in concert due to their identity being impinged upon a non-instrumental effect achieved with a computer.

If we were to re-consider the effects processor as a new type of instrument as a means to possibly resolve this disconnect, there are several parameters that need to be adjusted in order to make the effects processor more instrument-like. Most effects processors have no *dynamic actuation mechanism* critical to performing at the "note level" (Wanderley and Orio). Also, the definition must be framed to include a*n environment of a diverse series of effects processors* rather than a single effects processor by itself, given the multi-component and often very involved combinations of effects processors used to make one *post-processing sound.*

Most hardware manifestations of this idea refer only to single processors, which have awkward actuation mechanics, due to their typically non-real-time use. Devices such as KORG's *KAOSS Pad* have a very intuitive touchscreen-based actuation mechanic, as well as the ability to represent many different effects on the same field, although not at the same time. However, the idea of an environment representing the independent control of many different processors simultaneously still remains.

It becomes apparent that this instrument concept is best suited to existing entirely digitally within a software environment that is primed with event listeners in order to adapt some kind of dynamic chain.

The precedent of digital modularity in electronic music synthesis stretches as far back as the beginning of electronic music synthesis itself. Encapsulated digital functionality was first manifest in the form of the Unit Generator, conceived by Max Mathews. The Unit Generator influenced both hardware synthesis and programming language platforms alike.

The next chapter examines the history of ideas following the original inspiration of the Unit Generator, and how it has manifest in visual programming languages, software plug-ins and newer platforms since then.

Chapter 2

History: The Unit Generator and Software Instruments

"*If I've done anything, I am an inventor of new instruments, and almost all the instruments I have invented are computer programs. [...] So if I am remembered for anything, I would like to be remembered as one of the inventors of new instruments. I think that this occurred not because I did anything special, but because I was in at the beginning when computers first became powerful enough to deal with sound and music. I guess I didn't have any way of predicting the popularity of digital music, and that popularity was the result of many people other than me making it easy to do and far less expensive. And I certainly had no idea of the rapid advance in computer power that we have seen in the last 50 years, and I don't think anyone else did either, or at least very few people [did]. I didn't realize how popular digital music would become, especially for popular music...*"

- Max Mathews, March 2009 (Park 22)

Figure 11. Max Mathews with *Radio Baton*

This chapter contains a historical overview of the history of modular functionality in music software. A richer understanding of how the attempt to create an instrument out of many smaller modular sub-processes is gained over the course of this explanation.

We begin where nearly every topic begins in computer music - at a man named Max Mathews, who dedicated his whole life to executing on his vision of "the computer as an instrument." What we are mainly concerned with is his establishment of the idea of a the *Unit Generator* - a macro-sized set of instructions to a computer processor that is of enough utility to be re-usable in multiple contexts.

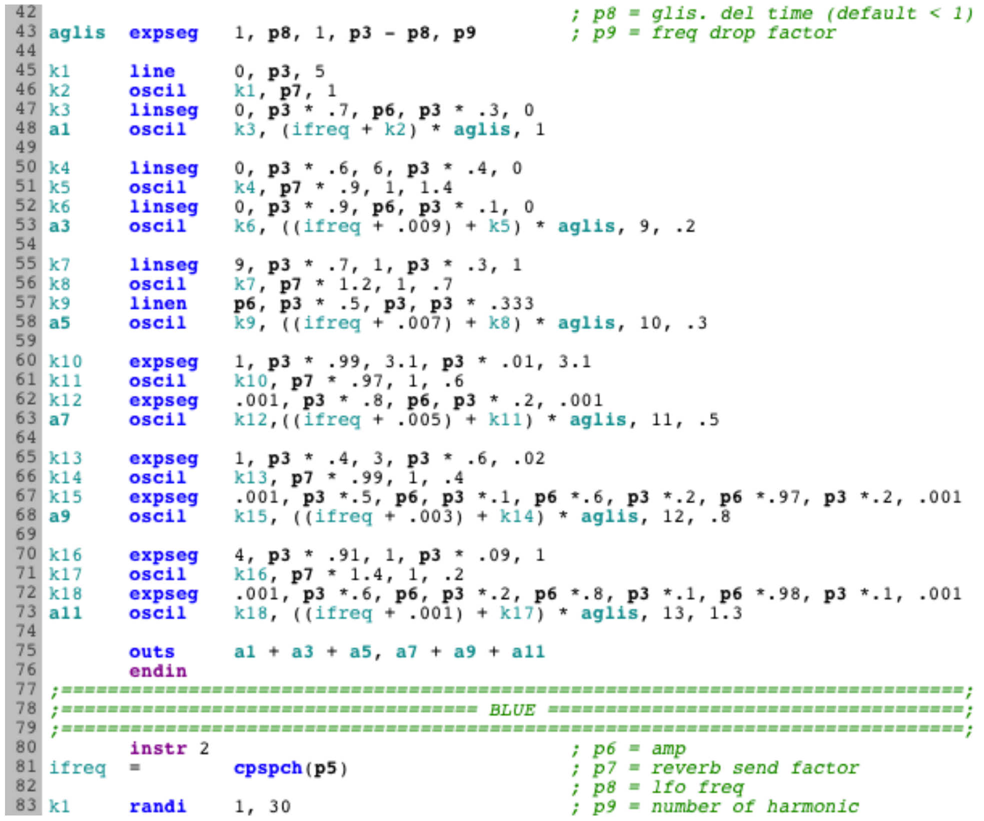
After seeing how the *Unit Generator* gave birth to the idea of the *object* within the visual programming environment, we begin to see how projects made of visual objects can themselves ultimately be encapsulated as a complete definition of an instrument. This new level of objectivity naturally cross-pollinated with the workflow of the DAW to culminate into the DAW plug-in. We see how the plug-in then in turn influenced the external control structure of the modular audio software object.

We then observe the way that the change that using plug-ins in music production began to change the interfaces that were being used to make music. The quest for re-embodied computer music resulted in the creation of an entirely new academic conference. We can then reflect on the complexity of modern popular electronic music production equipment setups, and the way in which a break from the "custom template of mapped controls onto a generic device" can begin to break away the complexity of digital modularity in music. By encapsulating many of these mechanisms so that they are hidden from the user, we can begin to focus on practicing the instrument, rather than figuring out how to build it each and every time.

2.1 Max Mathews, *Music-N*, and cross-platform functionality

Max Vernon Mathews is commonly described as "the father of computer music." (Chowning) Though *computer music* encompasses a tremendously diverse array of research interests, the luminary Mathews began and then continually contributed to many facets of it for over fifty years. Max was among the first to digitally render audio waveforms and then acoustically transduce them in 1957 (Di Nunzio). His trailblazing of the computer as an instrument via his *Radio Baton* in the 1970s (pictured in Figure 11) defined the beginnings of a parametric controller model for digital music performance interfaces (Boulanger et al.). His work on phasor filters (Park) and scanned synthesis (Verplank, Mathews, and Shaw) in the 1990s and 2000s showed a man driven by overall passion and vision far into his twilight years. The groundwork laid by Max Mathews in so many fields of digital music synthesis is fundamental to the work of so many.

The family of audio programming languages that descend from Mathews' original language *MUSIC*, which are commonly referred to as *Music-N*, are crucially organized around the idea of the *unit generator*, introduced in *MUSIC III* from 1960 (Roads and Mathews 6). Originating before the use of cross-platform language compilation via platforms like *FORTRAN* (Roads and Mathews 7) and *C*, the *unit generators* of *MUSIC III* - within this domain of digital audio synthesis - predates modern forms of object-oriented programming that frame similar relationships between prototyped data structures[1].



```
                                                  ; p8 = glis. del time (default < 1)
aglis   expseg    1, p8, 1, p3 - p8, p9           ; p9 = freq drop factor

k1      line      0, p3, 5
k2      oscil     k1, p7, 1
k3      linseg    0, p3 * .7, p6, p3 * .3, 0
a1      oscil     k3, (ifreq + k2) * aglis, 1

k4      linseg    0, p3 * .6, 6, p3 * .4, 0
k5      oscil     k4, p7 * .9, 1, 1.4
k6      linseg    0, p3 * .9, p6, p3 * .1, 0
a3      oscil     k6, ((ifreq + .009) + k5) * aglis, 9, .2

k7      linseg    9, p3 * .7, 1, p3 * .3, 1
k8      oscil     k7, p7 * 1.2, 1, .7
k9      linen     p6, p3 * .5, p3, p3 * .333
a5      oscil     k9, ((ifreq + .007) + k8) * aglis, 10, .3

k10     expseg    1, p3 * .99, 3.1, p3 * .01, 3.1
k11     oscil     k10, p7 * .97, 1, .6
k12     expseg    .001, p3 * .8, p6, p3 * .2, .001
a7      oscil     k12,((ifreq + .005) + k11) * aglis, 11, .5

k13     expseg    1, p3 * .4, 3, p3 * .6, .02
k14     oscil     k13, p7 * .99, 1, .4
k15     expseg    .001, p3 *.5, p6, p3 *.1, p6 *.6, p3 *.2, p6 *.97, p3 *.2, .001
a9      oscil     k15, ((ifreq + .003) + k14) * aglis, 12, .8

k16     expseg    4, p3 * .91, 1, p3 * .09, 1
k17     oscil     k16, p7 * 1.4, 1, .2
k18     expseg    .001, p3 *.6, p6, p3 *.2, p6 *.8, p3 *.1, p6 *.98, p3 *.1, .001
a11     oscil     k18, ((ifreq + .001) + k17) * aglis, 13, 1.3

        outs      a1 + a3 + a5, a7 + a9 + a11
        endin
;=========================================================================;
;================================== BLUE =================================;
;=========================================================================;
        instr 2                                   ; p6 = amp
ifreq   =         cpspch(p5)                      ; p7 = reverb send factor
                                                  ; p8 = lfo freq
k1      randi     1, 30                           ; p9 = number of harmonic
```

Figure 12. Excerpt of "Trapped in Convert" by R. Boulanger, written for *Music 11* (1979)

[1] For more information on the evolution of *Music-N*'s lineage, please see http://www.musicainformatica.org/topics/music-n.php as reference.

These macro blocks of pre-compiled functionality, interfacing in a graphable data flow relationship, even influenced the component-driven analog hardware modular synthesis systems of Moog (Park 20), Buchla and Serge.

The definition of an abstract "instrument" in an "orchestra" which is further comprised of abstracted unit generators variously comprised of line segments executed over time, oscillators, lookup tables, and amplifiers carries through to modern iterations of *Music-N*, as in *Csound*, pictured in Figure 12. The featured piece was written in 1979 for *Music-N*'s last platform specific manifestation, *Music 11* for the PDP-11. It can still run as intended on a 2014 computer, thanks to the revolutionary ideas of Mathews' cross-platform support.

Before Mathews' pioneering work in cross-platform audio synthesis environments, many routine building blocks of doing audio development work - such as the use of a basic sine wave oscillator, or the use of an amplifier which sums signals - would have to be implemented at the lowest level above binary, in that computer's processor's specific assembly language. This means that there can be no shortcuts in prototyping any system; every time the basic building blocks of modular synthesis are required, it must be defined for that processor's architecture.

The breakthrough of *Music IV* was in making the *MUSIC* environment cross-platform through the *FORTRAN* language (Roads and Mathews). This set an important precedent for all synthesis languages following it, though the common base language is now C, or an extension such as C++ or Objective-C. Cross platform construction platforms now describe almost every construction platform, so that units can be shared between machines and programmers working on an idea, the implementation of which will be interpreted by the new machine.

Constructing an audio engine on a computer platform that enjoys widespread cross platform support is key for constructing an instrument that can run outside of specialized hardware profiles. The environment must deal in modules that are "high-level" enough to achieve robust advancements towards an instrument's intended functionality, but "low-level" enough that they are general enough to design a power and flexible system with.

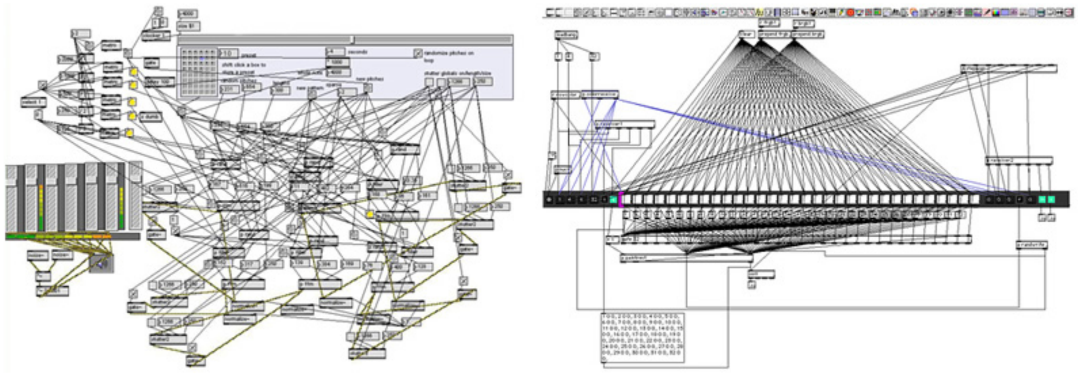


Figure 13. Typical *Max* patches

2.2 *Patches* in visual programming

Mathews' work on audio synthesis languages was highly shaped around his application context: as a trained violinist, he hoped the computer would be able to mutually recite a composition with him, serving as accompaniment and improvisation partner. Thus, the execution logic of a *Music-N* program struggles in the realtime domain by its use of *orchestra* and *score* terminology to describe runtime durations. Miller Puckette, who studied with Mathews in the 1970s and 1980s, made a crucial adaptation in removing this terminology in order to focus on real-time event listening. In the process of adaptation, Puckette made the crucial step of transforming the workflow of the unit generator into the mouse-driven *visual programming* domain.

In a process that took over a decade, Puckette reframed his own iteration of the *Music-N* environment called *Music500*, paired with a GUI[2] called *M* (Puckette, *Interprocess Communication and Timing in Real-Time Computer Music Performance*). *Music500* was developed to incorporate into *Music-N* Mathews' *RTSKED* framework, which handled multiple asynchronous control data schemes simultaneously into a realtime performance and synthesis context (Mathews). Consequently, Puckette's development efforts restructured *Music N*'s operative terminology away from the through-composed ideals of *orchestra* and *score* and objectified these control data messages.

[2] Graphical user interface.

Supplanted by the industry revolution of MIDI[3] and bolstered by the commoditization of the personal computer, *M* would eventually morph into a window-based data type called a *Patcher* (Puckette, “The Patcher”). *The Patcher* grew to engulf the design of Puckette's environment, becoming the core of the computer program *Max* by the mid-1980s. By the 1990s, the audio DSP-powered *Max/MSP* and its open source variant *Pure Data* began a lineage of "patch"-driven programming languages by themselves (Puckette, “Max at Seventeen” 33). Patches tend to look very complicated, as seen in Figure 13.

The framing of the *unit generator* into a chart of literally connected objects forming event-driven signal flow in the versions of *Max* of the early 1980s presents one of the earliest so-called *visual programming environments*. Visual programming environments are essentially the byproduct of the mouse-driven, window-based personal computer operating system from Xerox PARC etc. from the late 1970s to the mid 1980s (Nickerson).

The capacity for abstraction, encapsulation and duplication presented by this workflow adds a facet to software-based, digital modular synthesis that distinguishes it from hardware-based signal processing techniques. This primarily mouse-driven programming technique has a tradeoff in lateral bloat of programming inefficiencies, but it tremendously simplifies the overhead of getting a sophisticated data model/controller program running as a functional prototype (Puckette, “Max at Seventeen” 35–37).

2.2.1 The *Max* patch, lingua franca of computer music

After nearly thirty years of the continued popularity of the *Max* workflow within the computer music community, the *Max patch* is considered by some to be the "lingua franca" of the culture (Puckette, “Max at Seventeen”).

Thanks to features like *Max Runtime* and *Presentation Mode*, *Max* has grown from the domain of rig-specific peripheral-driven multimedia installations and concert performances. It has been used as an intercalary development platform for prototypes of proprietary standalone software, such as Ableton's DAW *Live*[4] and some of its audio processing plug-ins[5]. It has shipped with

[3] Musical instrument digital interface.
[4] http://www.musicradar.com/us/tuition/tech/a-brief-history-of-ableton-live-357837

mainstream hardware products as a firmware re-flasher and a controller re-mapping environment[6]. With some extra preparations, *Max* patches can even be sold as standalone applications through Apple's *Mac App Store*[7].

From the development of new, asynchronous communication protocols such as *OSC* (Open Sound Control) (Wessel and Wright), to the core mechanics of music software UX[8], to the concept of interdependent audio software modularity itself, *Max* and its ilk have had a profound influence on the modern music technology software landscape.

2.3 Software plug-ins

Thanks to the unfolding benefits of Moore's Law (Moore 114–117) and innovations such as the *USB*[9] standard of 1997, personal computer-based digital audio recording and editing began to enter the mainstream around the late 1990s to the early 2000s. Offered at a completely different price point with new use cases, music production equipment became just as much of a commodity as the personal computer had in the previous decade.

At this point in time, the commercial music production industry had a big incentive to develop modularized signal processing functionality, because such products have almost no distribution overhead or manufacturing cost.

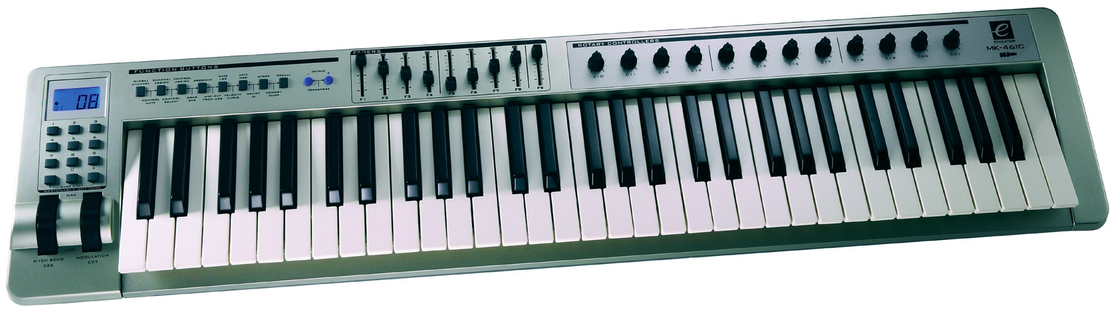


Figure 14. The Evolution *461C*, what most controllers looked like before Ableton *Live*

[5] http://www.kvraudio.com/interviews/gerhard-behles---pushing-ableton-into-new-territory-21702
[6] http://www.keithmcmillen.com/softstep/software
[7] http://www.cycling74.com/2012/04/19/get-your-max-standalone-on-apple%E2%80%99s-mac-app-store
[8] User experience.
[9] Universal Serial Bus.

The widespread adoption of Steinberg's SDK (Software Development Kit) for the VST 2.4 standard[10] had an important role in creating the culture of the *bedroom producer* (Gunderson), which created entire new industries revolving around the MIDI controller and the *software plug-in* for DAWs. MIDI controllers began to change their form, in order to better represent the software they were controlling: the decades-held representation of the piano keyboard offered by companies like Midiman, Edirol and Evolution (Figure 14) began to shift toward non-representative *grid controllers* inspired by the Akai *MPC* line, such as the *Monome* (Figure 15), the Novation *Launchpad*, the Akai *APC40* and the Ableton *Push*.

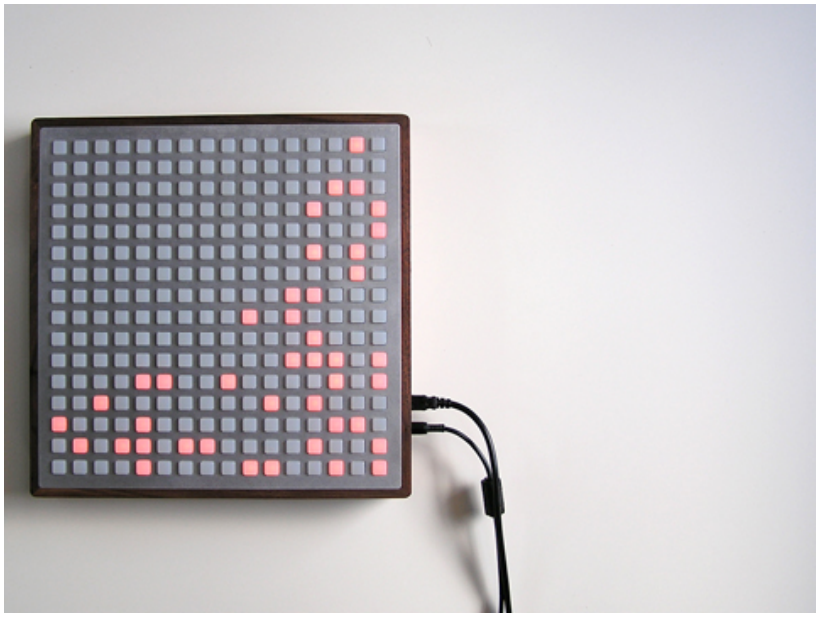


Figure 15. The *Monome* (2006) - the influential minimalist software controller

2.3.1 *Reaktor*

One of the most popular and enduring examples hybridizing the software embodiment of hardware analog modular synthesis and a *Max*-esque visual programming environment for managing control data in an event-driven fashion is a program called *Reaktor*[11] by Native Instruments.

Launched as *Generator* in 1996 by Stephan Schmitt[12], *Reaktor* originally shipped with its own hardware sound card for DSP calculations[13]. This software pre-dated the conception of the

[10] http://www.steinberg.net/en/company/developer.html
[11] http://www.native-instruments.com/en/products/komplete/synths-samplers/reaktor-5
[12] http://www.kvraudio.com/interview_with_native_instruments.php

cross-platform plug-in software instrument introduced a year later by Steinberg with their *VST* standard[14]. Native Instruments quickly conformed *Reaktor* to the nascent plug-in standard.

By having the ability to run *Reaktor* as a software plug-in within a DAW host, this environment functionally links an old conception of software digital audio synthesis to the emergence of a new one, based around single-channel closed systems that are constructed of black box processes arranged in a linear fashion.

Native Instruments have even embraced their own form of using their software as a development platform, much in the same way many develop software sold to end-users entirely within Max/MSP. Native Instruments products such as *Spark*[15], *Razor*[16], and *Prism*[17] are built on top of *Reaktor*, exemplifying how an open system can create new closed-system form factors again.

By hybridizing the hardware modular synthesizer, the software visual programming environment, and the DAW plug-in together as one software product, *Reaktor* occupies a unique niche in the software development platform market. Able to piggyback onto the transport controls of a host environment, able to open up on multiple operating systems, and with the ability for *Reaktor* projects to be encapsulated as standalone devices, *Reaktor* offers an exciting sandbox for development.

---

[13] http://www.soundonsound.com/sos/sep98/articles/generator.html
[14] http://www.musicradar.com/us/news/tech/a-brief-history-of-computer-music-177299
[15] http://www.native-instruments.com/en/products/komplete/synths-samplers/reaktor-spark
[16] http://www.native-instruments.com/en/products/komplete/synths-samplers/razor
[17] http://www.native-instruments.com/en/products/komplete/synths-samplers/reaktor-prism

2.4 Diverse, dynamic interfaces for virtual instruments

The development of new software instruments in this new patch-driven climate is not beholden to any standardized control scheme that could be modeled off of the typical keyboard instrument. This has caused the hardware interfaces that control these instruments to evolve just as quickly as the development of new software instruments has. Substantial efforts have been made throughout the past twenty years both in the academic and commercial realms, at times cross-pollinating.

2.4.1 Academic computer music conferences

The desire to re-embody the mechanics of virtualized synthesis is an obvious byproduct of the software environment. This is a topic that has been approached not only in the commercial product sphere, but also within the academic world. The ability to re-distribute encapsulated synthesis environments as a single computer file gave a much greater sense of ontology to software instruments. Artists shared *Csound orchestras* and *scores*, *Max patches* and *Reaktor Ensembles* with each other. Therefore, how can digital music artists fabricate interfaces that allow gestures to have throughput into an environment that would allow them to be expressive?

One such academic conference that has attacked this question head-on is New Interfaces of Musical Expression, or NIME. Launching in 2001, the NIME conference invites papers from fields as diverse as computer vision (CV), human-computer interaction (HCI), machine learning and artificial intelligence, fabrication, and mechanical engineering. NIME authors continually publish works such as the calibrated response of certain sensors (Freed), description of an artist's visual programming environment that they use in their performance practice (Kleinsasser), and philosophical papers about the nature of physical - yet virtual - musical instruments (Cook).

NIME's body adds to conferences such as the International Computer Music Conference (launched in the 1970s) and the Sound and Music Computing Conference (launched in the 2000s), which often provide a prophetic glance into new movements in controller design - such as work on mobile phones (Gaye et al.) and the Wacom tablet (Zbyszynski et al.) that pre-date the smartphone revolution.

2.4.1.1 *Digital luthierie: the reacTable*

A device called *reacTable* (Jorda et al.; Kaltenbrunner et al.; Jordà et al.) has managed to successfully cross domains from the academic realm to the popular recognition of the mainstream. reacTable creator Sergi Jordá cherry-picked ideas from HCI, computer vision, embedded systems, modular synthesis, visual programming, and the very nascent beginnings of gestural multi-touch vocabulary. *reacTable*, which is pictured in Figure 16, robustly espoused an unprecedented new form factor which pre-dated the popularity of mainstream touchscreen music applications found on *iOS* and *Android*.

Jordá coined the term "digital lutherie" to refer to the responsibility Digital Musical Instrument (DMI) designers have to re-embody the software environments they create into formats that can be injected with observable expression, akin to the way a luthier carves a guitar to refine its tone and character. (Jorda)

reacTable even saw on-stage use by many popular music artists, such as singer Björk, and alternative rock band Coldplay. Given that both of these artists make heavy use of offline electronic music production techniques in the studio and feature prominent synthesizer work in their music, both prominently featured reacTable in their sets as a means to bring appreciation and performance back into this aspect of their music, outside of the traditional keyboard instrument interface.

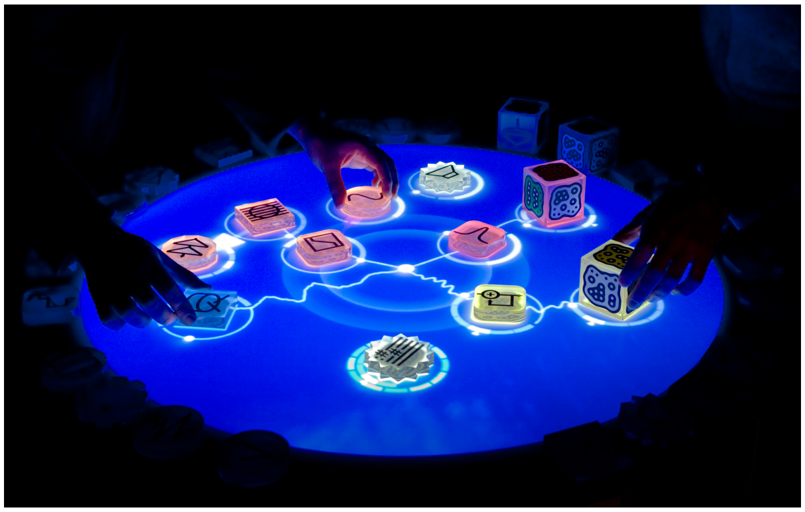


Figure 16. *reacTable*, mid-performance

Incorporating sample playback, DAW-inspired global controls for tuning, and a radius and proximity-based metaphor for output structure and signal flow, the *reacTable* spawned many new applications (Jordà and Alonso; Roma and Xambó) and iterative variations in the academic scene (Hochenbaum and Vallis; Partridge, Irani, and Fitzell; Crevoisier et al.) after its arrival for the remaining decade, and laid the foundation for many forthcoming *iOS* synthesizers.

2.4.2 The commercial product sphere

As plug-in instruments slowly managed to become the predominant source of content in the contemporary electronic music studio by the mid-2000s, MIDI controller manufacturers began to develop devices that were nothing more than blank-canvas packages of knobs and faders, such as the one in Figure 17, expressly for custom mappings that likely involved several software devices, often mapped across several pages.

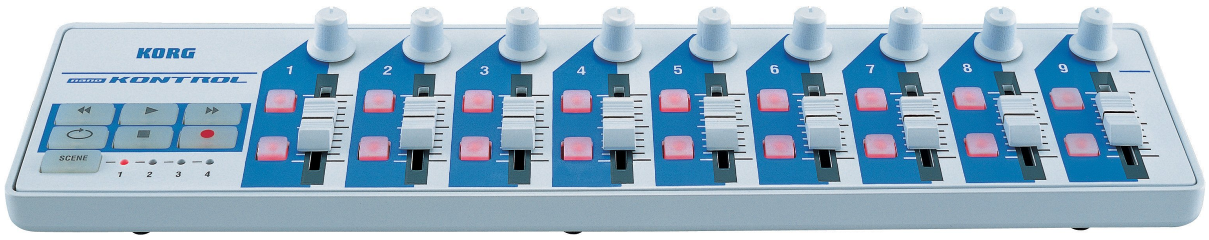


Figure 17. KORG's first edition *NanoKONTROL* (2008)

The burgeoning industry and culture surrounding these ideals became known as "*Controllerism*" (Attias and van Veen). Entire companies such as Keith McMillen[18] began selling products that spawned a cottage industry of selling *mapping templates* meant to optimize a user's workflow[19] when using a controller to drive a DAW.

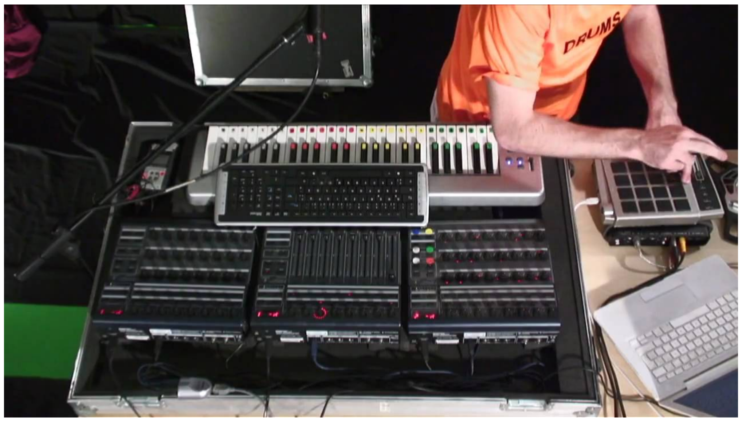


Figure 18. The controller sprawl of Tim Exile's live setup (2009)

[18] http://www.keithmcmillen.com/products
[19] https://www.youtube.com/watch?v=8TO3z8LtP5U

2.4.2.1 *Interface polymorphism: the Lemur Input Device and its descendents*

After a certain point of creating custom mapping templates for device-specific workflows around highly non-standardized software instruments, the use of a physical controller begins to fall apart in practicality. This is exemplified in Figure 18. It is apparent that at a certain point, it may perhaps be more useful to represent software instrument interfaces dynamically via a library of interface widgets that can adapt to the user's specific setup, via a touchscreen.

The *Lemur Input Device* by JazzMutant (seen in Figure 19) became a very early champion of the touchscreen controller, sporting a high-resolution display and floating-point control precision. Using desktop computer software, the user designs their own layouts with a widget toolkit that they could then upload to the unit. A rare out-of-the-box supporter of Open Sound Control (Wessel and Wright), the Lemur was publicly used on-stage by many popular artists such as Nine Inch Nails, Björk, Daft Punk, Richard Devine, The Glitch Mob, Ritchie Hawtin, Justice, and Ryuichi Sakamoto[20].

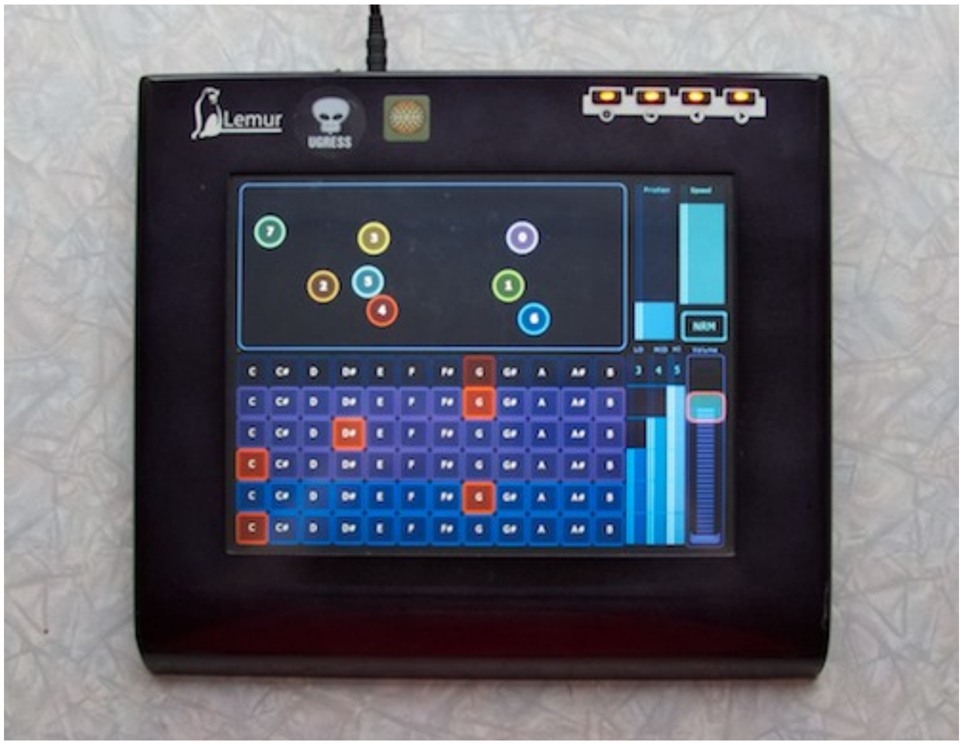


Figure 19. The original *Lemur Input Device* by JazzMutant (2002)

[20] http://www.jazzmutant.com/artists_lemurized.php

As live performances increasingly began to comprise of the realtime manipulation of software instruments, being beholden to physical controls could sometimes become unwieldy. This is why JazzMutant championed *Lemur*'s optimized connectivity with custom instrument platforms such as *Reaktor* and *Max/MSP*. Despite a lack of haptic feedback, user-designable digital touchscreen interfaces fared well as a practical counterpart to extremely diverse user-designed software-only musical instrument rigs.

In the *Lemur* environment, the interface designer has access to global variables, conditional scripting, OSC input and output, container objects, and procedural, iterative commands. This capacity for *scripting* distinguishes it as a highly capable platform for representing system-wide interface changes like those necessary for interfacing with fully functional software on the laptop it is communicating with. One could effectively create *the* interface for the complex modular software-based ecosystem, without the performer ever having to reference the host computer.

2.5 Smartphones, tablets and the *app*

The *Lemur* was just barely ahead of its time. We have observed the ways in which hardware interfaces themselves have become more software-oriented, customizable and virtualized, in order to meet the evolving input requirements of software instruments. However, the arrival of a completely new type of computer as the 2000s shifted to the 2010s consolidated many standing efforts into a new form factor.

The 2007 iPhone and its operating system ushered in a standardization of the touchscreen device as a haptic, multitouch, finger-operated personal pocket computer. Though this effectively put companies like JazzMutant out of business[21], the *Lemur* concept endured through this new era, through porting and adaptation. The popularity of programs like Hexler's *TouchOSC* (seen in Figure 20). Eventually, the original JazzMutant *Lemur* environment would be sold to a new company called Liine, who ported it to *iOS* for use with iPhones and iPads.

[21] http://www.jazzmutant.com/press_release.php

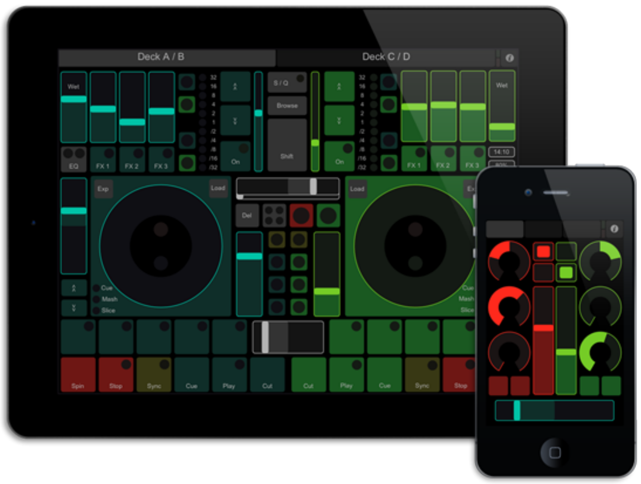


Figure 20. ***TouchOSC*, running on iPad and iPhone**

This model of using the touchscreen as a remote control is largely a holdover from the previous era of controllerism, however. Most of these instrument apps (including some of the ones in Figure 21) host the audio engine and the control interface as one self-contained program. Apps such as *Audulus*[22] and *Jasuto*[23] have adopted a post-*reacTable* (Kaltenbrunner, Geiger, and Jordà), proximity trigger-based, subtractive synthesis ethos; kind of a fiducially-driven Moog.

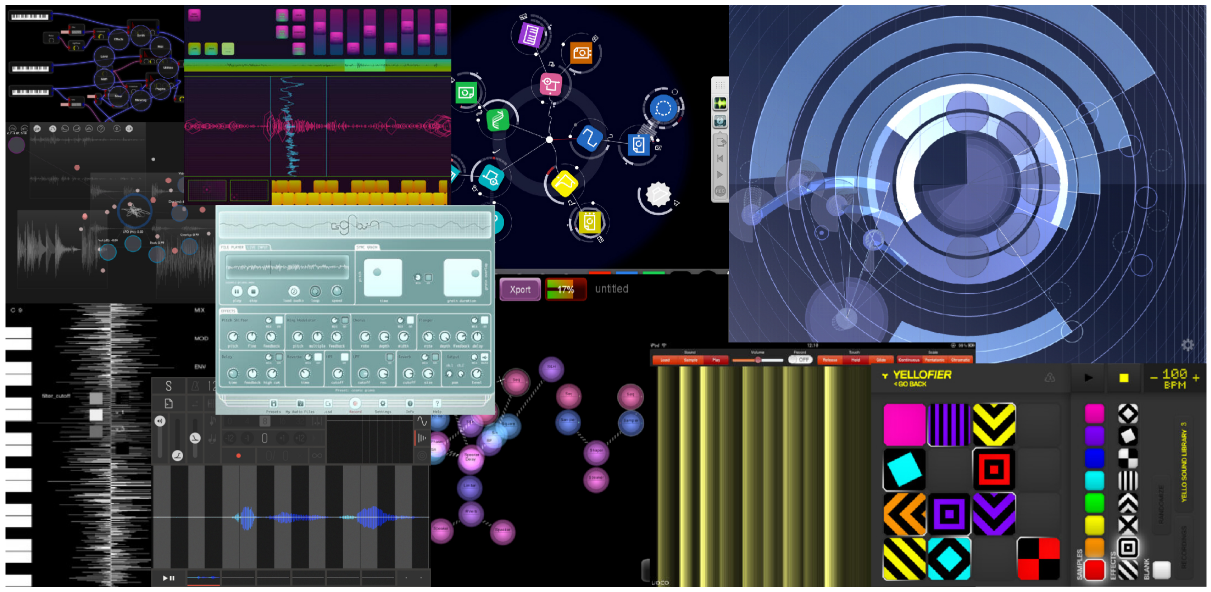


Figure 21. Lots and lots of performative iPad apps

Kevin Schlei's app *TC-11* (Schlei) is a unique merger of interface and localized synthesis. Schlei's program makes comprehensive use of the sensors of a tablet device - such as accelerometer and

[22] http://www.audulus.com
[23] http://www.jasuto.com/main

gyroscope - to a much greater extent, and is powered by a more traditional subtractive synthesis engine, somewhat akin in interface design and system architecture to Native Instruments' *Absynth*[24]. The gestural scalability of Chris Carlson's app *Borderlands Granular* (Carlson and Wang) is an artful example of the *painterly interface* (Levin). The general sense of performative composition through exploration was feels transcendent in the experience Carlson creates.

What's more, programs such as *AudioBus*[25] have managed to yet again re-encapsulate the runtime environment of the audio app so that it can be routed in an modular fashion through a dynamic serial interface. This is an example of the constantly shifting scope of modularity in electronic music devices; closed-operation platforms will be constructed from modular development sandboxes, only to be broken out again for cross-module interaction as a result of musician's workflows.

Interface-driven performative buffers are an extremely potent area of instrument design that encourages further manifestations. Through the *iOS* granular sampler applications of recent years - such as *Samplr*[26], *Yellofier*[27], *Curtis*[28], *MegaCurtis*[29], *Singing Fingers* (Rosenbaum and Silver) and *VOCO*[30] - back to *The Hands* interface of Michel Waisvisz (Waisvisz), live buffer-based performances can take many rich forms beyond just common live looping techniques. This feature alone can lead to satisfying concert presentations.

[24] http://www.native-instruments.com/en/products/komplete/synths-samplers/absynth-5
[25] http://www.audiob.us
[26] http://www.samplr.net
[27] http://www.yellofier.com
[28] http://www.thestrangeagency.com/ipad/#app-curtis
[29] http://www.thestrangeagency.com/iphone/#app-megacurtis
[30] http://www.jonathanmackenzie.net/apps/apps.html

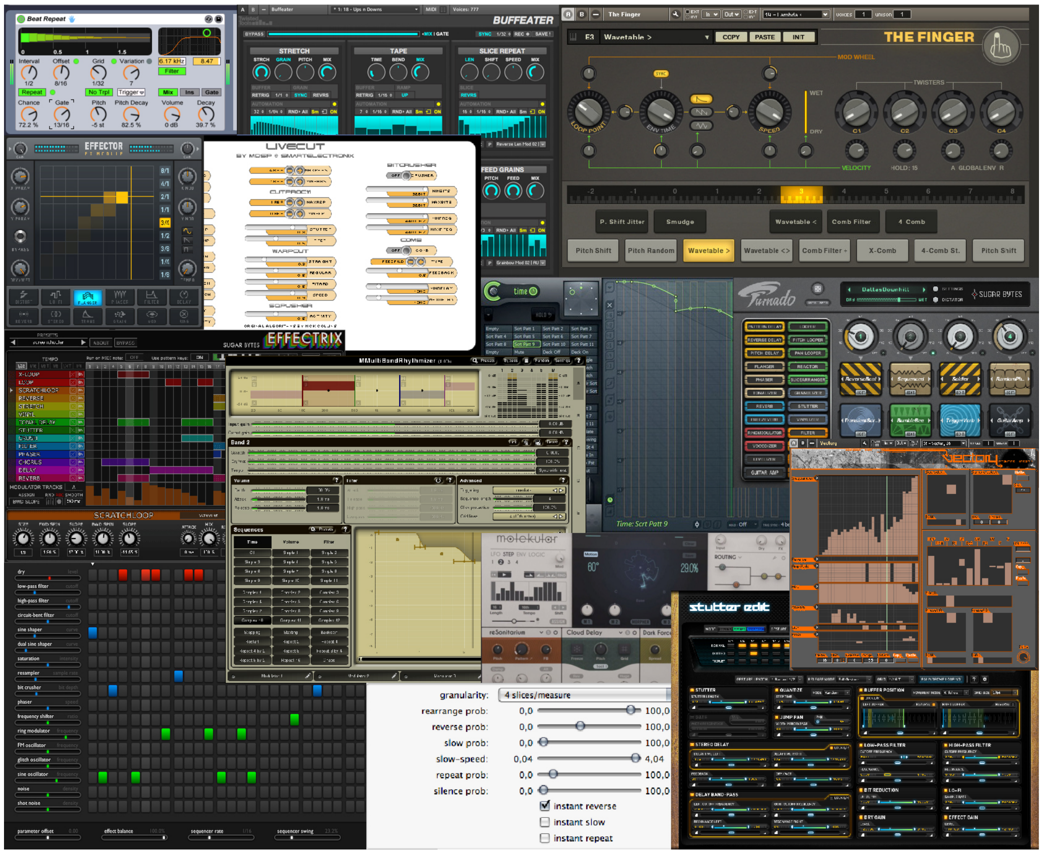


Figure 22. Lots and lots of multi-effect processor plug-ins

2.6 Dynamic multi-effect plug-in processors

The sea change of instrument virtualization has dramatically fragmented the look and feel of both digital instruments and the hardware that controls them. The multi-effects processor as a realtime performative plug-in instrument (as shown in Figure 22) seemed to be in the air as the 2000s changed to the 2010s, as hardware interfaces become as modular as the fragmented software ecosystems they are controlling. However, the strategy of implementing this complex idea as an automatable processor interface is subject to a wide variance in implementation.

Some plug-ins, like iZotope's *Stutter Edit* and Native Instruments' *The Finger*, allow the user to actuate cued sequences via mapped MIDI keyboard key presses. Others, like Sugar Bytes' *Turnado*, can execute looping phrases of effects processor automations that are activated varying amount by the twisting of eight consecutive knobs. Some, like *The Finger*[31], salamanderanagram's

[31] http://www.native-instruments.com/en/products/komplete/effects/the-finger

Optimus Prime[32], will dynamically add and remove elements in the order they are introduced, while others only have pre-defined serial paths with many objects available in the path.

Real-time signal processing environments such as these are novel to the new decade. However, none of these realizations seem to transcend the appearance of a performer behind a laptop manipulating a MIDI controller. Furthermore, their reliance upon cued automation structures give to the audience more of an impression of a conductor instructing a system, rather than a performer improvising an unmeditated gesture.

MeldaProduction's *MRhythmizer*[33] and Image-Line's *Gross Beat*[34] are direct competitors of the one-bar breakpoint automation multi-effect sequencer variety. Sugar Bytes' *Effectrix* and Sinevibes' *Sequential*[35] are similar in their one bar, 16th note step sequence serial multi-effect activation and de-activation mechanics. Image-Line's *Effector*[36] is like an exact software version of the KORG *KAOSS* Pad. Native Instruments' *Molekular*[37] - which is *Reaktor*-based, as Tim Exile's efforts such as *The Finger* and its precedents *Keymasher* and *Vectory* are - retains a four-slot ceiling, while letting you dictate between several routing options and allowing you to perform it with radius and angle-based knob controls. Twisted Tools' *Buffeater*[38] is another *Reaktor*-based performative multi-effects sequencing plug-in.

2.7 Summary

In Chapter 1, the growing utilization of effects processors in contemporary music was discussed, with virtualization through computerization being pinpointed as its leading accelerator. In Chapter 2, it is recognized that encapsulated, modular functionality has always been the most important design principle in computer music systems. Therefore, the digital software platform is perhaps a good place to consider thinking about an ecosystem of smaller objects as a musical instrument.

[32] http://salamanderanagram.wordpress.com/2012/02/22/optimus-prime
[33] http://www.meldaproduction.com/plugins/product.php?id=MRhythmizer
[34] http://www.image-line.com/plugins/Effects/Gross+Beat
[35] http://www.sinevibes.com/sequential
[36] http://www.image-line.com/plugins/Effects/Effector
[37] http://www.native-instruments.com/en/products/komplete/effects/molekular
[38] http://twistedtools.com/shop/reaktor/buffeater

The scope by which functionality becomes abstracted into a new modular sub-unit to be combined into an even larger environment has constantly been re-framed in software. The functional *Unit Generator* has gotten larger and more complex in definition, as computers are able to handle increasingly more intense operational blocks in tandem.

"Modularity" once referred to the *opcodes* of *Music-N* in the 1960s, where the ability to reference the same floating-point line segment executing over time was considered a re-usable unit. Then, "modularity" came to mean an entire project of functional blobs in the 1980s, where artists shared, re-combined and modified *patches* that were abstractions of the relationships between unit generators. *Patches* made of complex relationships between lower-level Unit Generators became a new kind of *Unit Generator*.

The culture of patches was transitional to the rise of *plug-ins* of the 1990s, when the commercial music production industry got involved. Commercial music production companies got involved once personal computers - not just major workstations - were powerful enough to host virtual instruments. Internal calculations are hidden from the user, who is presented with an often-skeumorphic graphic interface that is meant to represent an entire digital instrument or effects processor that has equal exchange to a piece of hardware. These *plug-ins* are compounded together in usage inside of the DAW. There, the new *sandbox* is the channel insert, using entire instruments made of abstracted blueprints of *Unit Generator* connections, which themselves are made of cross-platform, lower level *Unit Generators*.

Reaktor is a perfect representation of this mixed-scope sandbox, where a user can create a complete performance system - referred to in Native Instruments' terminology as an *Ensemble* - that presents entire virtual drum machines and sequencers alongside logic-level math operators; multimodal filters with complete interfaces live along side procedural iterators.

Reaktor and *Lemur*, when presented together as a new closed-unit solution, provide an opportunity to wrap the DAW functionality, digital modularity, and macro-level object manipulation into a computer-side software platform that could adequately model a dynamically re-routable effects processing chain, while being represented on the instrument as a context-sensitive, dynamically adjusting set of controls on a screen-based interface.

2.8 Introducing *Touchpoint*

With the release of products like iZotope's *Stutter Edit* and Native Instruments' *The Finger*, we see an industry that is headed toward the idea of a performative multi-effects processor. Plug-ins like these are meant to be considered on their own terms as an object responsible for effects processing that is featured prominently in real-time use. However, their connection to necessarily running from within a DAW - and their need to connected to a MIDI controller, inheriting all of their operation logic from note events - seems convoluted. It short-changes the opportunity for this concept to embody its own form factor, rather than confusing audience members further by assigning any miscellaneous post-processing task to a keyboardist.

It is from this starting point - the desire to somehow iterate on *The Finger* and *Stutter Edit* - that I began to concieve of *Touchpoint*, which attempts to unify the plug-in style of multi-effects processor with the painterly interfaces of mobile device-based synthesizer apps.

Chapter 3

Touchpoint

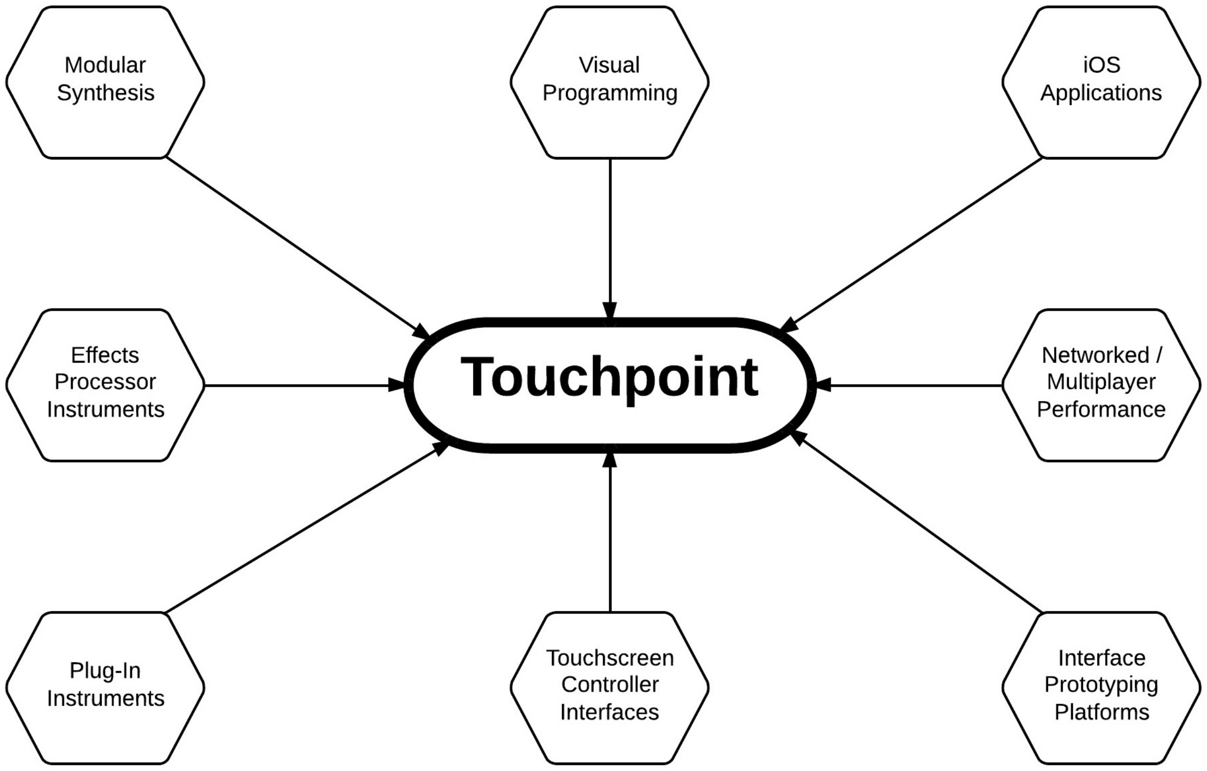

Modular Synthesis
Visual Programming
iOS Applications
Effects Processor Instruments
Touchpoint
Networked / Multiplayer Performance
Plug-In Instruments
Touchscreen Controller Interfaces
Interface Prototyping Platforms

Figure 23. ***Touchpoint*** **idea web**

Touchpoint is an instrument that was created to embody the act of assembling complex chains of signal processors on-the-fly, and manipulating their parameters gesturally. One "note" is correlated with the activation of a signal processor, which the performer can manipulate parametrically as though they were using vibrato on a string - in two dimensions. It weaves together ideas from many emergent fields of thinking about computer music, seen in Figure 23.

The control interface of the instrument is a multi-touch tablet; specifically an Apple iPad. This interface wirelessly drives an audio engine built in *Reaktor*. The main performance "pages" essentially work like a chain of KORG *KAOSS Pad*s that will dynamically apply an effects processor within the environment in the order pressed. By making the synthesis engine remote from the control interface, laptops can be removed from the stage altogether. By formatting communication wirelessly, *Touchpoint* can be used in single or multiplayer fashion. It is flexible to either scenario, or mixed configurations.

Whereas Chapters 1 and 2 attempted to set a stage for *why Touchpoint* was created, Chapter 3 explains the *what* and *how* behind its' construction. The structure of this chapter is such that four facets of the instrument are emphasized: the synthesis technique, the interface layout, and the networking technology. Each chapter will explain which platform was used to achieve this end and why.

3.1 Introduction

When work on *Touchpoint* began, its feature list was not specified to be anything outside of a DAW plug-in. Based on this perspective, comparisons between several similar plug-ins were made. In October of 2012, this feature comparison chart was rendered, to make sure *Touchpoint*'s feature set did not sufficiently match any of their profiles.

Table 1. Feature Comparison Chart of Similar Audio Plug-Ins, October 2012

	The Finger	Stutter Edit	Turnado[39]	Effectrix[40]	Beat Repeat[41]	SupaTrigga[42]	BBCut/LiveCut[43]
Active/Passive?	ACTIVE	ACTIVE	ACTIVE	TOGGLEABLE	TOGGLEABLE	PASSIVE	PASSIVE
Phrase locked?	YES	YES	NO	TOGGLEABLE	NO	YES	YES
Probability-based?	NO	NO	NO	NO	TOGGLEABLE	YES	YES
Sequenced?	NO	NO	NO	YES	NO	NO	NO
Enveloped?	YES	NO	YES	NO	NO	NO	NO
MIDI-controlled?	YES	YES	YES	TOGGLEABLE	YES	NO	NO
Serial FIFO?	YES	NO	YES	YES	NO	NO	NO
Polyphony	SIX	ONE	EIGHT	FOURTEEN	ONE	ONE	ONE

[39] http://www.sugar-bytes.com/content/products/Turnado/index.php?lang=en
[40] http://www.sugar-bytes.com/content/products/Effectrix/index.php?lang=en
[41] https://www.ableton.com/en/blog/guide-beat-repeat-quantize-courses
[42] http://bram.smartelectronix.com/plugins.php?id=6
[43] http://mdsp.smartelectronix.com/livecut

Touchpoint was created to satisfy a unique feature template, as per the characteristics listed in Table 1. Characteristics irrelevant to designing a DAW plug-in notwithstanding, this implies:

- Toggleable activation status; each processor can activate an input envelope or not
- Not phrase locked; effects last as long as the release envelope of the final remaining "note" is registered
- Not probability based; each effect is dependent upon the user's activation
- Not sequenced; axes are somewhat quantized, but effect duration and introduction does not resemble a duration setting or a step sequencer of some sort
- Enveloped per effect, tied to held-down notes/latched effects
- Not MIDI controlled, using a comprehensive, streamlined OSC specification
- Abiding by a serial FIFO ("first in, first out") stack algorithm for effects
- Hypothetically infinite polyphony. In practice, this was scaled down to nine "voices"

Touchpoint is an iPad-based performative modular effects processor instrument that focuses on dynamically recombined manipulation of non-linear effects processes as its core synthesis method. The functionally segmented paradigm of modular synthesis (depicted in Figure 24) is *replaced* with a more simplified, holistic set of objects that respond very differently, depending on how the objects are arranged, and the way in which their internal settings are scaled. This is accomplished by creating a new platform based equally in touchscreen modular synthesizer apps and DAW plug-ins. *Touchpoint* presents a new kind of approach to both touchscreen modular synthesizer apps and synthesis technique.

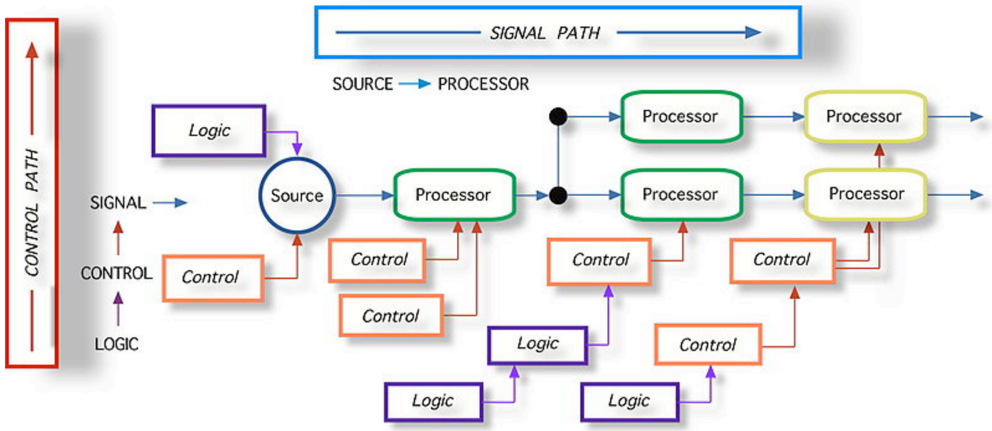


Figure 24. The *audio, control, logic* paradigm of modular synthesis

3.2 The synthesis technique

There are two core principles that influenced the design of *Touchpoint* in such a way that makes it novel among performative signal processors. The first principle effectively justifies effects processing as a mode of music synthesis, and it refers to the use of exclusively non-linear systems in the choice of available processors in the instrument.

3.2.1 Synthesis via non-linear processor combination

In DSP, a *Linear System* refers to any process that exhibits both *static linearity* and *sinusoidal fidelity*. If either of these properties is invalid, then a process cannot be described as a *Linear System*. *Static linearity* refers to the idea that a process can simply be described as the input offset by a certain constant; i.e. a single-process transposition can be observed in a plot of the incoming signal vs. an outgoing one. *Sinusoidal fidelity* refers to the idea that if the incoming input to a process is a sine wave of a certain frequency, that it will maintain integrity through the process and return the same sine wave at output.

These parameters can be additionally described by their *homogeneity* (the observable effect of a linear transfer from input to output) and their *additivity*. *Additivity* refers to the idea that a multi-stage process can still render individually distinguishable input signals. For example, hearing two voices speaking simultaneously does not sound like a compound new person, but rather the reception of two simultaneous, separate voices. (Smith)

For an example of a *Linear System* versus a *Non-Linear System*, think of a simple gain adjustment versus an amplitude modulator running at ring modulation speeds. In each case, a signal's amplitude is modified. However, a gain adjustment is just a simple offset in magnitude. Its output result can be described as a transformation of the input result that is invariant over time. Gain adjustment can be described as a *Linear System*.

On the other hand, amplitude modulation describes a gain adjustment that is driven by a modulating oscillator. When modulated at the audio rate, a bipolar input sine wave produces a pair of output sine waves that have a relationship that is dependent upon the spectral content of the incoming signal, as well as the amplitude and frequency of the modulating oscillator.

One could run several signals through a ring modulator processor and see different output results without an obvious clue as to their relationship in the absence of the input variables. Especially when considering the effect of aliasing above 22KHz and below 20Hz at a typical sample rate, a ring modulator does not exhibit signal-agnostic invariance, and can therefore be defined as a *Non-Linear System*.

This is important because *Non-Linear Systems* produce completely different outcomes depending on the order they are arranged in. The observable *additivity* of *Linear Systems* means that the order of applied processes is irrelevant; each stage can still be perceived in the sum output result. Each *Non-Linear System* will produce a different output result, depending on the upstream transformations that have been performed upon it.

This property of *Non-Linear Systems* describes most effects processing units, and can therefore be used as a useful metric for deciding which effects processors to use in the design of *Touchpoint*. Playing around with the ordering of modules gives the performance framework much deeper mechanic of exploration than if ordering did not significantly matter.

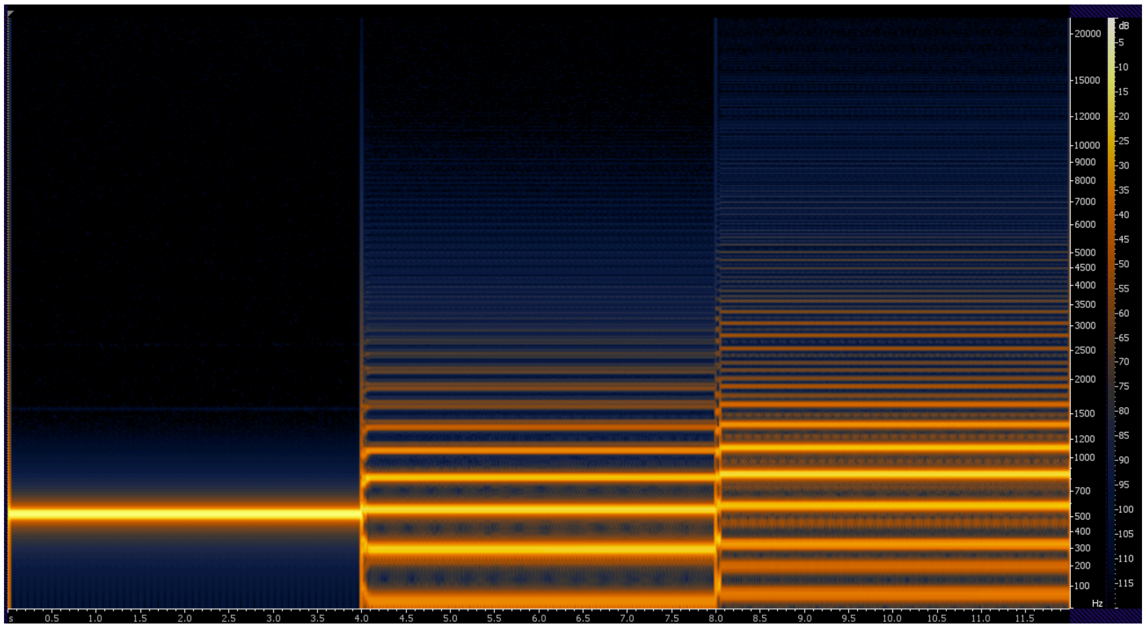


Figure 25. Comparison of order in *Non-Linear Systems*

The principle of dynamic non-linear re-combination is demonstrated in Figure 25. In the test audio from which this spectrograph was rendered in iZotope *RX3*, each configuration is played for four seconds, lasting twelve seconds total. At left, the input audio signal is first played by itself; a sine wave at 440Hz. In the following images, each of the effects processors had the same frequency and wet/dry strength to its respective modulation oscillator.

In the center figure, an *Amplitude Modulator* is placed before a *Frequency Modulator*. In this scenario, the fundamental pitch appears to have been tuned a chromatic half-step up, with the appearance of at least two perfect octaves somewhere below it, and a lot of typical clangorous *Bessel Function FM* activity in the midrange. In the figure at right, the same *Frequency Modulator* is placed after the *Amplitude Modulator*. The resulting sound has a dominant fundamental a whole step above the original pitch, with many overlaid pure tones above it that make it sound like a two-tone DTMF dial tone-type sound. As can be inferred from the Figure, many more pure frequencies are generated in this configuration than with the previous configuration.

If these processors had linear effects, their order would not matter; both cases should produce the same spectrum. However, each of these processors subtract and insert their own spectral content and amplitude envelope modifications, which are dependent upon the signal being fed to them, thus creating a dynamic output result that is dependent upon order.

3.2.2 The "perceptual modulation spectrum"

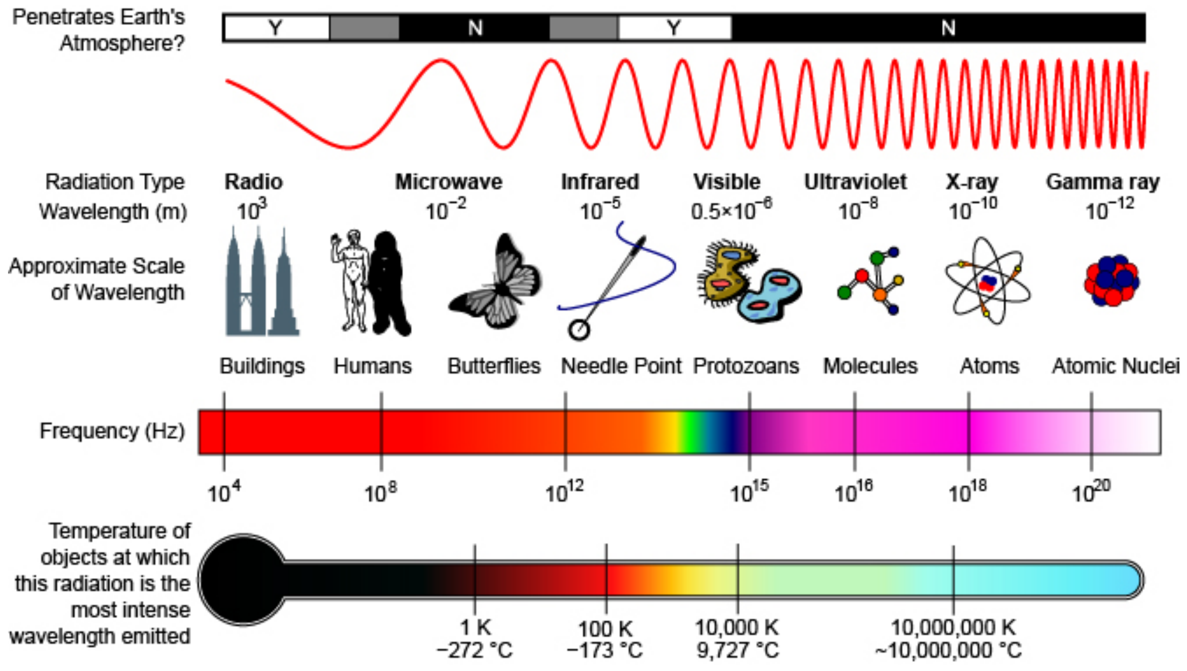


Figure 26. The electromagnetic spectrum

I like to call the principle which *Touchpoint*'s choice of available objects something like "the perceptual modulation spectrum", because it indicates the idea's relationship to the electromagnetic spectrum (Figure 26). Radio waves, compared to visible light and gamma rays, are typically thought of as separate phenomena, each with their own physical properties, uses, perceptual effects, and technology built to interact with them. In fact, they are all electromagnetic radiation of wavelengths separated by orders of magnitude.

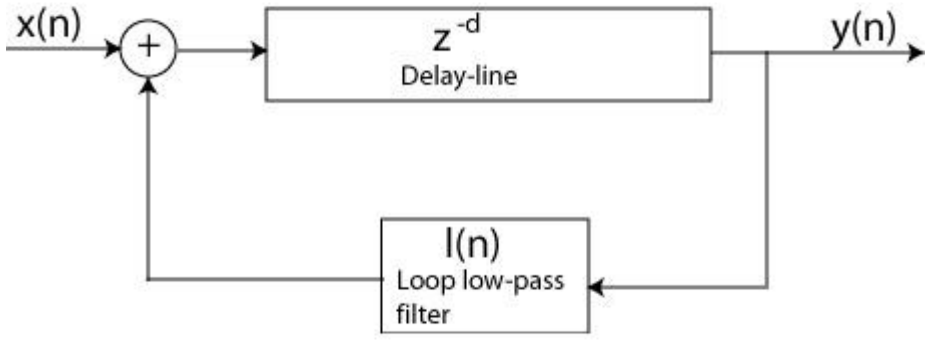


Figure 27. The Karplus-Strong plucked string algorithm

In a very simplified manner, this same unification can be presented to many of the different audio processing outcomes that are typically circumscribed as separate processes. For example, iterative, decayed repetitions in time of an input signal could describe a localization effect (Haas), an echo, a delay, or a resonating string (Karplus and Strong) (see Figure 27). They are merely separated by orders of magnitude with regards to the number of repetitions, their decay coefficient, and the amount of time that passes between them. This time could be represented in the digital domain as samples per second, given a sample rate.

This idea was inspired by the way BT and iZotope presented the choice of repetition sizes available for buffer size traversal in their product *Stutter Edit*. In the *Stutter Edit* GUI, all buffer size-based modifiers are reading from a table of selections listed at the top of the plug-in. This table is represented as a table of two halves, comprising rhythmic values at tempo and chromatic frequencies extending four octaves. Thus, *beat repeat*-type effects, *glitch edit*-type effects, *granular synthesis* (Roads) and *PCM*[44]/*wavetable synthesis* are all unified into one buffer-based object.

[44] Pulse code modulation.

To my knowledge, this is the first commercial product to represent rhythmic values and chromatic frequencies as one continuous spectrum. This representation (shown in Figure 28) sparked the line of thinking implemented in *Touchpoint*, where it has been nearly copied, with greater precision.

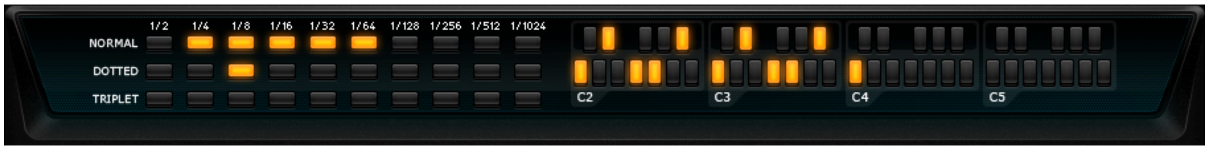


Figure 28. The upper portion of iZotope Stutter Edit's GUI

After studying its behavior closely, it was observed that, in *Stutter Edit*, depending on what the master tempo is set to, when sweeping upwards or downwards across the selected options, the table of four octaves of chromatic notes and the table of normal, dotted, and triplet rhythmic values begin to interweave with each other at different points. This makes sense; despite their graphical separation, these two tables interweave dynamically because one of them (rhythmic values) changes its values in samples per second at sample rate depending on a master tempo constant, and the other (chromatic frequencies) is just a static lookup table of buffer sizes.

After a certain minimum buffer size near the ~20-60Hz beginning of the audible frequency spectrum, the exact point of which depends on the input material used, looping buffers perceptually lose whatever their original pitched root note was, increasingly being dominated by an octave transposition of whatever the nearest root note of the chromatic frequency at that buffer size is. Increasing or decreasing the buffer size tends to simultaneously tune the pitch as well as modify the periodic sound's timbre. A looping buffer becomes a wavetable synth.

In *Stutter Edit*, there are a total of 74 buffer size options available to the user, which stretch from a dotted half note at tempo (three quarters of a musical bar) to MIDI notes C2 through B5. There are practical considerations for this clipped range of notes - MIDI note B1 is the closest number to 60Hz, which is barely audible as a pitch for looping buffers of audio material under most conditions. Likewise, note B5 is near 2KHz - most likely, input material with a fundamental frequency above this range is an error; it is unlikely. Also, beginning at ~60Hz is a sensible crossover point from the tiniest unit available in the rhythmic indices table: the triplet 1024th note. Nearly regardless of the master tempo set, the relationship between these tables displays a crossing over roughly halfway across the total run of selected indices.

3.2.2.1 Implementation in Touchpoint

In *Touchpoint*, there are 161 indices available, because on the rhythmic unit list of the two tables, I have included the full range of a full musical bar to triplet 1024th notes, and on the MIDI frequency list side, I have included all 128 chromatic MIDI notes. At 120 BPM[45], this spans a range from 0.5Hz to 12,543.85Hz. MIDI note "0" is just over 8Hz, so these lists start weaving together much sooner than in *Stutter Edit*'s version.

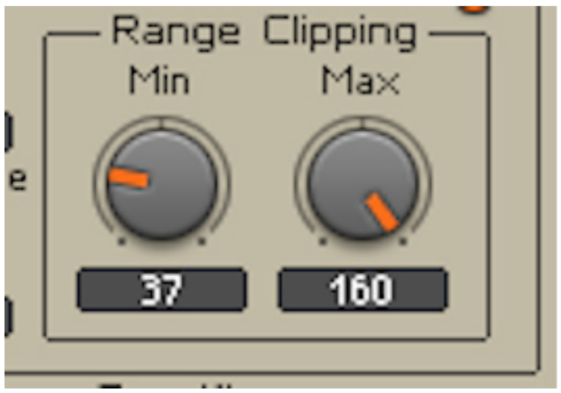


Figure 29. How *Touchpoint*'s quantization indices are represented in *Reaktor/Lemur*

This index list (depicted in Figure 29) unifies how each of the three processor modules of Touchpoint respond across the X-axis. Because I chose three parametric characteristics that align with my *perceptual modulation spectrum*, each module explores the minimum and maximum domains of several different perceptual phenomena, and sometimes the transitional space between them, such as the critical threshold between *glitch editing* and *wavetable synthesis* that was previously described regarding the buffer-based module.

Unfortunately, because *Reaktor* cannot accept dynamically re-arranged arrays of strings, these values must be expressed - rather obtusely - in the interface as integers ranging from 0 to 160. This also makes pinpointing an index value associated with a particular desired value that would live somewhere between the two interweaved lists, such as a 32nd note, or the note C2, difficult and different to locate every time.

[45] Beats per minute.

This has led, in my performance practice, to the memorization of certain shorthand ranges. For instance, if I want to deal exclusively with large rhythmic values, I know that it is a safe bet to scale a processor's range down to about 0 to 33. Likewise, if I wanted to deal with midrange chromatic notes, I'd set the processor's X-axis range to something like 60 to 100. If I want high-pitched, alias-heavy artifacts, 120-160 should do the trick.

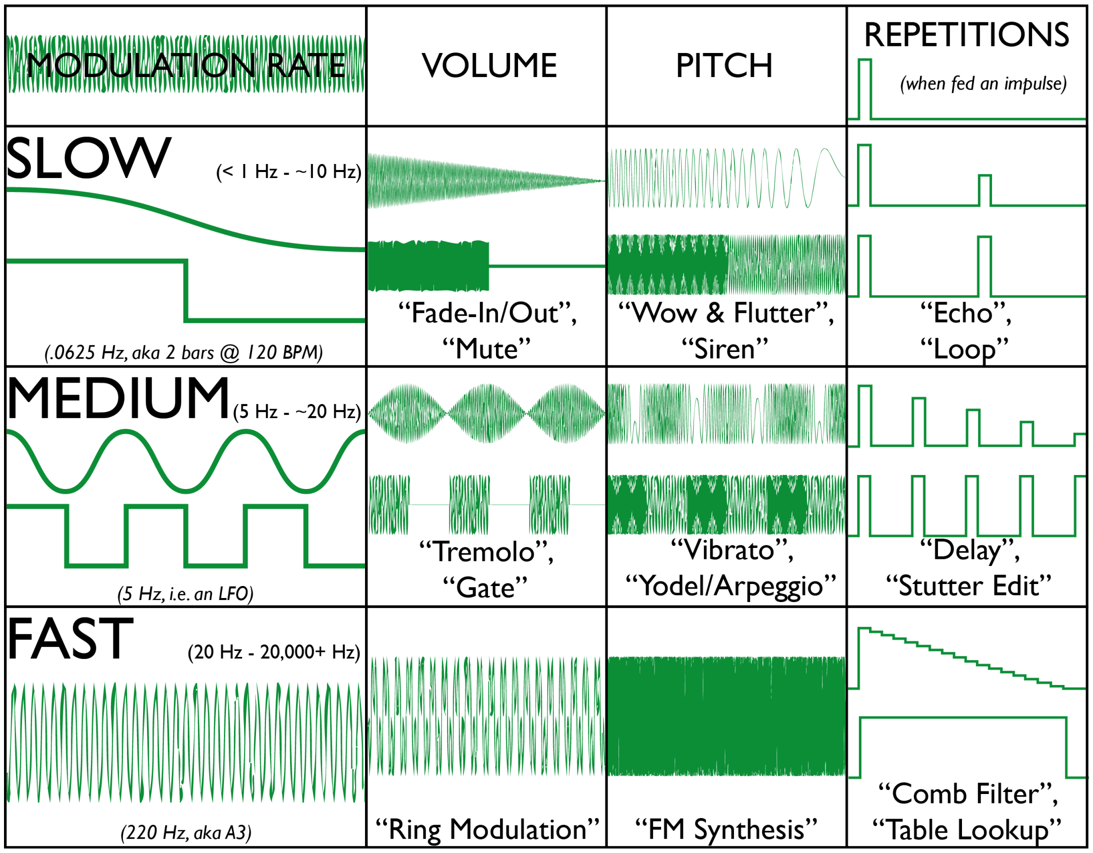


Figure 30. A simplified summary of multi-range modulation perception

A summary of each of the three modulation parameters as they are swept across the "perceptual modulation spectrum" - from below 1Hz to over 20KHz, with sinusoidal and pulse waveforms - given an A440Hz sine wave input signal, is depicted in Figure 30.

In practice, the 161-value table calculated per each master tempo update is applied to the X-axis value of each processor, which simply comes in as a 32-bit floating point number from 0 at far left to 1 at far right. If the current processor is a *Comb* and its type is set to *Stutter*, the relevant memory space in *Touchpoint*'s bank of processors will be sent an integer buffer size. (Floating-point buffer size interpolation has yet to be added.) If any other processor or mode is selected, the relevant memory space receives a floating-point frequency value in Hz, given that floating-point X-position's nearest index of the possible 161.

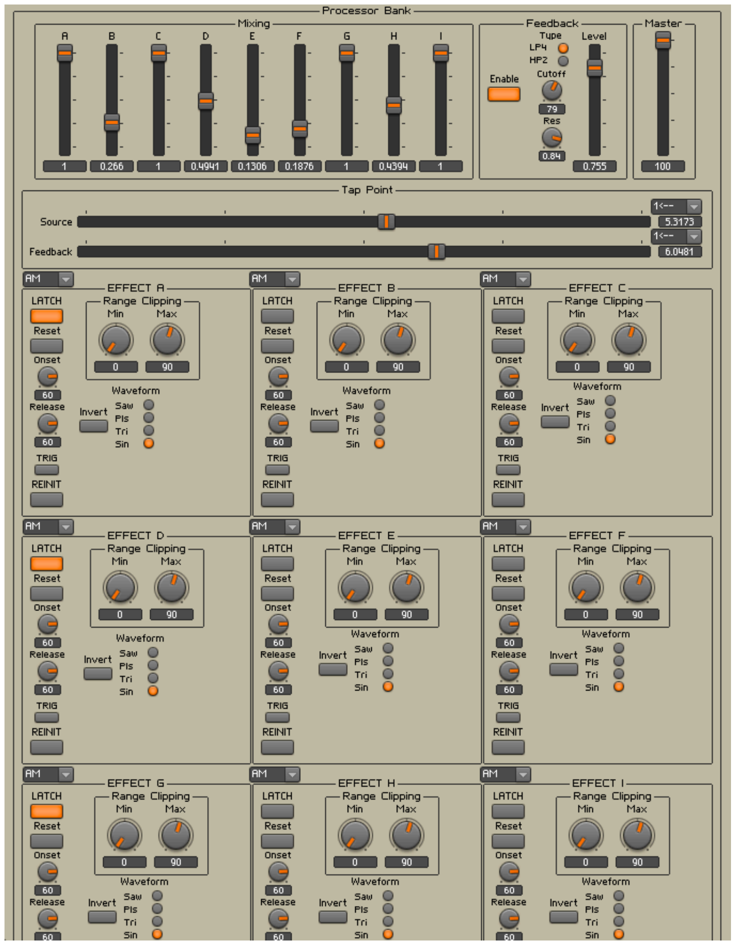


Figure 31. "*Side B*" configuration page of *Touchpoint* version 0.5.10a in *Reaktor*

Reaktor was chosen as the platform of choice for executing these synthesis methods - depicted in Figure 31 - over other popular audio development languages with similar functionality such as *Max/MSP*, *Pure Data*, *SuperCollider* and *Csound* for several reasons:

- *Reaktor* is optimized for the best possible sound quality at every stage of prototyping. *Max* and others may be clearer in programming style, requiring much less environment-specific safeguarding, but they require more sound-design and non-linearity modeling (i.e. tweaking) in anticipation of building release-ready products (ergo, things that just sound good from the beginning).
- By using a construction platform that can also run as a plug-in inside of a DAW, we inherit all of the built-in handlers for external clock synchronization, master tempo, parameter automation, etc. afforded to the VST/AU/RTAS/AAX et al. formats. This

would otherwise be a tremendous amount of external SDK[46] and API work to piggyback off of. This could be gained back in another environment, possibly by using *ReWire* and IAC[47] MIDI busses, or objects like "transport" in *Max*, but then MIDI ppqn[48] becomes involved in a way that shouldn't be necessary. If *Max for Live* was used, this would also piggyback onto a master DAW's features, but then the developer's hands would be tied to exclusively using Ableton *Live*. None of these options can be packaged as a self-contained group of files or as an installer.

- *Reaktor* is one of very few mainstream music production products that features just as robust of an OSC mapping implementation as it does MIDI. As of *Reaktor* 5.8, MIDI and OSC are even unified under the same "Learn" menu. The implementation is not perfect: *Reaktor* cannot parse or generate strings, and sending strings to *Reaktor* crashes the program. But, having the ability to use OSC in a mainstream product with no middleware translation is very useful.
- The Music Technology department of CalArts is sponsored by Native Instruments. The department encourages strong engagement with Native Instruments platforms. My internship in Berlin with the Research department of NI headquarters was the result of this partnership, and it allowed me to develop *Touchpoint*'s core concepts with some of the world's best *Reaktor* builders and product designers.
- Due to his tutelage under *Reaktor* developer Martijn Zwartjes, my advisor at CalArts, Owen Vallis, is fluent in communicating DSP concepts specifically using the *Reaktor* platform. Coming into CalArts primarily with experience in *Max/MSP* and *Csound*, my learning to use *Reaktor* would allow us to examine ideas together most effectively.
- Given its historical use in the JazzMutant era, *Lemur* is highly optimized to interface specifically with *Reaktor*. Native Instruments[49] and Liine[50] each provide extensive support for interfacing *Lemur* with *Reaktor*. Both companies publicize a folder of *Lemur* .jzml layout files that interface with a set of modified *Reaktor* Library Content Ensembles with pre-configured OSC handlers to work with the *Lemur* layouts[51].

[46] Software development kit, here, referring to the Steinberg VST 2.4 SDK, for example.
[47] Inter-application communication.
[48] Pulses per quarter note.
[49] http://www.native-instruments.com/en/support/knowledge-base/show/1000/how-to-set-up-lemur-with-reaktor-5.8-mac
[50] https://www.liine.net/en/support/lemur
[51] https://www.liine.net/en/products/lemur/premium/ab-lemur-bundle

- The *Reaktor Core* environment executes compilation and execution instructions at a lower level than the pre-compiled ("*Primary*") runtime environment, close to the processor hardware. Audio processing environments and projects tend to get very complicated very quickly, and the ability to leverage low-level execution with visual programming is favorable.

3.3 The interface

In this section, the control interface of *Touchpoint* on the *Lemur* interface is described. It begins with an overview of how to select objects for the environment and what each of their functions is. It then goes on to describe how to select the input audio signal source, as well as an explanation of how the "perceptual modulation spectrum" has been presented on the interface. The mechanic of activating the input signal from the processor fields is described.

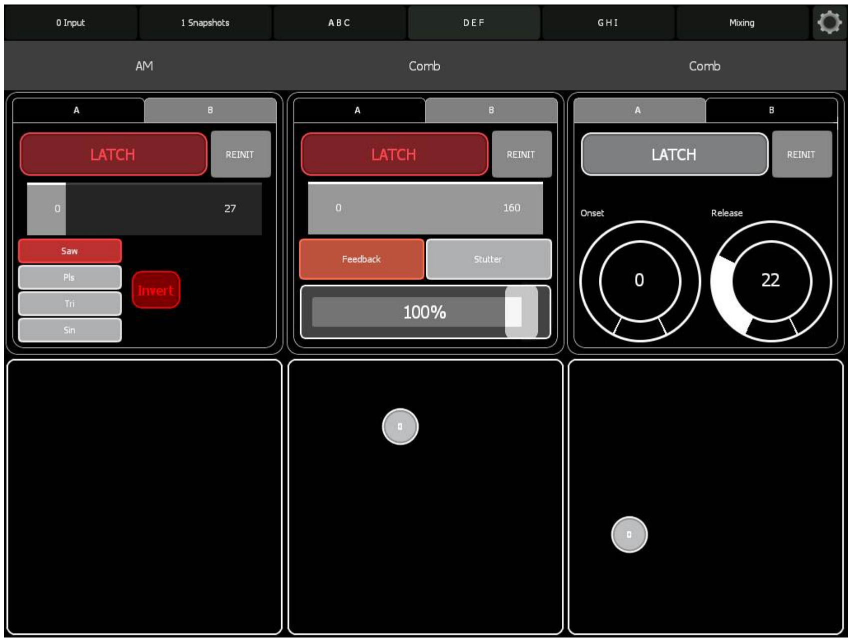


Figure 32. Version 0.5.10a of one of *Touchpoint*'s main *Performance Pages*

Pictured in Figure 32 is the typical appearance of one of the main *Performance Pages* in *Touchpoint.* There are three pages of three available effects slots, each one of which can contain one of three effects types. A slot can also be set to "OFF" which has no effect when touched.

Each processor was selected for its capacity to represent many different phenomena along a perceptual spectrum, and for its versatility in application. The menu for selecting them is detailed next, followed by an in-depth examination of each of the processors.

3.3.1 Object choices

Table 2. List of Interface Parameters, Global and Effect-Specific

Device	Parameters
GLOBAL	Onset, Release, Latch, Reinit, TouchEnv, Min Freq, Max Freq
Amplitude Modulation (AM)	Waveform, Polarity, Invert, Pulse Width
Frequency Modulation (FM)	Waveform, Polarity, Invert, Pulse Width, Depth
Comb	Type, Wet/Dry

The user selects between one of three objects using the menu shown in Figure 33. Their specific configuration options are listed in Table 2.

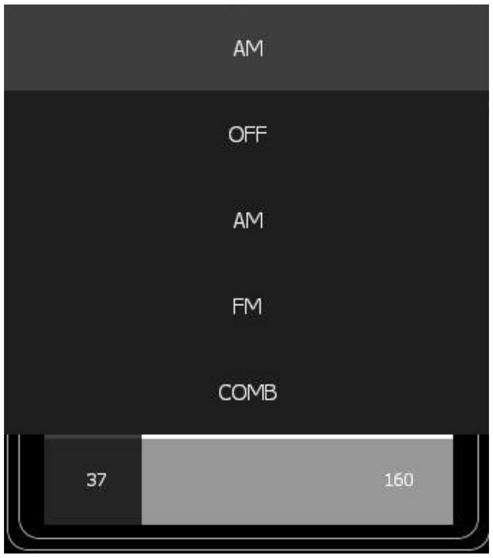


Figure 33. Effect selection drop-down menu per effect slot

The *AM* object changes the input's gain, with switchable polarity. Bipolar gain modulation inverts the phase of the incoming audio signal for half a cycle. The *FM* object has a crude zero-crossing counter-based pitch tracker at the input, as a means to perform realtime pitch shifting on live input, specified in semitones, also with switchable polarity (root-up or up-down). The phase of both of these modulating oscillators can be inverted, and can also be switched from sinusoidal to a pulse wave (with variable pulse width), a sawtooth, or a triangle. The *Comb* object is a sample buffer with two modes; with or without feedback. The *Feedback* mode acts as a rudimentary Karplus-Strong plucked string (Karplus and Strong) without a filter, and the *Stutter* mode simply loops input material at the specified length until release.

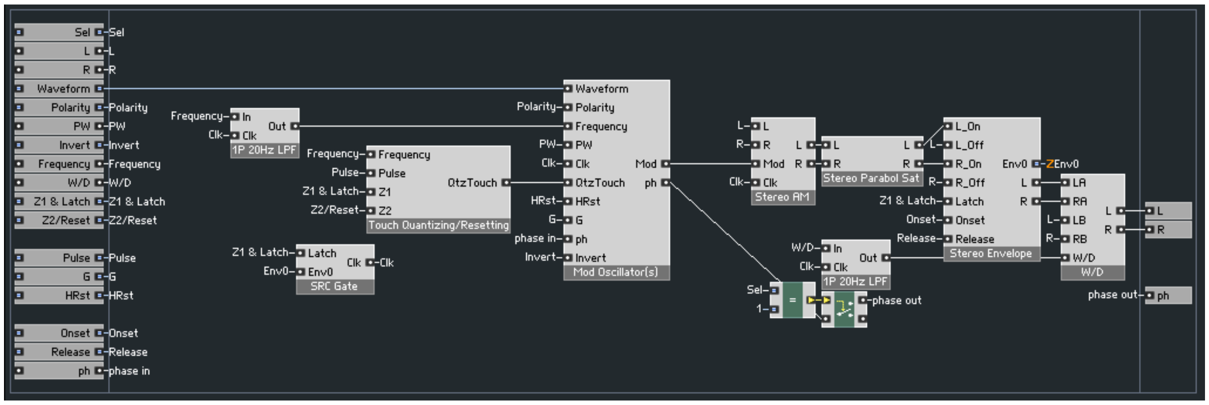


Figure 34. Top-level overview of the *Amplitude Modulation* effect in *Touchpoint*

3.3.1.1 AM effect

Table 3. Perceived Effects of Modulation of Gain by Sinusoid and Pulse Wave

Modulation Rate	Sinusoid	Pulse Wave
Slow (<1Hz-5Hz)	Fade-In/Out	Mute
Medium (5Hz-20Hz)	Tremolo	Gate
Fast (20Hz-20KHz)	Ring Modulation	Ring Modulation

The *Amplitude Modulation* module (shown in Figure 34) is essentially nothing more than the incoming audio signal multiplied by the value of an incoming periodic oscillator to change its gain, as well as phase, depending on the polarity setting of the modulating oscillator (see Figure 35). Each of the global and effect-specific controls is laid out in the settings page above the processor's XY field (see Figure 36). The application of the perceptual modulation spectrum upon incoming gain is expressed in Table 3.

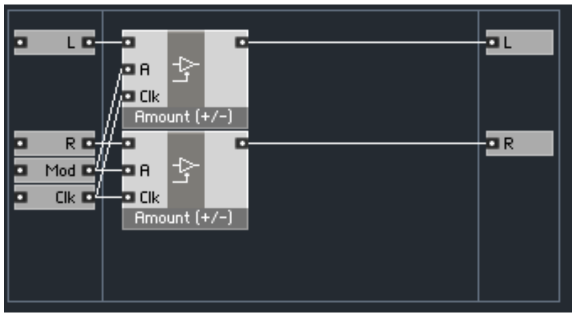


Figure 35. The *Amplitude Modulation* module's guts... it's just an amplifier

Audio is only piped into the module when the *Sel* input for that slot is set to 1, which indicates *AM*. Based on a menu selection, one of four basic waveform shapes (sine, triangle, sawtooth, pulse wave with variable pulse width) is rendered by modulation oscillator block.

Based on a *Polarity* flag, the multi-waveform modulation oscillator will either output from -1 to 1 or from 0 to 1. There is generally no need to use the *Bipolar* option at rhythmic units, since all that will change about the incoming audio is that the phase will be inverted 90 degrees for half of the cycle. At slower rates like these (sub-1Hz or so), generally only the *Unipolar* mode should be used, which will fade the incoming audio up and down out of silence.

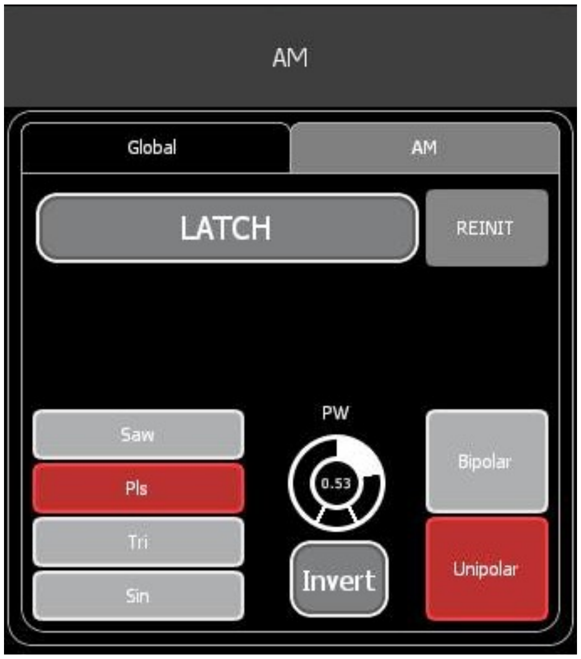


Figure 36. *Lemur*-side effect-specific menu for the *AM* effect

Based on the incoming *Frequency*, which is received by translation from that XY field's X-axis position and the scaled range assigned to it, slow rates will produce fades in *Unipolar* mode. At faster rates, a sinusoid will produce *sum tones* only at the frequency of the root note of the incoming audio signal plus the frequency of the modulation oscillator. This sum tone and the original note will be heard simultaneously. In *Bipolar* mode, the *sum and difference* tones of true ring modulation will be produced. These will be heard as the aforementioned *sum tone*, as well as a *difference tone*, described as the root note of the incoming audio minus the frequency of the modulating oscillator. In *Bipolar* mode, only the *sum and difference tones* will be heard; the original pitch of the incoming audio will be lost.

Interestingly, it is very easy for these *sum and difference tones* to wrap back around over 0Hz and 22,050 Hz (or whatever the Nyquist frequency (Nyquist) of the *Touchpoint* session is), respectively. This can cause the system to respond with two relatively close midrange frequencies at extremely high modulation frequencies, sometimes refracting twice over the critical bands of aliasing.

If using waveforms other than the pure sinusoid at audio frequencies, the resulting response will be more harmonically complex than two frequencies per one input frequency. Ostensibly, they will follow the rules of their harmonic construction: a pulse wave or triangle wave will produce *sum and difference tones* at every odd harmonic, a sawtooth wave will produce *sum and difference tones* of all harmonics falling off at -6dB/octave, etc.

If the pulse wave is selected, a variable pulse width knob will be presented in the interface that moves from -0.99 to 0.99. At slow rates with *Unipolar* polarity, this can effectively create the sound a shuffle to a gated step sequence. At faster rates with either polarity, moving the pulse width "thins" or "thickens" the resulting timbre, as one would expect with traditional keyboard synthesizers, except with the input oscillator being an incoming audio signal.

If an *Invert* flag has been selected in the interface, the outgoing value of the modulating oscillator waveform is multipled by -1, resulting in an inverted phase. An interesting application of this is to take a slow frequency, *Unipolar* sawtooth waveform and invert it, so that rather than acting as a linear fade-in, it sounds as though it's behaving like a sharp attack-release envelope which is being constantly pinged by a step sequencer, starting at full amplitude and fading down to silence over the exact duration of the wavelength period.

The raw 0 to 1 value of the Y-axis from the XY field is simply brought to a power of 0.4 for a logarithmic scaling which responds more naturally to the range of the adjusting finger before being send as a 1 to 0 value for adjusting the *Wet/Dry* balance of the *Amplitude Modulation*.

Hooks are placed into the running phase of the modulation oscillator which will be sent out of the master *Core* cell upon *Snapshot* storage, and which can be piped and merged into the running modulating oscillator phase upon *Snapshot* recall.

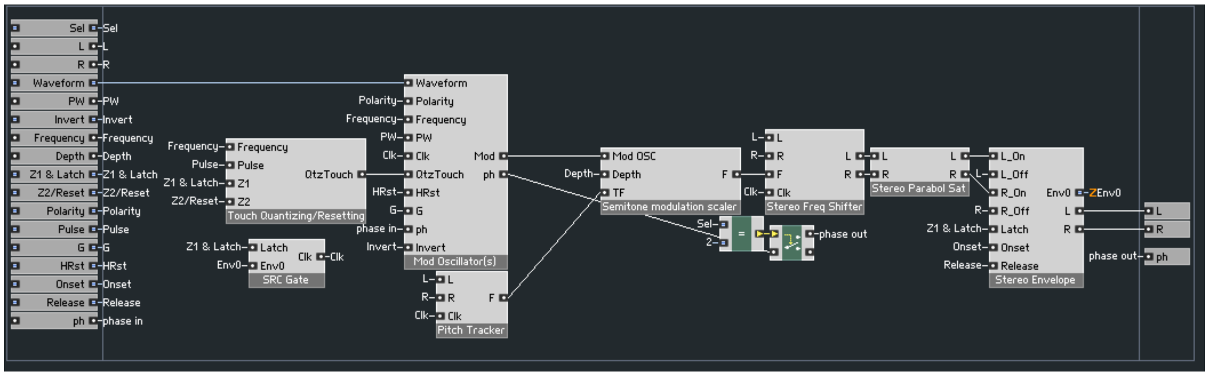


Figure 37. Top-level overview of the *Frequency Modulation* effect in *Touchpoint*

3.3.1.2 FM effect

Table 4. Perceived Effects of Modulation of Pitch by Sinusoid and Pulse Wave

Modulation Rate	Sinusoid	Pulse Wave
Slow (<1Hz-5Hz)	Wow/Flutter	Siren
Medium (5Hz-20Hz)	Vibrato	Yodel/Arpeggio
Fast (20Hz-20KHz)	FM Synthesis	FM Synthesis

The *Frequency Modulation* module (shown in Figure 37) was created around a principle I had in mind that required some extra research in order to implement. If changing the gain of an incoming audio signal in an unquantized, perceptually smooth way is such a simple mathematical operation, shifting its pitch should at least superficially be as well, even though I knew it wasn't as simple.

Several pieces of pre-fabricated *Reaktor* programming were re-tooled in order to suit my purposes. The application of the perceptual modulation spectrum upon incoming pitch is expressed in Table 4, and its effect-specific controls as they appear on the interface are shown in Figure 40.

Something needed to be created that could receive a modulation oscillator whose output describes +/- a certain amount of semitones and performs that transposition accurately - despite the logarithmic mapping of frequency across pitch indexes. If this processor was successful, not only could legato pitch shiftings be performed, but the processor could be sped up at audio frequencies to do clangorous, inharmonic *FM Synthesis* on realtime audio input.

The algorithm I used for pitch tracking inside *Reaktor* actually comes from another Tim Exile/Native Instruments which required excessively precise pitch tracking of a monophonic input signal, ostensibly a voice coming through a microphone: *The Mouth*.

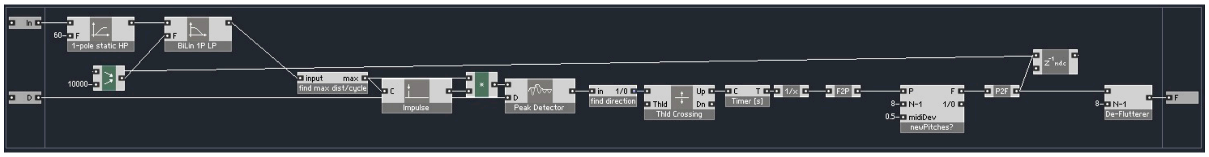


Figure 38. Pitch tracking algorithm, modified from Tim Exile's The Mouth

The pitch tracker found in *The Mouth* (shown in Figure 38) begins with a notch filter centered around a reasonable range for the human voice - a low-pass filter at 60Hz and an adaptive threshold at 10KHz, which is updated by further analyses. This is a basic erroneous data safeguarding measure - a filter set by a spectral envelope follower. From here, audio is fed into a magnitude estimator that pings an impulse generator, which generates unipolar impulses at the magnitude detected within a certain comparison period.

This gets fed into a peak detector, which drives a zero-crossing counter whose positive impulses drive a time co-efficient that is then converted from a period to a MIDI pitch, with pitch bend offset expressed in the mantissa. Essentially, this is a slightly optimized and modified zero-crossing counter-based pitch tracker. Such techniques have been in use in DSP since the early 1980s.

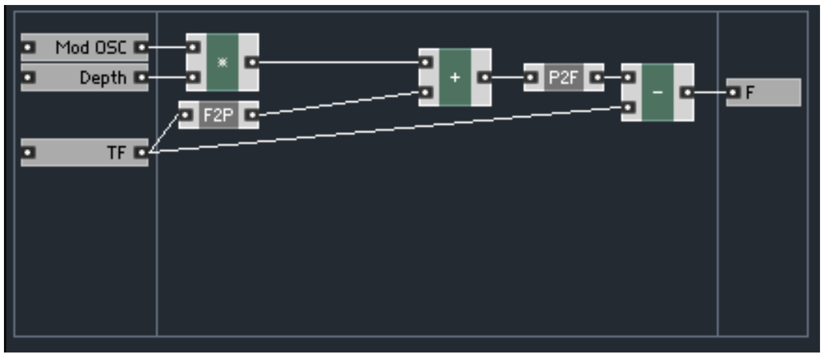


Figure 39. Semitone modulation scaler algorithm

This tracked pitch (expressed in Hz as a frequency) is now sent to a new algorithm (shown in Figure 39) which accepts three inputs: the tracked frequency, the current -1 to 1 (or 0 to 1) position of the modulating oscillator, which works identically to the modulation oscillator of the *Amplitude Modulation* block, and a *Depth* argument from the UI which is an integer indicating chromatic semitones from 0 (no change) to 36 (three octaves difference).

The full-range modulating oscillator is multiplied by the *Depth* argument, so that, for example, a bipolar octave's difference would change the modulating oscillator as going from -1 to 1 to going to -12 to 12. The tracked frequency is converted back to a MIDI pitch and added to the chromatically re-scaled modulation oscillator value, creating a net modulated delta pitch sum. This combined block is then converted back to a frequency and subtracted from the original tracked frequency. The end result of this process is a tracked frequency moving up and down the appropriate amount of Hertz to describe the exponential shift necessary to perform the shift as an input to a pitch shifter of some sort.

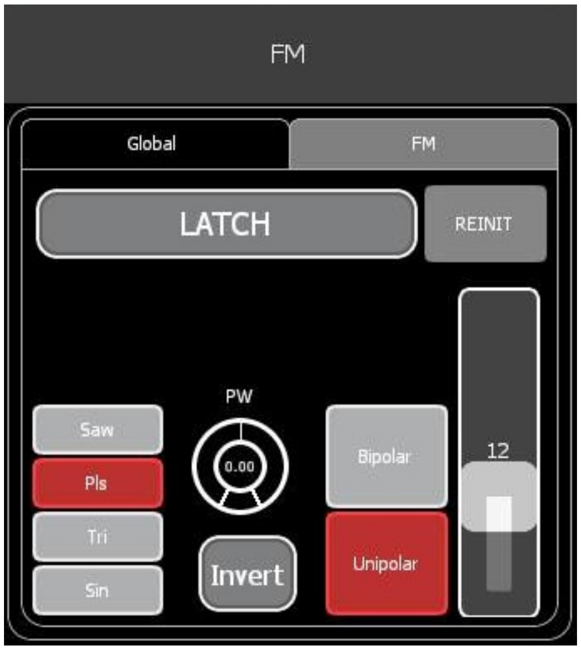


Figure 40. ***Lemur*-side effect-specific menu for the *FM* effect**

The exponentially-corrected shifter frequency vector is passed to *Frequency Shifter* Macro which is pre-fabricated from the *Reaktor* Factor Library. This Macro appears to involve phase shifting and some kind of Hilbert Transform-esque method to achieve its goal. This technique was deemed to be the appropriate choice since Fourier-based *Reaktor* techniques could not withstand receiving transposition messages so quickly without fracturing in sound quality and CPU usage.

As previously described, the *Frequency Modulation* effect's modulation oscillator block behaves exactly the same as the *Amplitude Modulation* effect's modulation oscillator. In application, this means that *Bipolar Polarity* settings will shift the pitch of the incoming sound both up and down from the originally detected pitch, which can obscure the original pitch entirely. *Unipolar* settings will shift the incoming pitch up the specified amount; flipping the *Invert* flag will make it travel that amount down instead.

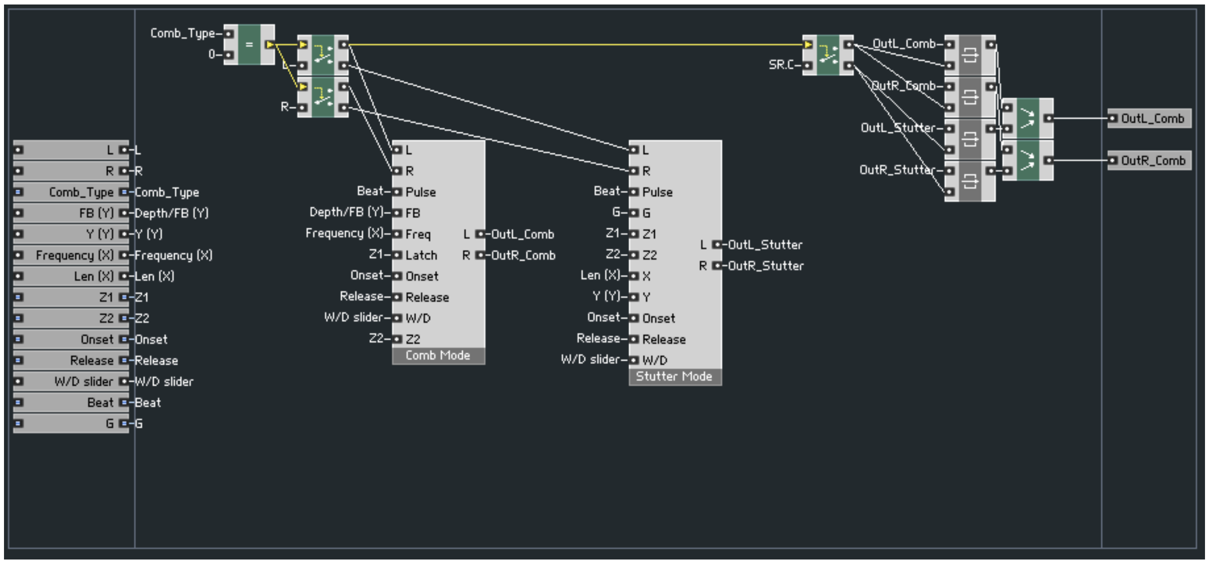


Figure 41. Top-level overview of the *Comb* effect in *Touchpoint*

3.3.1.3 *Comb effect*

Table 5. Perceived Effects of Buffer Repetition, with and without Feedback

Modulation Rate	with Feedback	without Feedback
Slow (<1Hz-5Hz)	Echo	Loop
Medium (5Hz-20Hz)	Delay	Glitch/Stutter
Fast (20Hz-20KHz)	Comb Filter	Table Lookup

The *Comb* module (shown in Figure 41) is constructed differently from the *AM* and *FM* modules, not only because it is not driven by a waveform oscillator, but also because it is fundamentally two sub-effects that are routed and selected by a top layer. So, not only does the

Sel flag have to match 3 for an effect type of *Comb*, but a *Comb Type* is specified; 0 for *Feedback* and 1 for *Stutter*.

The application of the perceptual modulation spectrum upon the repetition size of a buffer of incoming audio is expressed in Table 5, and its effect-specific controls as they appear on the interface are shown in Figure 43.

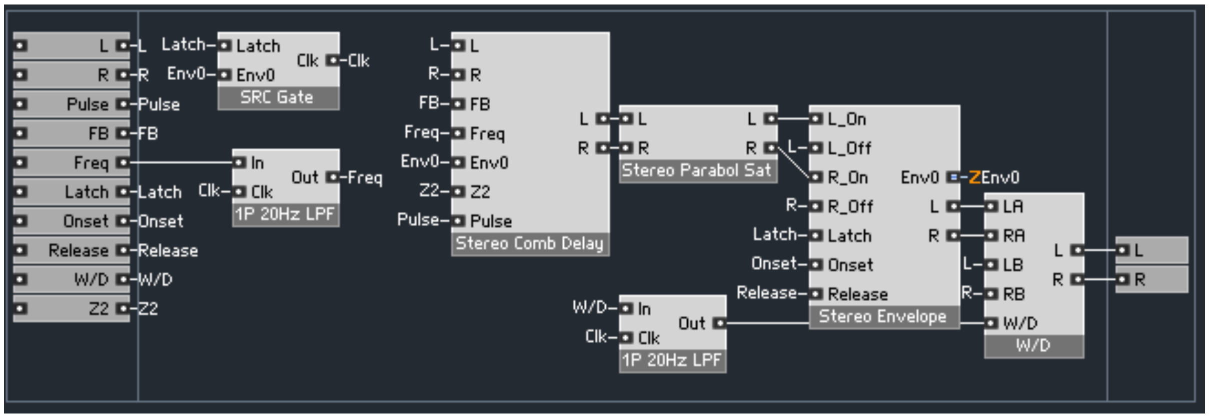


Figure 42. Top-level overview of the *Feedback* type of the *Comb* effect in *Touchpoint*

The structure of the *Feedback* sub-effect (seen in Figure 42) is familiar: a first-finger activation - or the end of a release envelope - clocks the *Sample Rate Clock* pulsing into the effect. The *Frequency* argument stemming from X-axis movements is low-pass filtered - as is a *Wet/Dry* filter, this time, from an effect-specific page for the *Comb* block.

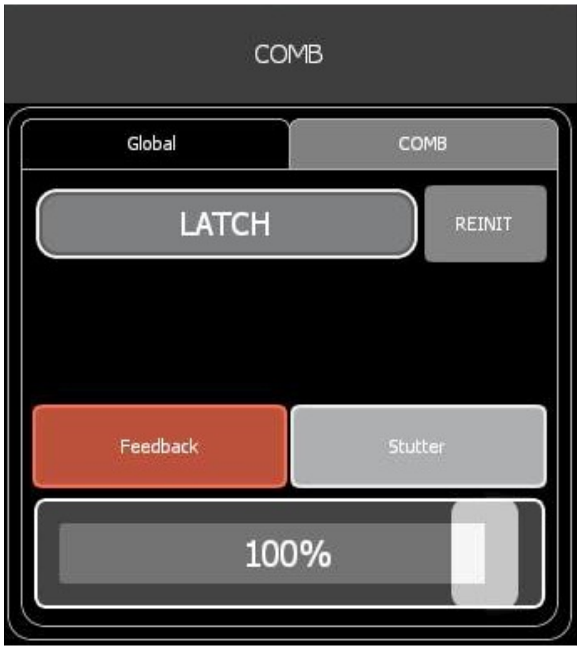


Figure 43. *Lemur*-side effect-specific menu for the *Comb* effect

Inside the *Feedback* algorithm (shown in Figure 44) is a filter-less version of the Karplus-Strong plucked string algorithm (Karplus and Strong). The audio signal enters into one channel of a two-channel mixer, where it passes through a four-point interpolated unit delay of a time specified by a frequency on the X-axis. This is then multiplied by a feedback coefficient between 0.5 and 1, provided by a mapping to the Y-axis. The delayed signal is recursively fed back into the second input of the two-channel mixer.

An algorithm metered by the envelope threshold used for the clock driving algorithm, incoming master tempo 16th note pulses, a possible reset message from a second finger and the incoming X-axis frequency makes sure that the signal from the previous time the delay unit was used does not reach the output stage before the incoming audio does, draining the line after the release envelope has closed.

As long as a finger is active, a multiplier for the delayed path is set to 1, allowing it to pass through into the recursion loop. As soon as the effect has ended, this "boolean timer" is set to zero, draining the line as it gets clocked out. Likewise, a second finger tap will also drain the line and begin re-filling it at the next master 16th note. This lends an aspect of a performative delay unit that is somewhat unique to *Touchpoint*.

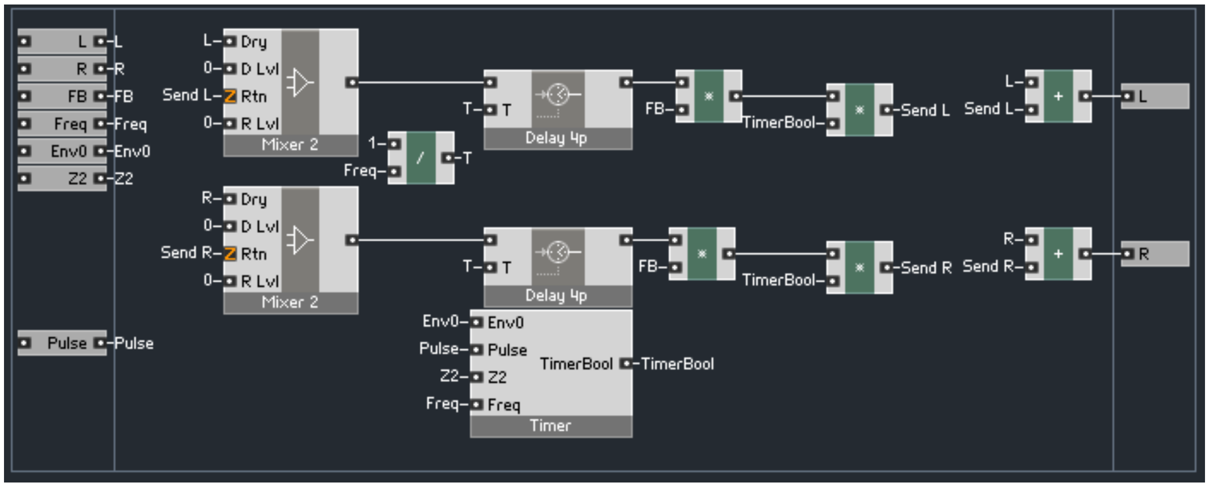


Figure 44. Inside the *Feedback Type* of the *Comb* effect

Once again, a large chunk of the code of *Touchpoint* is indebted to Tim Exile. The *Stutter* mode of the *Comb* effect (shown in Figure 45) is actually only a slight modification of Tim's *Glitch* Macro that is provided in the *Reaktor* Factory Library of *Core* Cell Macros. It has been cleaned up and slightly better annotated, and some of the trademark mechanisms behind all of the other effects, such as the envelope-recursive *SRC Gate* algorithm, 16th pulse quantized touch latching, and a crossfading envelope have been added.

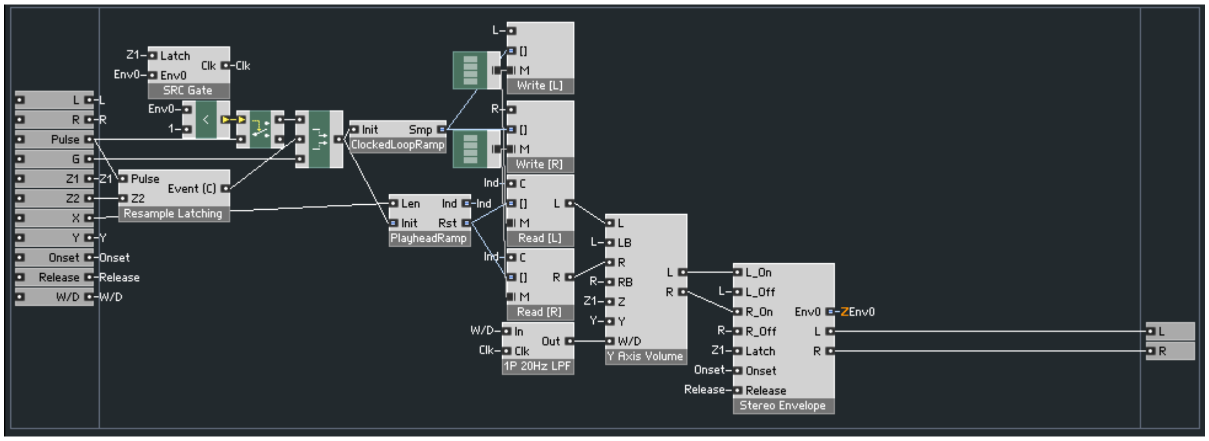


Figure 45. Inside the *Stutter Type* of the *Comb* effect

This is the only module that does not process the Y-axis input in any way - it simply maps from 0 to 1 as a volume scaler for the master output of the stuttered buffer.

A second finger tap will re-sample the stutter buffer at the next master 16th note. This is a very pleasing and dramatic effect - sliding up in frequency (and therefore down in buffer size) will re-sampling as fast the system allows (16th notes, by design) creates some very cool, tonal, formant frequency-esque *wavetable synthesis* sounds.

3.3.1.4 Object summary

The combination of each processor's versatility of application, in combination with the dynamic results of non-linear system combination, allows for an extremely potent set of tools that are capable of going in any number of directions with the slightest parametric change. By changing the frequency of an oscillator modulating a signal's gain, what once sounded like an amplitude envelope being periodically re-triggered has now placed a mirror to the root note of the incoming frequency and folded its spectral character around a pivot point. By changing the frequency of an oscillator changing a signal's gain, the sonic result could move from a subtle "wow and flutter" effect to a dramatic FM Synthesis timbre with one motion.

Whereas the use of the objects themselves within the main "chaining" interface has been described, there are many other factors before and after the main performance processors that much be accounted for in the interface, preparation-wise.

3.3.2 Input selection

Touchpoint begins with a input signal selection page (Figure 46). The user can select between an incoming stereo audio signal, or to use a simple multi-waveform subtractive synthesizer with a state-variable filter that is placed at the beginning of the chain. Regarding this input synthesizer, the user can select whether or not to turn the amplitude envelope on or off, meaning that the user can manually trigger the sound of the synth using an external MIDI keyboard, *Touchpoint* effect slot activations, or just let it constantly run through the system.

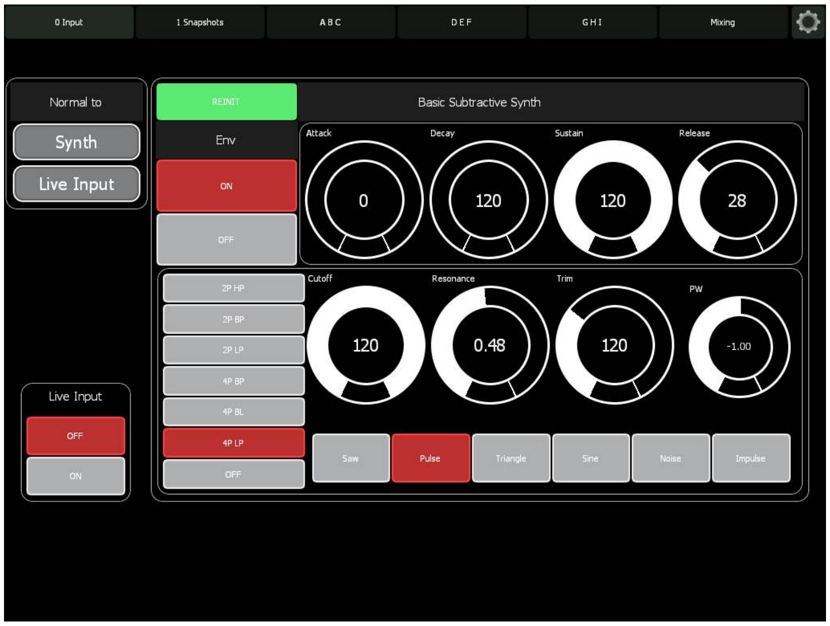


Figure 46. ***Input Selection/Synthesizer*** **page of** ***Touchpoint*****'s interface**

3.3.3 Range scaling

The resolution and scalability of the modulating oscillators that drive each of these objects is designed to allow for crossing over from musical timeframes, through control rates, to audio rates. See Section 3.2.2 for more information about this.

3.3.4 Touch interaction

Touching an XY field associated with a certain effects slot places the effect - with the relevant mappings for the X-axis (normally a frequency or buffer size in samples of some description) and the Y-axis (normally a depth or volume control of some sort) at the front of the effect chain. The effect introduction is dictated by the "*Onset*" and "*Release*" time constant knobs, from 0 to 80, in the *Global* page of every processor (shown in Figure 47). Each effect introduction and exit abides by that slot's envelope settings. Each effect has a "*Global*" page that lets the user "*Latch*" the effect - the same as holding their finger down at those exact X and Y coordinates - as well as adjust the slot-wide *Onset* and *Release* co-efficients, re-scale the range of the X-axis according to the 161 available indices, and reinitialize the effect with the "*REINIT*" button.

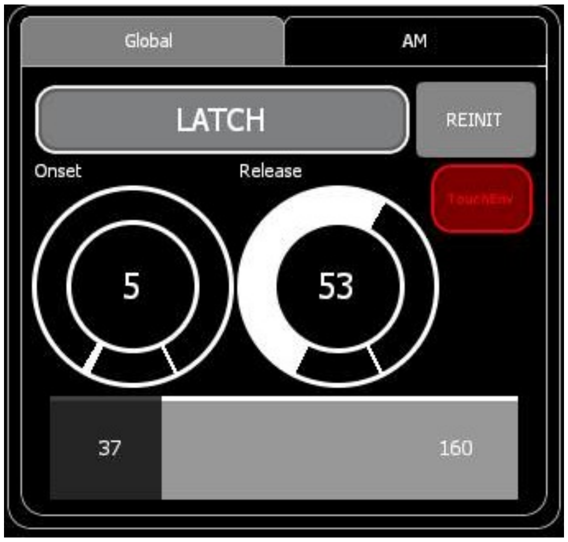


Figure 47. A typical *Global* sub-page

As of version 0.6a-rev1 of *Touchpoint*, each *Global* sub-page contains a toggle button labeled *TouchEnv*. This button links the activation of an effect directly to the activation of the amplitude envelope of the input synthesizer. In previous versions of *Touchpoint*, this was a programmed-in, system-wide setting. This feature allows users to modify their XY mappings without triggering a global audition, if, for example, they wanted to modify a sound during another slot's long release without re-opening the gate to the input to full strength.

Tapping an XY field with a second finger will reset the phase of a modulating oscillator. In the case of the *Comb* effect, a second finger tap will dump the contents of the current buffer and begin re-sampling. An application of this is using a very slow inverted unipolar sawtooth modulation oscillator to create a "dive-bomb" or "air-horn" type effect and then rapidly tapping with the second finger to repeatedly trigger the pitch shift from its beginning. One could use this same wave shape in AM mode to repeatedly trigger a sharply decaying sound, given a long enough frequency to the inverted sawtooth amplitude shape. In this capacity, the *AM* module acts like a triggerable amplitude envelope for the incoming signal. Another nice sound with this effect is to have a *Feedback-type Comb* module responding at a lower-midrange frequency with ~100% feedback ratio - so that it quickly saturates - and then dumping and re-sampling at the downbeat of a new phrase. It creates kind of a strange, saturated, tonal, distortion-heavy, sidechaining sort of effect.

3.3.5 Performative linear mixing

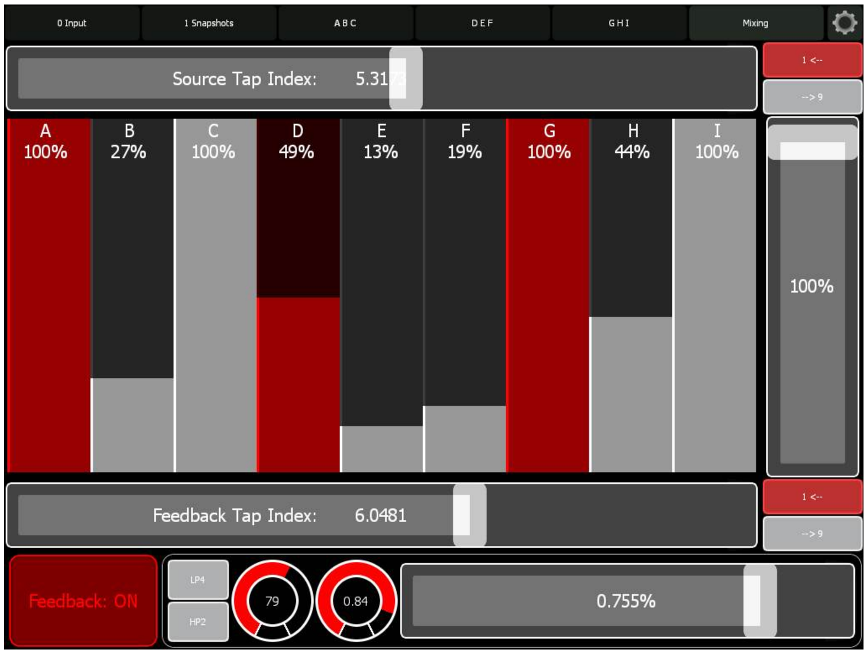


Figure 48. The performative *Mixer* of *Touchpoint*, versions 0.6 and below

For much of *Touchpoint*'s lifespan, re-serialization had not been implemented as intended. Prior to incorporation, the instrument simply worked as a permanent serial chain - without object copy and paste - which is still effective for testing and performing with many other aspects of the system. The user simply populates a chain of nine slots, flowing left to right, by a row of drag-down menus, which then displays the XY field and configuration options of that processor - some of which are global, such as the attack-release envelope of crossfading the processor's influence downstream, and some of which are processor-specific.

The performative *Mixer* (shown in Figure 48) allows for the ability to "*roll-back*" the influence of individual objects upon the resulting signal chain in either direction, i.e. crossfade in or out a per-object wet/dry control as a linear vector across the effect chain. Each object has its own signal strength (wet/dry) control in addition to this global presence fader. Finally, a toggleable, state-variable filterable feedback loop is placed at the end of the chain, with high-pass and low-pass modes and a variable feedback amount, up to 90%. This *feedback thread* also has a variable vector crossfade to choose which point of the chain to send back to the input from. Its *tap point* is independent from the *source tap point*.

This feature may be less interesting with objects that do not contain input analysis, but with the *FM* object, for instance, different per-object wet/dry levels can result in completely new pitch outcomes to the next object in the chain. The mixer becomes sort of a pitch sequencer, in this way. (The design of the *AM* and *Comb* objects may become additionally modeled after this interesting primitive analytical feature of the *FM* object in subsequent revisions of the software.)

Each active slot of the possible 9 is highlighted red, whether that be from held-down fingers or latched slots. This helps provide a helpful visual cue for which slots the player(s) mix will have an audible effect when being adjusted at that moment.

Lemur was chosen over other popular mobile device touchscreen interface prototyping platforms with similar functionality, such as *TouchOSC*, *Control* (Roberts, Wakefield, and Wright), *Beatsurfing*, *TUIO/TUIOdroid*, and *MIRA* (Tarakajian, Zicarelli, and Clayton) for several reasons:

- *Lemur*'s wireless connection support is much more stable compared to most others, whether connecting through an IAC bus application to send MIDI, or just inputting and outputting across local network IPs and ports to send OSC. In the author's experience, for whatever reason, *Lemur Daemon* has much greater stability than *TouchOSC Bridge*.
- By a similar token, connecting to a recipient mobile device to push a new interface layout, while still not perfect, works much better when using *Lemur Editor* versus *TouchOSC Editor*, or editing a layout on the device for *Control* or *Beatsurfing*. *Lemur Editor* is robust enough to allow for a streaming connection, so that in-app widget modifications and value changes are reflected instantly on the desktop editor, and widget modifications on the desktop editor will show up in realtime on the mobile device layout.
- *Lemur* allows for in-app widget modification to a layout. Cycling 74's *MIRA* and *Lemur*'s most popular competitor, *TouchOSC*, does not allow for this. *Beatsurfing* relies on in-app editing only, to the point where on-the-fly layout modification mid-performance is encouraged. This is undesirable in the case of *Touchpoint*.

- *Lemur* layouts possess global variables, arrays, internal memory, "*Container*" sub-windows, addressable object attributes, and an expansive object library, including math operations, and access to internal iOS features, such as accelerometer, internal time, and battery life. This is an incredibly important feature, as it allows for conditional behaviors and widget scripting. *Lemur* "scripting" enables the user to write function callbacks that output custom messages, change currently displayed *Container* tabs, modify widget attributes and more. This can sometimes allow for the near-perfect illusion of state-perfect synchronization with the computer software it's controlling. These features were used extensively throughout the *Touchpoint Lemur* layout. Very few of the facilities provided by the *Lemur* software were un-utilized in the *Touchpoint* layout.
- While *MIRA* actually provides many very clear advantages over its competitors, such as zero-configuration device pairing, robust gesture recognition (swipe, tap, double tap, partitioned window space, pinch, zoom, twist), and a transparent memory state which is synchronously linked to its master project, *Lemur* exists independently of host software. *MIRA* layouts have perfect synchronization with *Max/MSP* patchers, but they can only be used with *Max/MSP*, and have many object-specific special features. *Lemur* is just a really fully featured widget toolkit that outputs MIDI and OSC wirelessly.

3.4 Multiplayer functionality

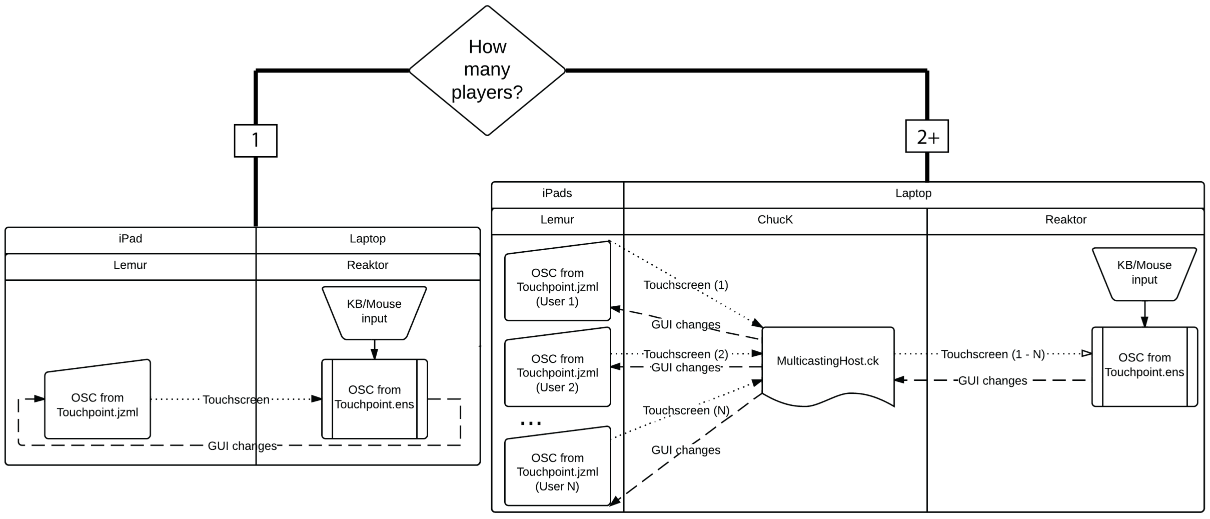
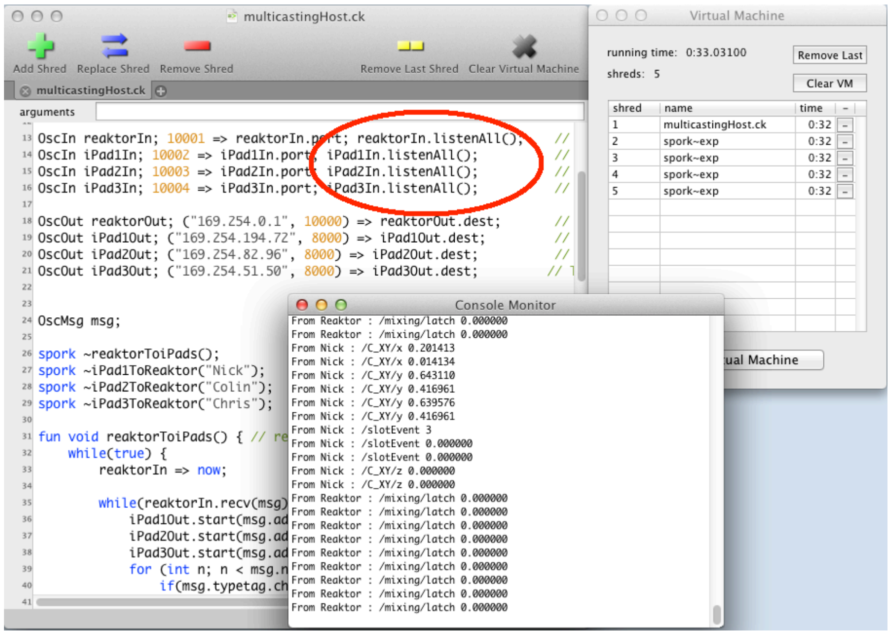


Figure 49. Single- and multiplayer software configurations of *Touchpoint*

Touchpoint can be performed by one musician, or many musicians can manipulate one session. Or, any other combination therein can be achieved: one musician could play the same control gestures into multiple *Touchpoint* sessions; networked sessions could play alongside solo or other networked sessions; so on and so forth.

If only one musician is using *Touchpoint*, two pieces of software are used across two devices. If More than one musician is using a *Touchpoint* instance, three pieces of software are used across two devices. The graphic in Figure 49 explains how this breaks down.

For the networking component, *ChucK* was chosen as the optimum middleware solution. One of *ChucK*'s many design principles is that it is optimized to be a lightweight client for low-configuration networked performances (Wang, *The Chuck Audio Programming Language. a Strongly-Timed and on-the-Fly Environ/mentality*). This, combined with the CalArts Music Technology department's relationship with *ChucK* creators Ge Wang and Perry Cook, made it the perfect augmentation software to handle multiplayer *Touchpoint* performances.



```
OscIn reaktorIn; 10001 => reaktorIn.port; reaktorIn.listenAll();    //
OscIn iPad1In; 10002 => iPad1In.port; iPad1In.listenAll();           //
OscIn iPad2In; 10003 => iPad2In.port; iPad2In.listenAll();           //
OscIn iPad3In; 10004 => iPad3In.port; iPad3In.listenAll();           //

OscOut reaktorOut; ("169.254.0.1", 10000) => reaktorOut.dest;        //
OscOut iPad1Out; ("169.254.194.72", 8000) => iPad1Out.dest;          //
OscOut iPad2Out; ("169.254.82.96", 8000) => iPad2Out.dest;           //
OscOut iPad3Out; ("169.254.51.50", 8000) => iPad3Out.dest;           // 1


OscMsg msg;

spork ~reaktorToiPads();
spork ~iPad1ToReaktor("Nick");
spork ~iPad2ToReaktor("Colin");
spork ~iPad3ToReaktor("Chris");

fun void reaktorToiPads() { // re
    while(true) {
        reaktorIn => now;

        while(reaktorIn.recv(msg)
            iPad1Out.start(msg.ad
            iPad2Out.start(msg.ad
            iPad3Out.start(msg.ad
            for (int n; n < msg.n
                if(msg.typetag.ch
```

```
From Reaktor : /mixing/latch 0.000000
From Reaktor : /mixing/latch 0.000000
From Nick : /C_XY/x 0.201413
From Nick : /C_XY/x 0.014134
From Nick : /C_XY/y 0.643110
From Nick : /C_XY/y 0.416961
From Nick : /C_XY/y 0.639576
From Nick : /C_XY/y 0.416961
From Nick : /slotEvent 3
From Nick : /slotEvent 0.000000
From Nick : /slotEvent 0.000000
From Nick : /C_XY/z 0.000000
From Nick : /C_XY/z 0.000000
From Reaktor : /mixing/latch 0.000000
From Reaktor : /mixing/latch 0.000000
From Reaktor : /mixing/latch 0.000000
From Reaktor : /mixing/latch 0.000000
From Reaktor : /mixing/latch 0.000000
From Reaktor : /mixing/latch 0.000000
From Reaktor : /mixing/latch 0.000000
From Reaktor : /mixing/latch 0.000000
From Reaktor : /mixing/latch 0.000000
```

running time: 0:33.03100
shreds: 5

shred	name	time	-
1	multicastingHost.ck	0:32	-
2	spork~exp	0:32	-
3	spork~exp	0:32	-
4	spork~exp	0:32	-
5	spork~exp	0:32	-

Figure 50. "multicastingHost.ck" running in MiniAudicle

Specifically, *Touchpoint* is using a feature of *ChucK* that is new as of the recent *ChucK* beta 1.3.2-rev3: address path-less OSC port listening is now possible thanks to the `.listenAll()` callback. Operational code written using this new event listener is shown in Figure 50.

Combined with *ChucK*'s capability to do UDP[52] multicasting, the operator of the *Touchpoint* host computer can simply configure the *ChucK* file for the relevant IP[53] addresses and ports, and fire it up file in the command line. The *ChucK* server spawns a processor thread per device to asynchronously receive control instructions from multiple tablets, forwarding them to *Reaktor*, while *Reaktor* synchronously reports GUI changes to each of the respective tablets.

3.5 An example workflow

Typically, when working with *Touchpoint*, the author has found it most constructive to think about sound design visually, such as in the graphic shown in Figure 51. Alternatively, a text narration of the thought process behind working with *Touchpoint* follows this paragraph.

The user begins a session at a typical 120BPM tempo with a single A440Hz sine wave for an input signal. Input source enveloping is completely turned off, so that the user can hear the full chain from beginning to end at all times.

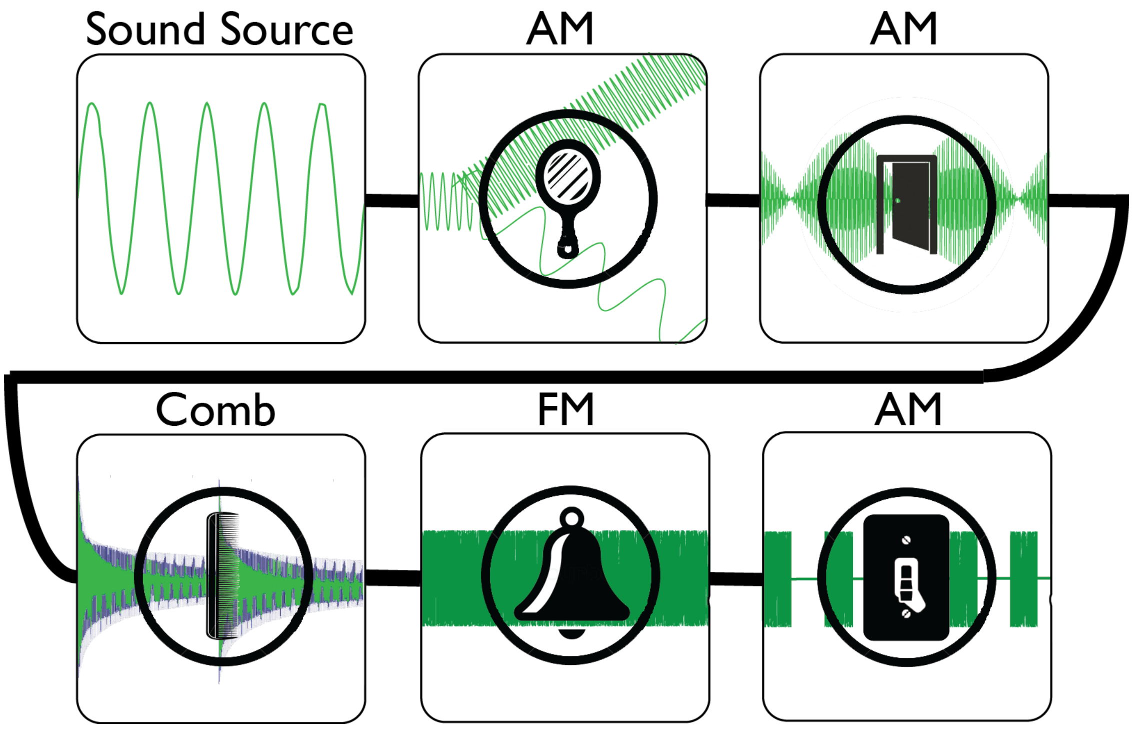


Figure 51. Graphic depicting rough visual representation of sound design in *Touchpoint*

[52] User datagram protocol.
[53] Internet protocol.

Intending to create sum-and-difference frequencies using ring modulation, the user adds an *AM* object to the empty chain, with a bipolar sinusoidal modulator shape, and scaled its range of indices from roughly 32 to 88: fast enough that they are outside of perceivable rhythmic units (sub-64th notes or so), but slow enough that they resultant frequencies will not alias and wrap back around from less than or equal to 0Hz and above 20KHz. The modulator is set to a bipolar sinusoid, and the user activates the *AM* object somewhere in the middle of the X-axis. This splits to original 440Hz tone into two midrange sine tones. They latch the X-and Y-axis settings, holding the activation envelope of this object open.

Intending to then create complex, inharmonic, clangorous, Bessel function bell-like tones based off of the two new fundamental frequencies, the user adds an *FM* object next into the chain, scaling its indices beyond 120 or so, to ensure that the rate of pitch change is at a very high audio-rate frequency. The modulator is set to a bipolar sinusoid and its total pitch shift is set to 12 semitones or more. The settings are latched.

Intending then to create a very short series of spectrally complex impulses based off of the resulting sound, the user adds another instance of the *AM* object, placing its indices from 0 to roughly 20; something like 1 bar to 32nd note triplets. The modulator is set to a unipolar pulse wave with perhaps a sub-10% pulse width. The user places their finger somewhere in the middle of the X-axis, creating rapid 16th note pulses out of their inharmonic, dense cloud of an input sound.

The user then uses those pulses as an excitation for a Karplus-Strong model by adding a *Comb* object to the chain, setting its *Type* to *Feedback*, scaling its indices to a midrange frequency (32-80) and setting the Y-axis, mapped to feedback amount from 50-100%, to somewhere in the middle. Each pulse now excites a resonant string with a relatively long decay time, greater than perhaps one second.

The user calls up another *FM* objects in order to gradually drift the pitch of these string excitations overall gradually within +1 semitones of the original pitch, setting its modulator to a unipolar sawtooth wave and latching it. Another *Comb* object is then created of *Stutter Type* and the full range of indices with a quick attack-release activation crossfade envelope and then performatively stutters the sound while also moving around the pitch of the previous Comb instance.

3.6 Conclusion

The rise of iOS and Android-based products has brought the age of ubiquitous computing to bear. Within a single form factor packed with common sensor technologies, embedded devices, personal computers, and controller interfaces can converge into one object. Nascent research created for previous iterative post-PC platforms like older mobile phones (Essl), PDAs (Tanaka), and Wacom tablets (Zbyszynski et al.) can be incorporated into an easily-developed application for an iPhone or Nexus device without abstracting its initial objects.

By subtracting the need to fabricate or otherwise augment a hardware platform from the scope of the product design, the software on the ubiquitous platform along describes the instrument. The success of well-known iPhone apps like SMule's *Ocarina* (Wang, “Designing Smule’s Iphone Ocarina”) and ZooZMobile's *ZooZBeat* (Weinberg, Beck, and Godfrey) are strong examples of this concept reaching a mainstream audience. Continuing the legacy of the original Lemur are interface-building programs like Hexler's *TouchOSC* and Charlie Roberts' *Control* (Roberts, Wakefield, and Wright), which make rapid, iterative prototype development of similar instrument apps on these devices possible.

The reason for choosing Apple's iPad as the target tablet is fairly simple and practical.

- By conforming to a very steep compatibility curve, Apple products ensure a consistent user experience and hardware profile range. Generally, if a developer makes something work on iOS, it will work on all iOS devices. While this is certainly a design goal for Google with their mobile operating system Android, their decentralized structure, separating hardware manufacturer from software developer, means that Android devices can run a whole range of different profiles. Developers sometimes have to maintain carefully annotated lists of devices that their application is not to run well or not run well at all on.

- Apple has gone to great lengths to ensure that communication between any iOS device and any modern variant (~10.7+) of their desktop operating system, OS X, is as smooth as possible. *Touchpoint* was developed on an Apple MacBook Pro laptop, and so a minimum of driver issues, file format compatibility, etc. was encountered vs. using Microsoft Windows or a Linux variant.
- iOS devices are the single most popular kind of mobile device running under a unified support platform by a vast majority. Most people you encounter will own either a modern iOS device or an Android device of some description in addition to their desktop or laptop computers, and certain crucial softwares used in development do not exist or do not run nearly as well on Android.
- The iPad 2 - which was used for the development of *Touchpoint* - is a very practical form factor for consideration as a stage instrument. It is lightweight and highly visible, it supports up to eleven independent registered touches, it packs away easily, and it has a long battery life. It is great with connecting to ad-hoc networks, and it is the oldest currently supported iOS tablet, serving as the current baseline for compatibility.

Touchpoint has been designed in such a way that it is capable of going in many different directions with just three basic objects. These objects have had as many of their common controls presented in the same way as possible, so that only the differences between them become the "*cockpit control problem*" that often plagues complicated new synthesis interfaces.

It was developed in an unconventional but high sound quality construction platform that shares many of its roots with the classic history of music synthesis development platforms by being a modular synthesizer and a visual programming environment. This modularity translates through to the design of the instrument that was built within it as well.

Despite this reliance upon the visual programming prototyping format for rapid feature iteration, the overall program structure of *Touchpoint* was organized in such a way that it can easily be related to a text-programming format, decoupling controllers and event handlers and actual audio processing.

The choice of a touchscreen interface development platform benefits at the cause of rapid prototype version iteration, in addition to being the ideal interface suited to such an instrument. The use of a secondary lightweight music synthesis language that specialized in networked messaging makes extension of the *Touchpoint* architecture easy. The design of the program already inherently suggests multiplayer extensibility, and an unobtrusive middleware client is the ideal solution, after building it directly into the program.

The technical design of *Touchpoint* has evolved as more performances have been made with it, each one placing it in a different context and presenting new unforeseen issues or refinements that could be made to make the mechanics of performance more effective. The next chapter is a chronicle of personal responses to each of the performances, and show they've shaped thinking about *Touchpoint*.

Chapter 4

Performances

Touchpoint can be used pointillistically as a conversation between multiple players, or between a *Touchpoint* player and another instrument. It can process that same instrument in realtime, or sample them and process that sample, or re-sample in real time, or jump back and forth between grabbing the sound of the instrument and "letting them through."

Touchpoint performances can evolve slowly, like a modular synthesizer performance. The musician builds a house of cards, and then slightly adjusts its facets until knocking it over and starting over again. Or, the instrument can be used in rapid fire combinations. Combinations can be quickly assembled, stored, and then slowly moved back and forth between.

The number of configurations and presentation settings that *Touchpoint* is set up to explore already, requiring only rehearsal, is extremely large. In order to find out which ones would require re-programming, adaptation, reinforcement, or other responses, I needed to get as many documented performances as possible with the instrument featured in different contexts.

The main set of concert presentations organized for this purpose was a set I called *Touchpoint Series*. Each concert aimed to present *Touchpoint* in a different capacity. These shows and others are reflected upon in the following chapter.

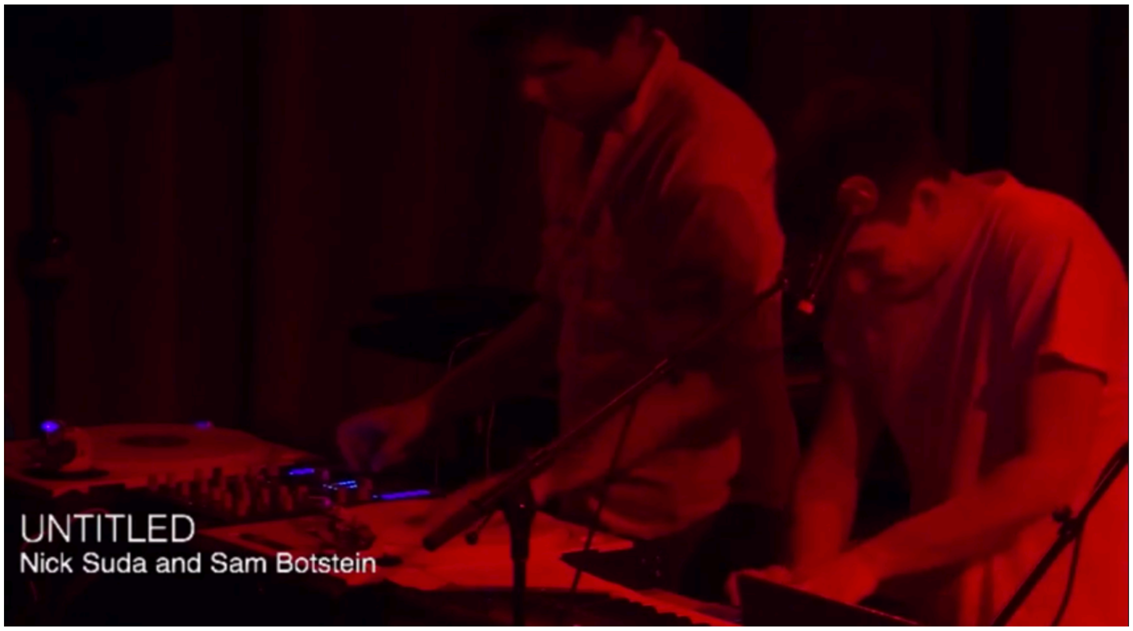


Figure 52. Sam Botstein (turntables) and the author (*Touchpoint*) in concert, 2013

4.1 December 2013 *Grids, Beats & Groups* duet; Sam Botstein on Turntables

This performance (shown in Figure 52) was scheduled as part of semesterly class performance in a course called *Grids, Beats, and Groups*, offered by Jordan Hochenbaum and Owen Vallis. The course devotes its focus entirely to what it means to perform electronic music in groups, as a duo or larger. It is the class that birthed the original Ableton *Live/TouchOSC/Reaktor* master effects chain that would eventually become *Touchpoint*, and I took it for three of my four semesters at CalArts.

This semester was different from the previous two in that I was not planning to make compositions for a specific laptop setup to be performed with a specific partner that I had rehearsed with. This would one would be a semi-composed improvisation, realized on the spot. The tools would take the focus over the material. This would simply be a conversation between myself and Sam Botstein, who played turntables for this event.

Rehearsals were very brief; we identified a few key points we had stumbled across as a result of jamming, and decided to try to string them together into a deliberate order. We would let the moments arrive with as much space between them as we felt necessary, shooting for an overall runtime of ten minutes.

The signal flow was such that his master stereo output flowed directly into my stereo input, and so we were one sound source. The choice of another electrophone that does not make acoustic noises on its own was an interesting match, because it created the possibility that one of us could cut the gain entirely on the signal path and it would be unclear who was responsible.

If Sam brought his master fader down, the intensity of my effects would also decrease. If I placed a slower-speed modulation oscillator on the incoming gain of the signal, it could interrupt scratching phrases that he was trying to communicate over other things. This kind of dynamic showed how two electrophones can merge into one perceived instrument, even beyond the merging capabilities of several instances of *Touchpoint* by itself.

The kinds of rapid gestures that are used with the turntable create some fascinating signal amplitude excitation methods. For example, *scribble* scratching was an excellent way to excite a resonant comb filter. I could somewhat emulate the sound of a *scribble* scratch by using a *Unipolar AM* object in a sinusoidal shape at an index that was at the threshold of what a scribble scratch frequency is - probably something like 10-15Hz.

However, this was my first ever public performance with *Touchpoint*, and it certainly required more rehearsals than we managed to clock in together. In fact, I learned with this performance that relying too heavily on improvisatory benchmarks - especially when they're literally notated graphically for reference - just results in a rushed performance that doesn't allow moments to exist for as long as it would be appropriate for them to say what they needed to say.

That being said, this was probably the most literal representation of two people using two different interfaces as the source and the processor, acting as operators of one system. It is an arrangement I look forward to attempting more in the future. The response from the audience was remarkable, as well - they really seemed to enjoy the musical conversation that Sam and I were having, through our eye contact and our gestures. This demonstrates that the audience doesn't care if you're using an electrophone-type instrument; performative authenticity is derived from the observation that the performers are having a conversation. They appreciate when one performer can stop talking and listen.

To view this performance, go to youtube.com/watch?v=_mZ_yvGwqB0.

4.2 Colin Honigman: *Creative Electronic Music Ensemble* member

Early into the Spring 2014 semester at CalArts, I was approached by fellow MFA student Colin Honigman. He had just joined Wadada Leo-Smith and Mark Trayle's group called the Creative Electronic Music Ensemble, and he wanted to play *Touchpoint* in it. Typically, the band is equally comprised of traditional instrumentalists and folks playing various electrophones - synthesizers, samplers, drum machines, groove boxes, etc. This semester, Colin found himself as the sole electronic "player."

As the sole performer of an electrophonic instrument, having Colin show up to rehearsals with a laptop and a limited bag of tricks would not have been sufficient to meet the highly dynamic and lucid nature of performance. This provided me with an otherwise unavailable research opportunity: I could provide someone besides myself with a "fork" of the rapidly developing software, allowing them to develop their own performance practice with it, outside of my influence.

I attended the final rehearsal of the ensemble, as well as their recital. Listening closely to Colin's playing, I was struck by some of salient differences in our basic approaches to playing *Touchpoint*:

- As the designer of the system, I tend to use the entire chain at once, making choices based on my constant awareness of the number of slots I have available in the system as a whole. Colin makes much greater use of subsets of modules, usually separated by page. Where I can spend only a minute tweaking three processors before moving up or down a page, Colin will often spend ten or fifteen minutes playing just one or two of the modules at a time.
- Colin's playing tends to be much more pointillist than mine. While this is necessitated by playing in an improvisation-based ensemble, it also serves as a much more effective test of *Touchpoint*'s viability as a note-level instrument. My performances end up feeling much closer to being a modular synthesizer session, where sound is coming from the instrument the entire time, with limited silence. Whereas Colin uses the instrument more

like a black box of wild sounds, which he is artfully probing. This encouraged me to think a lot more about finding the specific sweet spots in the two-dimensional modulation ranges, which are worth presenting in isolation, rather than just sweeping through ranges along the axes all the time. This difference is covered in Colin's own words in Section 4.3.2.1.

- Colin's playing in an ensemble demonstrated how much more effective slow movement can be when underscored by other forms of musical activity. My focus on developing the system has often led me to skimp on considering effective accompaniment for *Touchpoint*, and how that changes the music you make with it.

Early feedback from Colin led to the addition of a new interface control to the *Global* page: by toggling the new *TouchEnv* button, you can disable activation of the sound source from activating the effect slot. That is, slots can serve a passive actuation role within a chain, their changes being audible in the presence of other effects, but not initiating an audible chain in their own right.

4.3 *Touchpoint Series 1-3*, February - May 2014

The *Touchpoint Series* was booked so that I had a guaranteed series of opportunities to try out some ideas that needed long-form exploration. They were each small, informal events with a small attendance. Each performance was video documented, and reflecting on these actually taught me a lot about how *Touchpoint* performances can improve, perhaps equally with performing in the concerts themselves. What follows is a reflection of all three presentations.

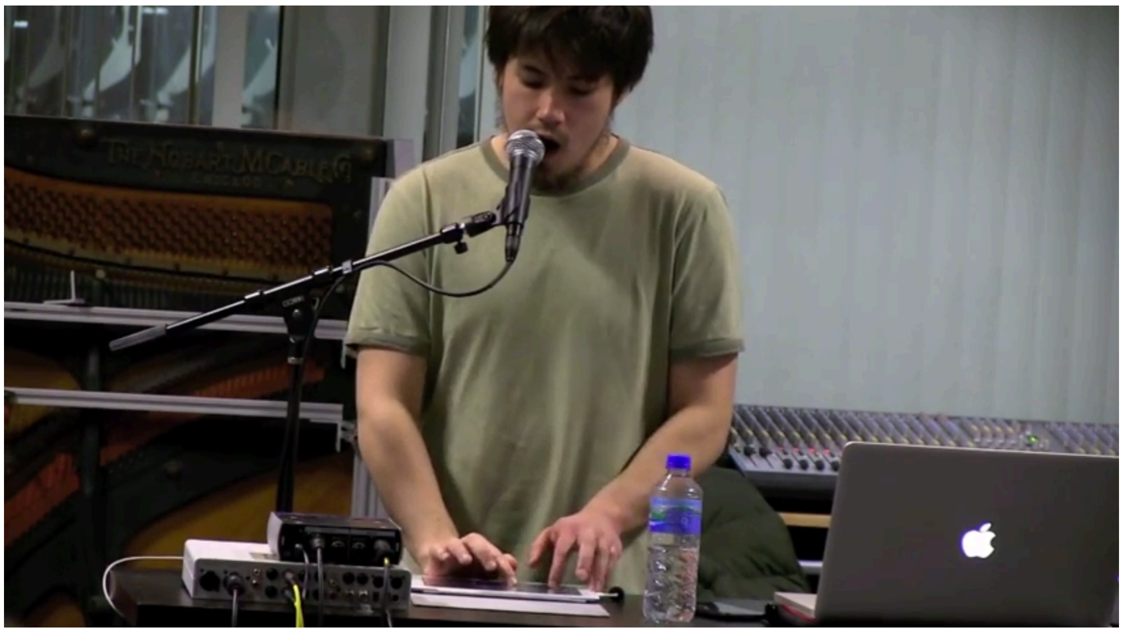


Figure 53. The author in *Touchpoint Series #1: Solo Performance*, February 2014

4.3.1 *Series #1: Solo Performance*

Series #1 (shown in Figure 53) took place on February 28th, 2014. In this concert, I played as the input source and as the *Touchpoint*-ist. I made a last-minute decision to improvise a 30 minute set using only my voice through a microphone as the input signal.

This led to an interesting mechanic of using up the first of my nine slots every time as a *Comb* effect set to *Stutter* mode, so that I could sing a note and then loop it. Turning my voice into a periodic signal usually involved singing into the mic before I free the *Stutter* buffer, so that, upon release, the system would continue to receive input. I would then "unlatch" the *Stutter* buffer and then place my finger at the upper left corner of the XY field, so that my singing looped at one bar at tempo, at full volume.

One of the ugly side effects of doing this is that the *Stutter* buffer has no zero-crossing protection. Therefore, depending on where recording started and ended, the unprocessed re-looping of the buffer sounded very obvious. Applying any effects process to it easily obscured it, but this set involved frequently starting with a "fresh" sound via an initializing Snapshot recall, and thus the audience is confronted with the potential for a "popping" loop frequently.
However, using my voice as a signal input allowed for many interesting excitation opportunities. Similarly to how Sam Botstein's *scribble* scratch really embodied the loudness of a comb filter

excitation, so too did my hissing into the microphone. More often than not, the two kinds of sounds I would sample of myself were either singing a constant note (good for spectral refraction, pitch shifting etc.) or hushing (good for gating, comb filter excitation, and so on).

One of the most musically intriguing possibilities that had arisen from this performance was that I could get a chain of low frequency *FM* units going as a melodic step sequencer, turning a single sung note into an elaborately syncopated, pseudo-step sequenced arpeggio. I could then modulate that entire sequence additionally by putting another slower *FM* behind everything else. Since each *FM* unit has a basic pitch tracker in it, by unlatching a held sung note in the first-slot *Stutter* buffer and singing a note higher or lower, I had the equivalent of an unprepared modulation at my disposal. If I had decided to sing with a glissando up or down to the next pitch, the pitch sequence would have followed my lead!

This performance used version 0.5.10a, as the *Grids, Beats, and Groups* concert had. This version still used a static chain of nine processors with the performative *Mixer*, rather than using the *The Finger*-style contextual stack management. This means that any time I wanted to drain the chain and start over, I had to essentially hit a panic button. Moment creation was predicated around building a chain like a modular synthesizer, and not about very quickly reacting to a sound I had just previously made. Upon watching the performance, I was disappointed to realize that without the "stack" mechanic, watching a *Touchpoint* performance in this state fails to achieve the basic premise of embodying effects processors in a meaningful way. I had just replaced the modular synthesizer - or the laptop - with an *iPad*. The audience was simply watching somebody prod at a screen without much attention paid to anything else.

To view this performance, go to youtube.com/watch?v=9iF8bfGumjc.

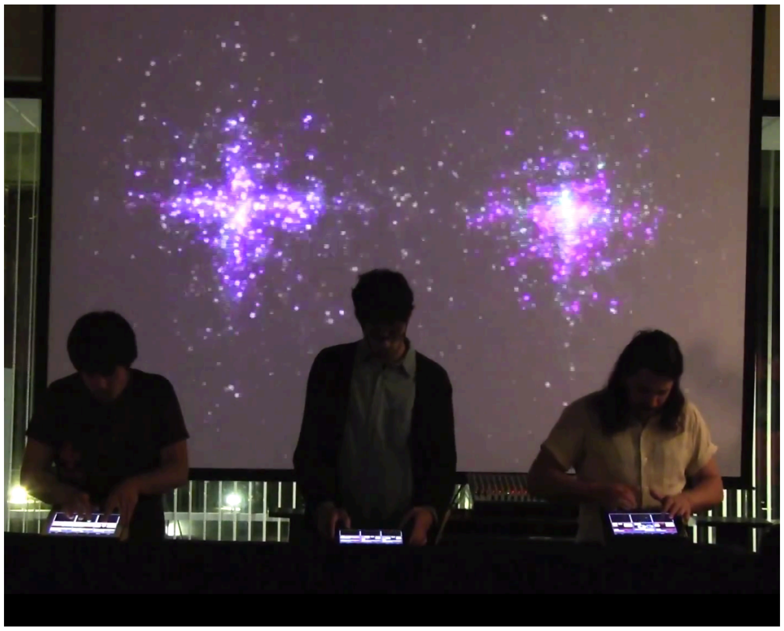


Figure 54. Suda, Knollmeyer, & Honigman, *Touchpoint Series #2*, March 2014

4.3.2 *Series #2: Multiplayer Session*

The *Series #2* concert (shown in Figure 54) used version 10.6a-rev1, which completely revised the internal structure of the instrument, revised its OSC specification, and added the *TouchEnv* functionality. This performance also necessitated the development of the *ChucK*-based middleman to send synchronous multiplayer commands to *Reaktor*, which responded in kind with global GUI updates.

In *Series #2*, I was adamant that the performance require three people of varying levels of experience with the instrument. Consequently, I got my friends Colin Honigman - who had been using it in Mark Trayle's *Creative Electronic Music Ensemble* for two months - and Chris Knollmeyer - who hadn't ever used it at all - to participate. We decided to arrange ourselves so that Colin had the first page with slots A, B, & C, Chris had the second page with slots D, E, & F, and I had the third page with slots G, H, & I. (This version of *Touchpoint* still didn't have dynamic re-serialization presently implemented.)

We decided on this order to play to each performer's style, experience level and strengths. Colin was adept at using *Touchpoint* as a primary source, rather than as an entire orchestra, so it made sense to put him at the beginning. By occupying the middle of the chain, Chris' actions would effectively be the least influential, which reduced the risk of cancelling all three of us out by engaging an effect he didn't know how to get out of. Finally, by placing myself at the end, I could act as though six slots were already running for me, and I was just playing with the whole sound its present state.

I suspected that if this performance didn't work, it would be easy to see that it could be the fault of the performer with the least experience. I consider the performance somewhat unsuccessful, but it was not for this reason. By my assessment, I believe that the particular scenario I had manufactured - "three people, each in charge of three processors, one page each" was too much. This caused as many as nine modulators to be manipulated simultaneously, which can actually overload to the point of silence from exceeding the capacity of the system's built-in saturators.

There were too many people making too many decisions about too many processes at the same time. It would be just as difficult to appreciate from the performer's point of view, as it would be the audience's. In fact, my favorite moments of this show where when I would decide to step back and completely pull out of the chain, letting the other two performers work by themselves.

There were moments when I thought we were all having a mutual moment, when I realized I had actually just looped some of their performances, being at the end of the chain. Their input during an entire two-or-three minute section was muted while I ran a one-bar loop that I modulated over them. Certainly, a successful three-or-more player performance that is possible, but such a performance should require more guidelines than the very open requirements I provided at this concert.

In May of 2014, I asked Colin and Chris to reflect on their experiences with *Touchpoint*, both during and around the time of the concert. I asked them what they found successful and unsuccessful. I asked them to compare it to similar systems that they are familiar with. I asked them what their favorite parts and least favorite parts about using it were. Their responses were valuable and insightful. What follows is a verbatim transcript of their responses.

4.3.2.1 Reflections from Colin Honigman

"*Touchpoint is a novel combination of touch interface and synthesis technique that results in a multi-faceted and expressive instrument. To begin with, it is a fairly simple interface that can be more or less understood without* *much explanation. On the surface, it is easy to begin playing, and it is obvious that many sounds can be achieved. With practice and exploration, the instrument is found to be capable of a multitude of sounds, with the added ability to manipulate those sounds on micro and macro levels.*

The choice of three sound generators for each unit allows for a surprising amount of variation, and provides continuously surprising and pleasing effects. Looping, while not an independent feature, is achievable through knowledgeable manipulation of the comb filter. In a performing context, Touchpoint allows for great control and expressivity, and is especially useful in an improvisatory context. Repeatable performances are achievable, but require practice, and the careful use of presets.

However, the exploratory approach is my personal favorite. This approach also makes Touchpoint an excellent tool for sound design and sample creation. While extended performance requires more practice, for production purposes, this instrument can be used to easily create a large amount of content with very little effort. Moving back and forth from simple to complex timbres, many exciting sonic possibilities reveal themselves, through the interaction of the modules.

Sonically, Touchpoint seems very similar to a modular synthesizer. However, the ability to traverse the timbral spectrum is unprecedented when compared to the analog method of patching cables. Although, because it is digital, it suffers and benefits from its digital behaviors. There were a few 'glitches,' in the developer's mind, that I found to be unique features that allowed for the creation and performance of different styles. For instance, tapping a second finger on the XY pad to restart the phase could cause a percussive clicking sound that allowed for precise rhythmic performance otherwise not really achievable without this 'glitch.'

While Touchpoint is similar in theory to a KAOSS pad, this is only a consequence of the XY touch interface itself. In my past experiences with it, the KAOSS pad feels like one is merely manipulating the parameters of different effects. With Touchpoint, the interaction is more expressive, and the output can be more surprising, especially as modules are locked, stacked, and manipulated simultaneously.

I learned and played this instrument while rehearsing with an Electronic Music Ensemble, an improvising electro-acoustic ensemble. I found that I had the ability to improvise a large range of dynamics and sounds that worked well with electronic and acoustic instruments alike.

The instrument intrigued the other ensemble members, who would say things like, 'I don't understand what you're doing, but I like it.' Like the acoustic instrumentalists, I found myself 'warming up,' creating a basic starting palette at the beginning of rehearsal that I would change and explore variations of throughout each piece. I was noting the combinations of parameter values, positions, gestures, and modules that seemed to work well.

With each practice (both ensemble and personal), I found that I had more control and increased ability to repeat sounds and transitions from one sound to another. There was always the element of surprise, as there are so many combinations available that even very small changes can have drastic results, especially when creating complex signal chains. Personally, I look forward to continue playing this instrument, for both performance and production purposes."

- Colin Honigman, May 2014

4.3.2.2 Reflections from Christopher Knollmeyer

"*Touchpoint is a valuable tool for performance because it offers the user immediate control of complex processes. Upon seeing the visual interface, the first elements to gain attention are the three XY fields. These are obviously very hands-on and can be used impulsively. This impulsive capacity allows for a greater sensation of spontaneity, an element missing from much live electronic music.*

The option to configure three spaces with three processors gives significant freedom to the range of playability, but is nonetheless limited to these three qualities of effect. If I were to ask for something more within the Touchpoint interface, it would be the ability to enter a specified range of each XY field for more specified results. This could lend use to more arranged moments between a group of players with one or more musician performing on Touchpoint. For example, I would like to have a delay time or modulator frequency happening close to a specific song tempo. A choice of scaling options could behoove the XY field as well."

- Christopher Knollmeyer, May 2014

To view this performance, go to youtube.com/watch?v=Nke4oA-ZGbI.

4.3.2.3 Transduction into another domain

As an additional facet of this performance, I contacted fellow Music Technology MFA student Gabriel Rey-Goodlatte and asked him to offer a particle system-based visualization program he was working on as a projected visual accompaniment to our performance. Visible in the background of Figure 54, the program was simply running on his laptop off to the side of the venue, seeded from the microphone input reading overall signal loudness and spectral content.

While this was a very primitive utilization of multimedia content, it wasn't my last consideration of crossing representations, or working with Gabriel on it. I first became familiar with Rey-Goodlatte's program during work we were simultaneously doing on a production called *Echo's Chamber*. This production made heavy use of networked signals triggering multiple events across a path through the performance space. Throughout the 20 minute experience, literal acoustical transduction, impulses from OSC, remote control from touchscreen interfaces, and feature extraction were all employed to trigger scenarios downstream. Ideas generated from this production led to the presentation in *Series #3*.

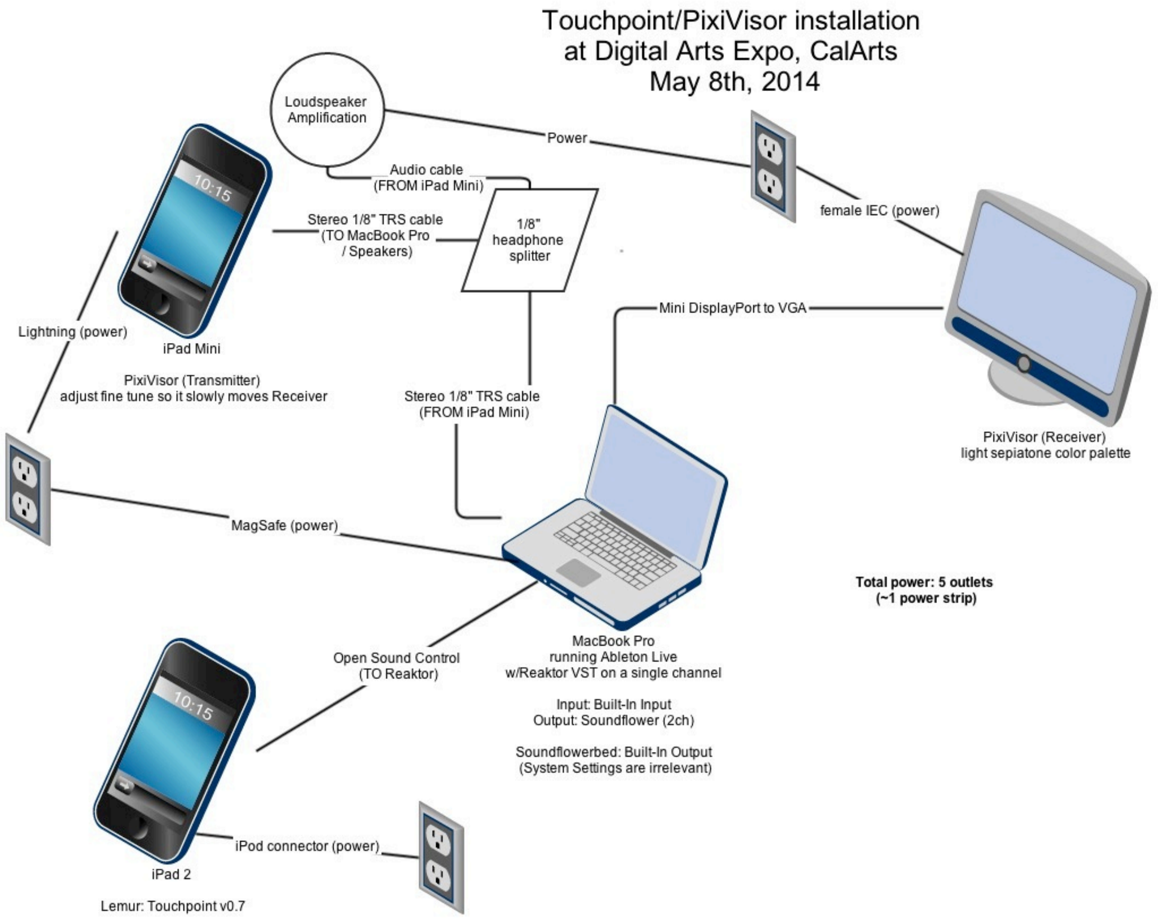

Touchpoint/PixiVisor installation at Digital Arts Expo, CalArts May 8th, 2014
Loudspeaker Amplification
Power
Audio cable (FROM iPad Mini)
Stereo 1/8" TRS cable (TO MacBook Pro / Speakers)
1/8" headphone splitter
female IEC (power)
Mini DisplayPort to VGA
Lightning (power)
iPad Mini
PixiVisor (Transmitter) adjust fine tune so it slowly moves Receiver
Stereo 1/8" TRS cable (FROM iPad Mini)
PixiVisor (Receiver) light sepiatone color palette
MagSafe (power)
Total power: 5 outlets (~1 power strip)
Open Sound Control (TO Reaktor)
MacBook Pro running Ableton Live w/Reaktor VST on a single channel
Input: Built-In Input
Output: Soundflower (2ch)
Soundflowerbed: Built-In Output (System Settings are irrelevant)
iPod connector (power)
iPad 2
Lemur: Touchpoint v0.7
10:15

Figure 55. Flowchart illustrating the setup of *Series #3: Audiovisual Installation*

4.3.3 *Series #3: Audiovisual Installation*

Series #3 was the most divergent presentation of the set, in many capacities. Firstly, it took place as part of a larger event, that being the 2014 Digital Arts Expo at CalArts. Secondly, it was not a musical performance, but rather a single-person, interactive installation. The technical setup requirements of this installation is detailed in Figure 55. Thirdly, it was not in CalArts Machine Lab, but in the lobby of the Roy O. Disney Concert Hall. Finally, the *Touchpoint* premise of contextually serialized, non-linear processor-based audio synthesis was not the core conceit of the show: rather, *Series #3* was more of a response. It was a reaction to the possibilities of this system in a greater context with the systems of others, given its newfound implications.

As a visual thinker, I was looking for an interesting way to frame the sonic results of *Touchpoint* literally and instantaneously as a visual. Gabriel Rey-Goodlatte, who was a curator in the Digital Arts Expo, recommended a program called *PixiVisor* for me to check out. *PixiVisor* is a playful "porting" of the principles behind how scanlines work on a television screen. A massively cross-platform application, *PixiVisor* can run on Windows, OS X and Linux PCs, as well as smartphone operating systems like iOS, Android, and even more uncommon ones like Windows Phone, Symbian, BlackBerry, and even PalmOS.

In *PixiVisor*, two devices run the application. One device is set to *transmitter* mode and the other to *receiver* mode. The device that is "transmitting" can play back a selection of low-resolution, simple, monochromatic animations, typically comprised of maybe under 30 frames. Each frame is then played out of the device's audio output as a rather grating, short burst of upper-midrange frequencies; a rather annoying "beep" which happens in rapid succession.

The content of the *transmitter* is sent to the *receiver* literally by piping it in as an audio connection. Typically, the headphone output of the *transmitter* is connected to the microphone input of the *receiver*. Although, one could literally go from speaker to microphone, introducing signal loss and noise from the air in the reception process.

The developer designed this system as an exploration in this aforementioned context, but the broader implication is that low-resolution analog video signal has now been transduced into another domain, availing it to all of the myriad of audio processors available, in service of creatively exploring its transformation. Therefore it becomes an obvious inquiry to see what happens when *Touchpoint* is placed between the *transmitter* and the *receiver*.

The results (demonstrated in Figure 56) were both surprisingly codifiable, as well as an effective visualization of the rapid changes that can occur during dynamic processing using the instrument.

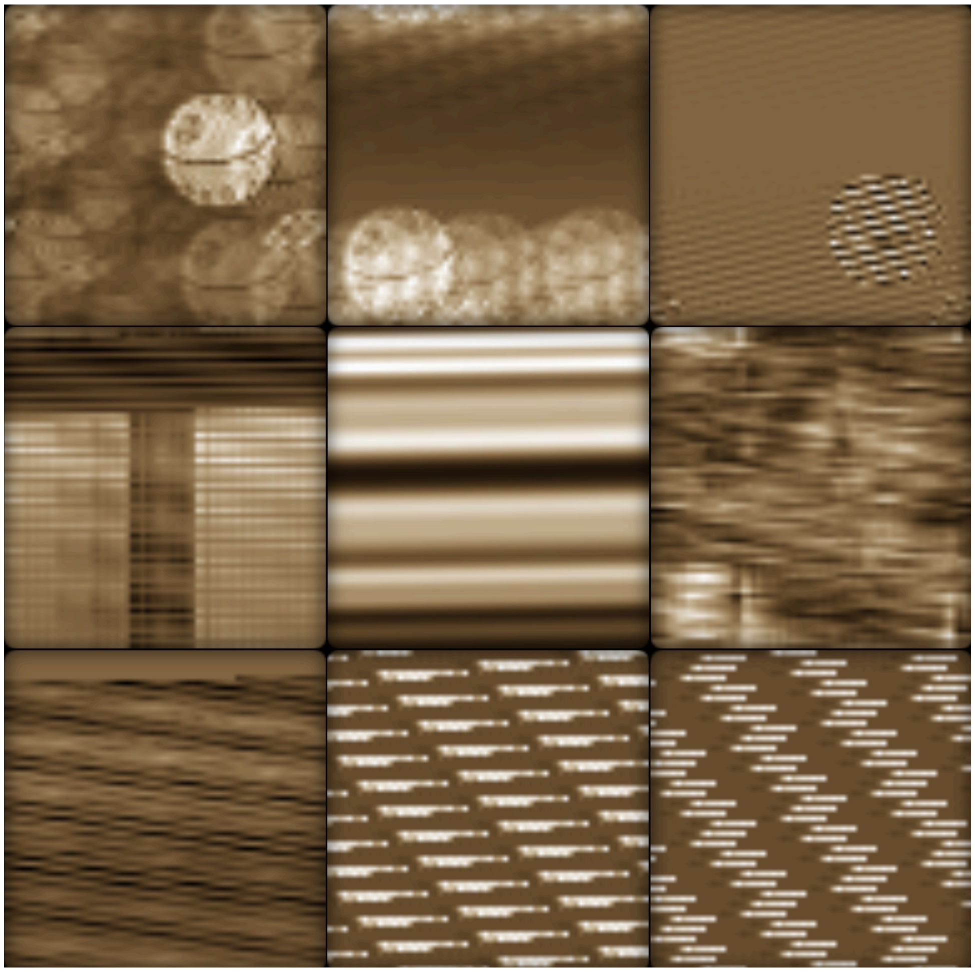


Figure 56. Selection of various results from *Series #3: Audiovisual Installation*

For example, the *Comb* module in *Feedback* mode tended to create a *ghosting* effect. If looking at a bouncing ball, feeding a high-strength, low-frequency delay line into the system would generate many faint, offset instances of the bouncing ball which cascade over time. Yet, a high-frequency line would tend to create very tightly-knit copied which are more vertically aligned with each other.

The *Comb* module in *Stutter* mode would create very striking skipping sub-frames of the image which were also looped and offset. These had the quality of looking like "sub-scanlines" of the existing sample frequency. They resemble the poor reception artifacts of trying to view an old cable television channel you don't have.

The *AM* module tends to create *bands* of solid-color "interference" super imposed over the top of the image. This makes sense, given the generation of additional pure frequencies that move relative to the input pitch, and which are synthesized from the root pitches of the input frequencies. Sweeping up on the amplitude modulation frequency tends to rotate these *bands* in a clockwise direction, as they simultaneously shrink.

Polarity of the modulating oscillator also effected the image. If the oscillator was unipolar, the *bands* tended to be a solid white tone. However, if the oscillator was bipolar, these interference *bands* alternated between being solid white and solid black.

The *FM* module created effects that were similar but different to the *AM* module, in ways that are difficult to articulate. It tended to garble the *shape* of the depicted objects in a way that, at slow frequencies, made it wiggle. At high frequencies, the image would take on more of a noisy *embossed* impression. Polarity is represented in linear motion away from the original center point, with unipolar modulation drifting from center to right and bipolar modulation drifting both the left and the right.

While the novelty behind these results is heavily impinged upon the author of *PixiVisor*'s imaging codec, the fact that each processor introduced an effect which was parametrically different from each other and predictable in an intelligible way with the audio effect infers that my choice of processors based on the modulated sonic parameter holds some weight. It reinforces the idea that these choices were not arbitrary; they are grounded in principle.

Images with simple linear motion were most effective in this setup. Two of my favorites were a loop of the Death Star bouncing as though it were a basketball, and a square which simply expands out from the center of the picture. More sophisticated images, such as a loop of a galloping horse, or anything in which there is activity within a scene, tended to look more washed out and noisy beyond recognizability. Visual effects visually crossfaded out from the original image following the same amplitude envelope as the audio effects.

4.3.3.1 openFrameworks interface: Simplifying Touchpoint

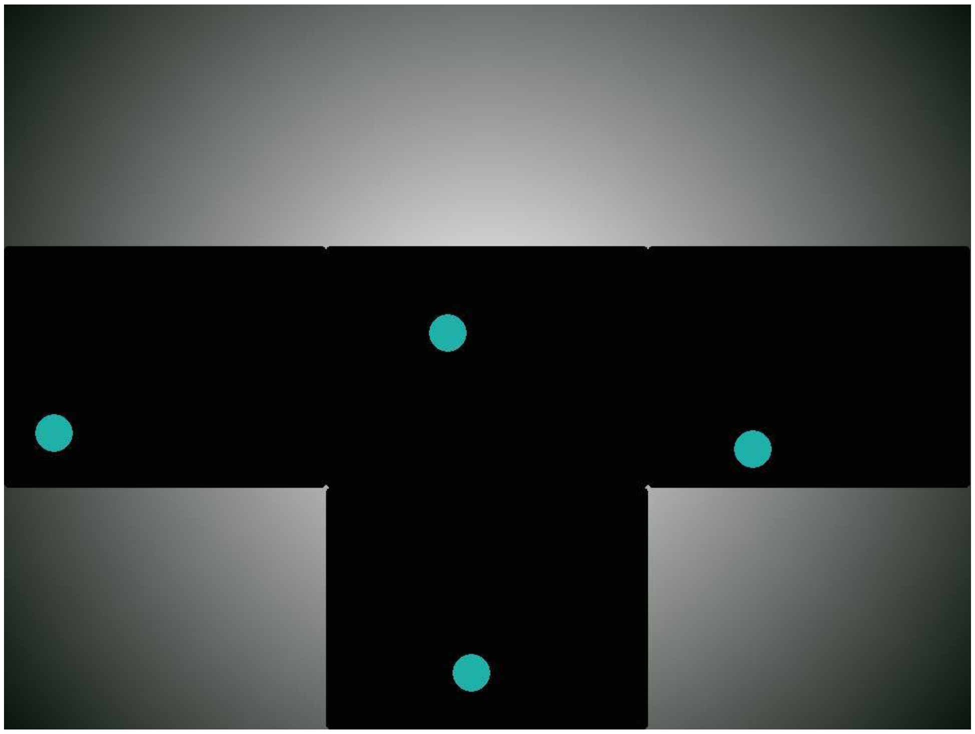


Figure 57. A simplified *iPad* interface done in *openFrameworks* for *Series #3*

One of the challenges in adapting *Touchpoint* to an installation experience was knowing that I couldn't possibly present the existing Lemur interface, with all of its "cockpit control" problems, to anybody who walked up to it.

I needed to strip away the interface experience to its absolute core. I felt that this core experience is the idea of traversing the XY fields throughout the whole system, which, as it turns out, is not exactly something that was possible to implement in the Lemur interface version anyway.

So, I stripped away the selection of modules and their settings by creating several *Snapshots* which rotated every few minutes by an AppleScript running on the host computer. I also ventured into my first major use of C++ by creating an entirely new, super minimalist interface in *openFrameworks*. A rendering of this version, with four fingers active in four different modules, is shown in Figure 57.

For the first time, this version decoupled the player's fingers from the active processors: You could now put one finger down and drag it across the screen, triggering the *Onset* and *Release* envelopes of each module as you entered or left it. You could also place two fingers down on the same module, and each movement from each point would trigger a new message, creating very wild, jittery, glitchy results. Additionally, the foundation has now been built in the most bare bones sense for a completely customized UI for the fully-featured version of the instrument.

Overall, I feel this project was a fascinating examination of disembodying the control signals, the transformative results, and the representation of using *Touchpoint* as separate objects. It has inspired me to further pursue the metaphorical qualities of *Touchpoint* as a source for data mapping and abstract, parametric manipulation.

4.4 California Electronic Music Exchange Concert series @ UCSD, April 2014

In early 2014, I was invited to come to the University of California at San Diego to play in the annual California Electronic Music Exchange Concert, or CEMEC. CEMEC is a mini-tour of sorts that involves the work of composers from CalArts, Mills College, UC Santa Barbara, UC San Diego, and sometimes, Stanford.

While I knew that my performance (which opened the show) would be different from the others in the sense that it was an improvisation instead of a composition, I came away from the performance feeling very strongly that I needed to put more work into what I had done. Seeing so many talented musicians perform with very non-traditional interfaces in such a way that had a really captivating form and a great sense of space left me feeling very humbled. I need a lot of work as a composer. I feel that in the process of becoming a digital luthier, I have lost a sense of composition, instead turning every showing opportunity into a showroom floor blast of going through the motions.

I left the concert feeling a new-found sense of reflection. I really need to write some actual music with the new instrument, instead of spending all of my time ironing out the technical details only.

4.5 Conclusion

The performances conducted with *Touchpoint* throughout 2013 and 2014 were all carried out before it had reached the desired specification for version 1.0. While I was worried this would lead to inadequately prepared material, in fact it greatly influenced the instrument's development in very constructive ways. Colin Honigman's pointillistic use of the instrument as a note-level modular synthesizer led me to adding the XY-field activated input envelope for the *Series #2: Multiplayer Session* performance, under the *TouchEnv* button.

Identifying that two separate implementations of the *Touchpoint* concept have developed within the building process has allowed me to refine each one to better serve their typical presentation contexts. One system is better fit to solo concerts with a static input source, where configurations akin to modular *patches* can be built and more slowly manipulated over time. The use of a dynamic stack system is best suited to quick, rapid stabs, perfect for spontaneous conversations with another performer on another instrument.

Simplifying the *Touchpoint* interface for *Series #3: Audiovisual Installation* so that anyone can use it re-emphasized the value of arranging each of the processor across a two-dimensional plane, and using your fingers as probes into points across that space, and what happens when you move from one point to another.

Treating *Touchpoint* to these diverse performance contexts has shown to me that it is possible to do too much with the system. This eventually became a point of fatigue with the CEMEC concert. Each processor in its own right could be used for an entire etude, and while dynamically re-combining them leads to very rich results, it becomes clear that within reason there is a certain maximum ceiling of processors that can be used before the signal is lost altogether (best demonstrated with my perceived failings of *Series #2*). This may be a failure point of effects-based synthesis, or of its specific implementation in *Touchpoint*. Further testing of the principles of *Non-Linear Systems* will be conducted in order to better determine this.

Chapter 5
Conclusion

This thesis has attempted to justify the consideration of effects processing as a mode of synthesis and as something that should be adapted into a musical instrument in its own right. Historical context was provided in the introduction by explaining the importance of effects processors in contemporary popular music, due to the advent of recorded music, which objectified moments of sound and framed them as the definition of an entire sound or style.

It was argued that one of the byproducts of computerization was virtualization of all existing musical sound generators, which greatly accelerated the use of effects processors, and therefore made crucial their use in defining the sound of contemporary popular music. However, public perception of the "ingredients" of popular music have mostly not evolved beyond the conception of an ensemble of modern band instruments, and therefore a large cognitive disconnect is established when none of the sounds featured in the music resemble acoustic instruments. Traditionally, processes that cannot be pantomimed by an on-stage performer have been relegated to the keyboardist, the off-stage tech, or the front-of-house engineer. It is then argued that the time has come for these techniques to be embodied by a new on-stage band member.

It was also identified that although computerization is the force that brought the current confusing climate of concert performance to the state that it is in, computerization is the natural pathway out. Computer music tools have a long history of encapsulating ecosystems of smaller processes into a new, closed form factor that is to be treated as its own device. This concept befits the idea that an environment of diverse effects processors is what could define the processor as an instrument, if each processor were treated as an available "notes" within that instrument.

The history of modular functionality within computer music tools was then discussed. The growing encapsulation of object ontology within these paradigms is identified. The fifty year-old definition of what a *Unit Generator* is - via the *opcode* - tends to be much more atomically finite than modern definitions, which can include entire classes of callback functions with robust input and output definitions, or indeed even comprehensive embedded hardware and software platforms as they are combined with other platforms. Being that the objects of modularity within digital music synthesis environments are so high-level and cross-platform in the current client, using a generic construction platform for modular instrument design is logical and well suited for the task of adapting into an instrument, versus constructing everything from a much more atomically finite level from the ground up in a text environment.

It was observed how the modularity of musical devices had also influenced hardware peripherals. Where once there was only the piano keyboard, there is now an entire industry of abstracted "Controllerist" devices. It is a matter of historical momentum that eventually the customizable touchscreen interface was developed, just a few years before a completely new form factor of personal computer - the multitouch device - came about. New mobile devices created a totally sandboxed platform for sensor-driven multi-touch applications to be run on a large number of devices, creating a new culture of music synthesis applications for mobile devices.

By focusing on the effects processor environment as new kind of instrument to be built within a modular visual programming environment, careful consideration must be given to making this system as flexible as possible, in terms of signal processing. In considering the benefits of sticking only to *Non-Linear* processor models, we arrive at a whole new way of thinking about effects processors as a creative process. This facet of synth design is normally treated as a decorative function to be appended at the end of a signal chain in permanent serial routing. In fact, placing these as central to the synthesis concept and allowing their order to be performatively declared treats them as though the ecosystem of objects responds in the way a musical instrument does.

Based on revelations related to the way a buffer can be presented as a loop or as a chromatic frequency, it was identified that this principle can be applied to several different parametric categories of signal processing, which liberates their application greatly outside of previously established processors. The idea of the "perceptual modulation spectrum" is relatable to the electromagnetic spectrum, bridging many phenomena that are typically described as separate as, in reality, parametrically unified.

An overview of each construction platform was presented so that the specific benefits that distinguish each platform can be considered in the context of building *Touchpoint*. Then, performances with the instrument were discussed. Performing an effects processor in isolation is a novel concept that immediately presented opportunities for refinement. By keeping the signal input and control networking structure as flexible as possible, in which it can be a source of sound unto itself, or a processor to an input sound, or in which it can move between these roles within the same performance, *Touchpoint* can be readily presented in the many different contexts that a post-signal, "secondary" form of instrument would inspire.

The concept of a unified signal processing system - which encapsulates a certain framework of digital unit modularity, in order to become a single unit of live performance - is certainly not novel. Indeed, many facets of *Touchpoint*'s overall construction have been created before in other contexts:

- A software embodiment of the live-looping pedal, executed in *Reaktor*, which leverages the user interface and much greater storage capacity of the personal computer to make a much broader auto-accompaniment and overdubbing ecosystem (Aldrey),
- Similar software plug-in projects which also piggyback off of the features of the DAW, while providing their own capacity for modular sandboxing with variable "tap points" in both the *source path* and the *feedback path* (Gibson and Polfreman),
- Encapsulations of high-level, sophisticated signal processors and routing matrices, for performance within a *line mixer* metaphor, for a laptop and audio interface (Kleinsasser),

However, the placing of the simplified modular tool set of *Touchpoint* within the context of a visible, *personally sized* touchscreen interface appears to be a novel fusion. By equally inheriting the aesthetics of the KORG *KAOSS Pad* and Native Instruments' *The Finger* with an instrument-sized footprint, *Touchpoint* attempts to stake out a reasonable middle ground between the "magic" mystery of agency (Schloss) that often makes computer music performance problematic in presentation, and the tasteful performance of an instrument that requires technique and discretion.

By presenting each processor as a set of control widgets on a touchscreen, they are objectified with just enough physical presence that they can be understood as individually important agents within a greater environment. However, by existing as virtual instances, their routing becomes the kinesthetic property of examination, and the powers of non-linear processor re-arrangement as a mechanic of performance can be simplified to instrument-type gestures.

5.1 Primary Contributions

There are several algorithms within differently scoped levels of the architecture of *Touchpoint* which are novel, either from the perspective of having been executed in a visual programming environment, as well as what the sum of the parts does for creating an overall instrument, which interacts with the musician in such a way that its outcomes are both satisfying and predictable.

- The appropriation of the dynamically-instantiated and destructible audio effect stack from Native Instruments' *The Finger*, which led to the conceptualization of "*dynamic non-linear processor serialization*",
- The appropriation of the ranging and precision mechanics of buffer sizing from iZotope's *Stutter Edit*, which led to the conceptualization of "*the perceptual modulation spectrum*",
- The appropriation of the zero crossing counter-based pitch tracking algorithm from Native Instruments' *The Mouth*, which enabled the application of legato pitch shifting and FM synthesis upon realtime, monophonic, spectrally simple audio input, and several other features.

5.2 Final Thoughts

As evidenced by the current deluge of performative effects processor plug-ins, the era of the flexible digital multi-effects instrument is nascent and active. As evidenced by the current deluge of new *painterly interfaces* and interesting ports of existing synthesis control schema on mobile devices, the opportunity to present the mobile device as a new vessel for an instrument is heavily being acted upon.

Given the limited nature of working with a plug-in inside of a host, interface-wise, this format on its own is somewhat held back in the goal of the performative effects processor. Given the abundance of examples of both powerful custom controller mapping programs and fascinating explorations of the *painterly interface* on mobile devices, this area seems ripe for exploration of a new concept of synthesis which is accelerated by the touchscreen format.

Therefore, the definition of an effects processor instrument that combines the advantages of working with a DAW plug-in with the advantages of using a big, friendly *painterly interface* seems an appropriate match.

Touchpoint is ambiguous in many ways, and its true application potential could only be finitely defined by greater use from more people. It is not the perfect manifestation of processes that have only recently become *real-time*, but it attempts to iterate from the work being done in both the *plug-in* and *app* worlds to begin to think about how to embody these abstract sorts of goals.

5.3 Future Work

There is still much work left to do in *Touchpoint*'s second major iteration that will define its UX. As alluded to by Chris Knollmeyer in Section 4.3.2.2, *Range Scaling* should have an additional *Appearance Scaling* characteristic that enables values to be placed along an axis in a logarithmic or some other non-linear fashion. Index quantization should behave in a kinesthetically satisfying manner, such as the Haken *Continuum*[54]-like methods that were adopted in Moog for their key input correction mechanic in their *iOS* app *Animoog*[55].

Currently, the user reconfigures a single 9-slot serial chain specified by configuration menus, rather than as some sort of gestural performance command for effect selection, instantiation and

[54] http://www.hakenaudio.com/Continuum
[55] http://www.moogmusic.com/products/apps/animoog-0

removal. The intended form factor of the final application is to become a self-contained *iOS* application, with a mature, icon-rich interface, an extended touchscreen gesture vocabulary, and an embedded audio engine. For the second iteration of *Touchpoint*, the author is currently experimenting with *creative coding* libraries such as *openFrameworks*[56], *Cinder*[57] and *JUCE*[58] for a diminutive audio engine back-end mock-up.

Blocks of object chains should be groupable and re-routable. Polyphony should be extended out as far as computationally reasonable. After somehow unifying with the *dynamic stack* mechanic, performative mixing should extend to parallel chains of serial chains, instead of just one serial chain. There should be separate interface windows for *turntable-style* resynchronization of the phase of all active modulating oscillators, similar to programs like *Sinkapater* (Harriman). The ability to create and edit the use of custom breakpoint oscillator waveform shapes, created by tapping and dragging, is planned for its own menu window.

---

[56] http://www.openframeworks.cc
[57] http://libcinder.org/
[58] http://www.juce.com

Bibliography

Aldrey, Leonardo. *A Real Time Performance System for Interactively Leyered Audio Sequences*. Skolan för datavetenskap och kommunikation, Kungliga Tekniska högskolan. Print.

Attias, Bernardo Alexander, and Tobias C. van Veen. "Off the Record: Turntable and Controllerism in the 21st Century (Part 1)." (2011): n. pag. *scholarworks.csun.edu*. Web. 26 Mar. 2014.

Boulanger, Richard et al. "Conducting the MIDI Orchestra, Part 1: Interviews with Max Mathews, Barry Vercoe, and Roger Dannenberg." *Computer Music Journal* 14.2 (1990): 34–46. Print.

Carlson, Chris, and Ge Wang. "Borderlands: An Audiovisual Interface for Granular Synthesis." *Proceedings of the 2012 International Conference on New Interfaces for Musical Expression* (2012): n. pag. Web. 18 Sept. 2013.

Cascone, Kim. "The Aesthetics of failure:'Post-Digital' Tendencies in Contemporary Computer Music." *Computer Music Journal* 24.4 (2000): 12–18. Print.

Chowning, John M. "Digital Sound Synthesis, Acoustics and Perception: A Rich Intersection." *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)., Verona, Italy*. N. p., 2000. Web. 31 Jan. 2014.

Cook, Perry. "Principles for Designing Computer Music Controllers." *Proceedings of the 2001 Conference on New Interfaces for Musical Expression*. N. p., 2001. 1–4. Print.

Crevoisier, Alain et al. “Sound Rose: Creating Music and Images with a Touch Table.” *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression*. IRCAM - Centre Pompidou, France: N. p., 2006.

De Moraes, Lisa. “Ashlee Simpson and That Lip-Syncing Feeling.” *The Washington Post* 2004 : C01. Print.

Di Nunzio, Alex. “Genesi, sviluppo e diffusione del software ‘Music N’ nella storia della composizione informatica.” Università degli Studi di Bologna, 2010. Print.

Essl, Georg. “Speeddial: Rapid and on-the-Fly Mapping of Mobile Phone Instruments.” *New Interfaces for Musical Expression*. N. p., 2009. 270–273. Web. 27 Mar. 2014.

Freed, Adrian. “Novel and Forgotten Current-Steering Techniques for Resistive Multitouch, Duotouch, and Polytouch Position Sensing with Pressure.” *New Interfaces for Musical Expression*. N. p., 2009. 11. *Google Scholar*. Web. 7 Apr. 2014.

Gaye, Lalya et al. “Mobile Music Technology: Report on an Emerging Community.” *Proceedings of the 2006 Conference on New Interfaces for Musical Expression*. IRCAM—Centre Pompidou, 2006. 22–25. *Google Scholar*. Web. 7 Apr. 2014.

Gibson, Darrell, and Richard Polfreman. “An Architecture for Creating Hosting Plug-Ins for Use in Digital Audio Workstations.” *Proceedings of the International Computer Music Conference 2011, University of Huddersfield, UK*. University of Huddersfield, UK: N. p., 2011. 507–510.

Gunderson, Philip A. “Danger Mouse’s Grey Album, Mash-Ups, and the Age of Composition.” *Postmodern culture* 15.1 (2004): n. pag. Web. 4 Feb. 2014.

Haas, Helmut. “Über den Einfluss eines Einfachechos auf die Hörsamkeit von Sprache.” University of Gottingen, 1949. Print.

Harriman, Jiffer. “Sinkapater - An Untethered Beat Sequencer.” *2012 Proceedings of the International Conference on New Interfaces for Musical Expression*. University of Michigan: N. p., 2012.

Hearst, Andrew. "Technology Puts the Recording Studio on a Hard Drive." *New York Times* 11 Nov. 1999. Web. 18 Feb. 2014.

Hochenbaum, Jordan, and Owen Vallis. "Bricktable: A Musical Tangible Multi-Touch Interface." *Proceedings of Berlin Open Conference '09. Berlin, Germany*. N. p., 2009. Print.

Jorda, Sergi. "Digital Lutherie: Crafting Musical Computers for New Musics Performance and Improvisation." *PhD diss., Universitat Pompeu Fabra, Departament de Tecnologia* (2005): n. pag. Print.

---. "The Reactable*." *Proceedings of the International Computer Music Conference (ICMC 2005), Barcelona, Spain*. N. p., 2005. 579–582. Web. 2 Mar. 2014.

Jordà, Sergi et al. "The reacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces." *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*. ACM, 2007. 139–146. Web. 2 Mar. 2014.

Jordà, Sergi, and Marcos Alonso. "Mary Had a Little scoreTable* or the reacTable* Goes Melodic." *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression*. IRCAM—Centre Pompidou, 2006. 208–211. Web. 2 Mar. 2014.

Kaltenbrunner, M. et al. "The Reactable*: A Collaborative Musical Instrument." *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE'06. 15th IEEE International Workshops on*. IEEE, 2006. 406–411. Web. 2 Mar. 2014.

Kaltenbrunner, Martin, Günter Geiger, and Sergi Jordà. "Dynamic Patches for Live Musical Performance." *Proceedings of the 2004 Conference on New Interfaces for Musical Expression*. National University of Singapore, 2004. 19–22. Web. 27 Mar. 2014.

Karplus, Kevin, and Alex Strong. "Digital Synthesis of Plucked-String and Drum Timbres." *Computer Music Journal* 7.2 (1983): 43–55. Print.

Kleinsasser, William. "Dsp. Rack: Laptop-Based Modular, Programmable Digital Signal Processing and Mixing for Live Performance." *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*. National University of Singapore, 2003. 213–215. Web. 1 Apr. 2014.

Levin, Golan. "Painterly Interfaces for Audiovisual Performance." Massachusetts Institute of Technology, 2000. Web. 23 Jan. 2014.

Mathews, Max V. *RTSKED, a Scheduled Performance Language for the Crumar General Development System*. Ann Arbor, MI: MPublishing, University of Michigan Library, 1981. Print.

McNamee, David. "Hey, What's That Sound: Kaoss Pad." *The Guardian* 9 Mar. 2011. *The Guardian*. Web. 25 Mar. 2014.

Moore, Gordon E. "Cramming More Components onto Integrated Circuits." *Electronics* (1965): 114–117. Print.

Nickerson, Jeffrey V. "Visual Programming." New York University, 1994.

Nyquist, Harry. "Certain Topics in Telegraph Transmission Theory." *American Institute of Electrical Engineers, Transactions of the* 47.2 (1928): 617–644. Print.

Park, Tae Hong. "An Interview with Max Mathews." *Computer Music Journal* 33.3 (2009): 9–22. Print.

Partridge, Grant, Pourang Irani, and Gordon Fitzell. "Let Loose with WallBalls, a Collaborative Tabletop Instrument for Tomorrow." *Proceedings of the 2009 International Conference on New Interfaces for Musical Expression*. N. p., 2009. 78–81.

Puckette, Miller. *Interprocess Communication and Timing in Real-Time Computer Music Performance*. Ann Arbor, MI: MPublishing, University of Michigan Library, 1986. Print.

---. "Max at Seventeen." *Computer Music Journal* 26.4 (2002): 31–43. Print.

---. "The Patcher." *Proceedings of the International Computer Music Conference* (1988): 420–429. Print.

Roads, Curtis. *Microsound*. MIT press, 2004. Web. 26 Mar. 2014.

Roads, Curtis, and Max Mathews. "Interview with Max Mathews." *Computer Music Journal* 4.4 (1980): 15–22. Print.

Roberts, Charles, Graham Wakefield, and Matthew Wright. "Mobile Controls On-The-Fly: An Abstraction for Distributed NIMEs." *Proc. NIME*. Vol. 2012. N. p., 2012. Print.

Roma, Gerard, and Anna Xambó. "A Tabletop Waveform Editor for Live Performance." *Proc. of New Interfaces for Music Expression* (2008): n. pag. Web. 2 Mar. 2014.

Rosenbaum, Eric, and Jay Silver. "Singing Fingers: Fingerpainting with Sound." *Proceedings of the 9th International Conference on Interaction Design and Children*. ACM, 2010. 308–310. Web. 27 Mar. 2014.

Sanneh, Kelefa. "The Rap Against Rockism." *The New York Times* 31 Oct. 2004. *NYTimes.com*. Web. 25 Mar. 2014.

Schlei, Kevin. "TC-11: A Programmable Multi-Touch Synthesizer for the iPad." *Proceedings of the 2012 International Conference on New Interfaces for Musical Expression* (2012): n. pag. Web. 18 Sept. 2013.

Schloss, W. Andrew. "Using Contemporary Technology in Live Performance: The Dilemma of the Performer." *Journal of New Music Research* 32.3 (2003): 239–242. Print.

Smith, Steven W. "The Scientist and Engineer's Guide to Digital Signal Processing." (1997): n. pag. *Google Scholar*. Web. 8 Apr. 2014.

Tanaka, Atau. "Mobile Music Making." *Proceedings of the 2004 Conference on New Interfaces for Musical Expression*. National University of Singapore, 2004. 154–156. Web. 27 Mar. 2014.

Tarakajian, Sam, David Zicarelli, and Joshua Kit Clayton. "Mira: Liveness in iPad Controllers for Max/MSP." *Proceedings of the 2013 International Conference on New Interfaces for Musical Expression* n. pag. Print.

Trayle, Mark. Regarding Touchpoint v0.5.10a. 20 Feb. 2014.

Verplank, Bill, Max Mathews, and Rob Shaw. “Scanned Synthesis.” *The Journal of the Acoustical Society of America* 109 (2001): 2400. Print.

Von Hornbostel, Erich M., and Curt Sachs. “Systematik der Musikinstrumente.” *Zeitschrift für Ethnologie* 46.1 (1914): 553–590. Print.

Waisvisz, Michel. “The Hands: A Set of Remote MIDI-Controllers.” (1985): n. pag. Print.

Wanderley, Marcelo Mortensen, and Nicola Orio. “Evaluation of Input Devices for Musical Expression: Borrowing Tools from Hci.” *Computer Music Journal* 26.3 (2002): 62–76. Print.

Wang, Ge. “Designing Smule’s Iphone Ocarina.” *Proceedings of the International Conference on New Interfaces for Musical Expression. Pittsburgh*. N. p., 2009. Web. 27 Mar. 2014.

---. *The Chuck Audio Programming Language. a Strongly-Timed and on-the-Fly Environ/mentality*. Princeton University, 2008. Print.

Weinberg, Gil, Andrew Beck, and Mark Godfrey. “ZooZBeat: A Gesture-Based Mobile Music Studio.” *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. N. p., 2009. Web. 27 Mar. 2014.

Wessel, David, and Matthew Wright. “Problems and Prospects for Intimate Musical Control of Computers.” *Proceedings of the International Conference on New Interfaces for Musical Expression* (2001): n. pag.

Zbyszynski, Michael et al. “Ten Years of Tablet Musical Interfaces at CNMAT.” *Proceedings of the 7th International Conference on New Interfaces for Musical Expression*. ACM, 2007. 100–105. Web. 27 Mar. 2014.