Notes on Developing Personal Laptop Improvisation Software

Nathan Ho

May 17, 2019

Abstract

This thesis describes the author's philosophy on the creation of an instrument for laptop improvisation, and a personal project, *FORAY*, that embodies these principles. It employs completely minimal hardware requiring no controllers or outboard gear. Musically, in support of its goal as a live instrument, *FORAY* employs strictly synthetic sound design with zero samples or recorded elements, and eschews the use of step sequencers and piano rolls in favor of algorithmic composition methods.

Rather than an in-depth technical discussion of every aspect of the constantly evolving *FORAY* system, this essay primarily focuses on the process of developing and using the instrument. The hope is that aspects of the project philosophy might resonate with other artists, and the lessons learned while developing *FORAY* will serve as useful advice.

1 Introduction

The motivations and approaches to live creation of electronic music are highly diverse and personal. With the advent of powerful modular computer music environments, technically oriented musicians can devise their own musical workstation tailored to their needs. When the system they develop reaches a threshold of maturity, feedback loops happen between the artist and their rig: their musical style motivates them to extend their workstation, they discover surprising outcomes that shape their stylistic practice, in turn encouraging them to develop the system further. The artist entering a dialog with themselves as an instrument builder and as a musician is a spectacular phenomenon unique to electronic music.

Years ago, as a programmer who had just started learning about electroacoustic music, I had such a vision and began building my own live system for laptop improvisation with fantasies that I would use and develop it for years. I quickly discovered that it is a much harder endeavor than I had imagined, and abandoned it months later when I grew discouraged about the quality of its output. This grew into a cycle — restart project with delusions that "this time around I'll finally make *the* instrument I've been dreaming of!", realize that I have been working on it for weeks with no good musical output, trash it, and start over.

Following an 18-month break, I started another iteration of the project, titled *FORAY*, this time with a mind to avoid past mistakes and focus, above all else, on the quality of the music I produced rather than grand visions of a personal artistic revolution. After eight months of work on the software, I debuted it with a hour-long solo performance, and for the first time the project felt like an accomplishment.

This thesis serves as an informal reflection on my personal philosophy on electronic music, some of the challenges faced while developing the project, and some sparse technical details of the software. It is the guide that I wished I could read before I started this journey.

I have mostly shied away from elaborating on the intense technical details of sound design and algorithmic composition, since much of it is personal to my work and constantly in flux. Although I acknowledge such details that would be of interest to fellow computer musicians, *FORAY* is an artistic work rather than a research project, and I am entitled to keep some secrets about my personal process. I will exercise the right to draw boundaries for what I am comfortable sharing in a publicly available article.

2 The *FORAY* Philosophy

Live electronics present infinite possibilities in the realm of technical setup, and being productive in such an open-ended space requires working with contraints. As my practice as a music technologist matured, I developed preferences for qualities of my ideal electronic system.

The first, and most important, is **productivity**. I am known for having perfectionist tendencies and have trouble resisting the urge to excessively tweak my own work. An ideal music-making environment should be inviting and conducive for making music, and discourage excessive tweaking. My background is in piano improvisation, where the notes played in a recording are nearly impossible to alter in post-production. Completely live creation of music lends a finality to the product, and this is the most appealing aspect to me of live electronics.

- **Price:** It should be as cheap as possible to build and maintain. The advances in the past decades in CPU power and signal processing algorithms have rendered expensive gear unnecessary for the creation of quality electronic music.
- **Durability:** When I sit down to make music, I should not be worrying about whether my software or hardware will break. It should not burn out and stop working in five years.
- **Portability:** In support of Productivity, the system should be easy to transport, set up, and tear down, and easily allow making music on the go.
- **Diversity:**
- **Surprise Factor:**

2.1 Minimal Hardware

FORAY runs on a basic laptop, and live control is exerted using only the laptop trackpad. The on-screen interface consists of a simple array of knobs and buttons to control parameters and trigger musical events.

No MIDI controllers or other outboard gear are used. Confining the system to software ensures the best price, portability, and durability, factors which I consider critically important to a serious music technology practice.

Hardware interfaces are ubiquitous in both commercial and academic electronic music production because of the immediacy of reaching out to a physical knob. Immediate physical interfaces are beneficial to creative flow, and indispensible for critical interfaces such as in live audio engineering (if a mic starts feeding back, whoever is at the mixing board should be able to react immediately). However, the convenience of purely software interfaces should not be discounted for their portability and low cost.

Many electronic musicians employ commercially produced MIDI controllers such as keyboards, faders, knobs, and grid controllers such as Ableton Push. While such commercial products are likely the cheapest and most durable options than any other form of outboard gear, they still add complexity to the system and make it harder to transport.

Many academic live electronics performers develop homebuilt interfaces. Some can be built relatively cheaply, but the real disadvantage is realiability. From my experiences seeing performances and technical demos with custom interfaces, creating a musical interface that achieves acceptable reliability is an ambitious pursuit. It takes hard work and experience with electronics and DIY fabrication to create a sturdy object that is guaranteed not to break ten minutes before a performance.

Laptops have other sensors including a built-in microphone, accelerometer, and webcam. Unfortunately, accelerometers and webcams have variable quality according to the device. Light sensitivity has numerous potential issues that depend on the environment. I currently do not make use of the keyboard either — I have had success using only the laptop trackpad — but I am open to adding keyboard control as the project develops.

Again in the interest of simplicity and portability, *FORAY* is strictly for the creation of stereo music, since I normally do not have access to a multichannel speaker setup.

2.2 Minimal Software

FORAY is built entirely in the free and open source SuperCollider programming language. There are no other programs involved so that the number of moving parts can be minimized.

Although SuperCollider is a powerful environment with a wide variety of audio plugins, it lacks some high-quality DSP algorithms such as good reverberators and compressors. I employ a handful of custom-written UGens. One (a reverberator) has been publicly released and incorporated into the SuperCollider project for others to use, and more are pending.

2.3 Pure Synthesis

FORAY uses only real-time audio synthesis, with no use of sampling or live input.

Live input has material disadvantages. It requires the setup of a microphone, which immediately opens up a number of potential issues: the need to use an audio interface, the need for an environment with low background noise, the danger of feedback when dealing with a speaker system, and the cost of the microphone and cable. All these are points where the system can fail.

Samples are not as problematic in my overall design philosophy, but I still avoid using them. The process of selecting and auditioning samples is not suitable for an improvisational environment, and samples lack the endless parametric capabilities that synthesis has. Most importantly, I simply have a personal preference towards synthesis and I enjoy the challenge of creating entirely synthetic music.

2.4 Real-Time Composition

FORAY sequences notes and sound effects with algorithmic composition methods. No timeline views, piano rolls, or step sequencers are used.

I eschew timeline views due to my concerns with productivity. The timeline editing features of DAWs encourage perfectionist tendencies, making it hard to resist endless tweaking. Compare a recorded improvisation on an acoustic instrument — the result is nearly impossible to change or tweak, forcing the artist to embrace its imperfections.

My musical aesthetics lean toward highly complex, non-repetitive music that constantly changes and evolves. Step sequencers and piano rolls are therefore not suitable for my work.

In place of repetition and traditional note entry, *FORAY* employs a slew of methods for real-time algorithmic composition, parametrically controlled in real time. In this way, creating music in *FORAY* is similar to conducting an improvising ensemble.

There are many musical works that use algorithmic generation as a facet of the artwork's premise and presentation. With no intentions to criticize such works, I do not use algorithmic composition to publicize or promote my work, but rather because of the convenience.

2.5 What *FORAY* is not

Research: *FORAY* is not a research project. It has no novelty in signal processing, interface design, or algorithmic composition. It is not a project built for other people. *FORAY* is a system built solely for my own musical practice, so it is constantly undergoing evolution.

Performance: I am careful to avoid the term "performance" when describing *FORAY*, because a "performance" recalls the history of acoustic music traditions which have aspects of visual engagement. I have used *FORAY* on stage, and there is an undeniable visual experience to me being on stage making various strange facial expressions, but engaging visuals are no goal of the project. A tenet of my system — one which I hope is not too delusional — is that good-sounding music is sufficient for an enjoyable listener experience.

The choice to use fully improvised electronics is rooted in the desire for productivity that comes from escaping timeline views and embracing imperfections and mistakes that inevitably occur in live music.

Live Coding: Despite the fact that I use similar hardware and software tools, *FORAY* is not a live coding project. There is no live design or modification of algorithms, a process I find unnecessarily slow.

2.6 Musical Style

FORAY's musical style falls under Western art electronic music with heavy influences of ambient music. The style uses free-floating rhythms, traditional harmony, and a lack of repetition and droning in favor of constant evolution.

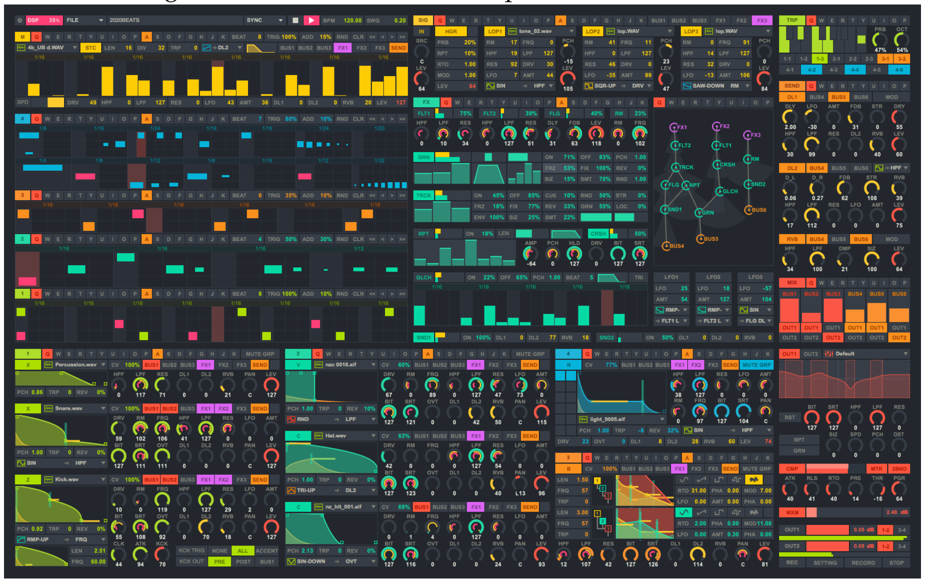
2.7 Prior Art: Yotaro Shuto's *2020*

The primary inspiration for *FORAY* is Yotaro Shuto's *2020*, a live performance software instrument developed for macOS. The alpha release was in 2015. Immediately impressive about this interface is its maximalism, featuring hundreds of knobs with no menus or modality — it is intended as a complete music-making solution where every parameter is in reach. 2020's documentation contains a particularly striking passage on live electronic performance [*sic*]:

> 2020 is like a musical instrument. to become a master of it, you'll need time, inspiration, and endurance.
>
> **5 minutes :** Install it in your computer.
>
> **1 hour :** You will slightly understand how is it.
>
> **1 day :** You will be able to run through all functions.
>
> **1 week :** You will be able to compose FIRST SONG.
>
> **1 month :** You will be able to acquire Tips & Tricks.
>
> **6 month :** You will be ready to perform on a stage!
>
> **1 year :** 2020 will be a part of your body.

FORAY essentially began as my own personal take on *2020* — a live environment where everything is within reach. Like *FORAY*, *2020* contains

Figure 1: A screenshot of the alpha version of the *2020*.



features for randomized sequencing and sound design. Unlike *FORAY*, *2020* is heavily sample-based, employs a steady tempo, and contains "semi-modular" features.

3 Prior Art: Eurorack

Eurorack refers to a large family of hardware products for sound synthesis, sequencing, and processing that can be patched together with cables.

Like *FORAY*, Eurorack rigs are strictly for the real-time creation of music, and lack timeline views. The general direction of the Eurorack market also overlaps significantly with areas of interest in the *FORAY* project, specifically those of experimental sound design and algorithmic composition. Algorithmic composition products include Mutable Instruments Grids, a generative drum sequencer.

4 Sound Design and Mixing

4.1 One Big Patch Syndrome

I consider *FORAY* a successful project, but years of failure precede it. I made at least four or five previous attempts at creating a live electronics rig, which all failed for several personal reasons. I lacked experience with sound design and production, and was still in the process of discovering my style. The most critical mistake in these failed attempts was One Big Patch Syndrome.

One Big Patch Syndrome happens when an artist overthinks the generality of their system and starts developing technical scaffolding instead of making music. "If I just create the one big patch that allows me to make any sound I want, I'll finally achieve musical expression in my electronics!" This attitude just leads to frustration, and usually bad music. For example, in one earlier attempt at an electronics rig, I started by attempting to classify all electronic sounds I could think of, organizing them into a set of categories, and mapping different parts of a MIDI keyboard to different categories. I will leave it to the reader to imagine how musically interesting the result was.

To avoid One Big Patch Syndrome, stop thinking about whether the expression of your system is optimal and whether it achieves full generality in timbre. Instead, just make music. Design some sounds and create a coherent and high-quality musical work. Then design some more sounds, and add more and more parameters. When you have great musical materials, live control will immediately follow. Worrying about the "diversity" of your patch in the early stages of the project will paralyze its development.

The first steps toward the creation of *FORAY* was a set of three different patches that I realized had some properties in common, and could be merged into a single synthesizer with a flexible set of parameters. I then made a little GUI interface with knobs to dial in parameters. Over time I added more parameters that seemed interesting, removed parameters that felt less musically useful, and eventually added a preset system for saving and loading settings. I used a very basic algorithmic sequencer that simply picked random chord tones, and slowly made it smarter and more musically sophisticated. By starting with sound design I was happy with and incrementally building the system around it, I ended up with my first ever success in live electronic rig development.

As specific advice for experienced software engineers building their own workstation: code quality should be extremely low on the priority list when building a personal musical system. When developing custom music software, work quickly ands sloppily, focusing on creating sound rather than making well-architected software. There are many technologically oriented artists who possess only rudimentary coding skills, but are still able to produce successful works because they focus on the quality of the results.

4.2 Mixing and Dynamics

Dynamics are a particular challenge for musicians working with real-time generative and algorithmic music. Heavy use of algorithmically generated note sequences and random modulation makes the balance between different elements inherently unpredictable, and can only be controlled after the fact.

Avoidance of clipping is of course important, and I added a limiter as a safety net in the early stages of the project. Quickly I realized that the limiter could be driven hard as a creative effect. All elements are being mixed in real-time *through* the limiter, rather than the limiter being applied as a downstream stage after mixing. Severely overdriven limiters often create sweet spots where two elements interact with each other and fight for headroom. *FORAY*'s architecture also places the limiter after reverberation, which has an interesting effect on the perception of space.

Seasoned audio professionals frown on aggressive master compression, usually preferring to carefully fine-tune balance at the stem level rather than allow an algorithm to set levels for them. While good advice for more traditional audio production settings, this is absolutely not something I am concerned with as an experimental electronic musician. Imperfect mixing is something to be expected in a live rig with unpredictable generative elements, and I am more than happy to allow an algorithm to exert control over dynamics so I can prioritize sound design and arrangement. Furthermore, the stylistic aims of *FORAY* are huge, maximalist, and intended to be listened at a high volume, and the sound of heavy compression is part of its character. Other forms of electronic music, however, may benefit from quietness and high dynamic range, and may prefer the use of a limiter only as an anti-clipping measure.

In the master bus, I use two soft-knee limiters in series: first a "color" limiter with a fast attack and release, and a second "mastering" limiter with 100ms attack and 1s release and a bit of lookahead. Both are set to nearly the same threshold. The color limiter is intended to be driven hard to create the aforementioned dynamic interactions, and the mastering limiter tames transients so the output track is slightly brickwalled. There is an additional 10 dB of headroom just to be absolutely safe from digital overs. I don't find any need to adjust these limiter settings while playing, so no controls were created — just a visualization of the gain reduction amount of the color limiter.

Exploring the use of limiters in the creation of musical works, I arrived at a playing style I am satisfied with. During quieter sections of works, I tend to stay 5-10 dB below the limiter threshold. During louder sections I aim for moderate gain reduction of 3-5 dB, and in climactic moments I drive the limiter to 10 dB or more. This way, the overall work is dynamic, and the loudest sections are appropriately loud.

4.3 Sound Design

Sound synthesis is a personal practice that can only be developed with experimentation over time, so I am most comfortable keeping secret the specific sound design patches I created in *FORAY*. However, I am willing to describe an overarching philosophy on how I synthesize sound.

My personal go-to sound design strategy is the "effects cocktail": various combinations of filter, chorus, phaser, flanger, reverb, sample rate reduction, pitch shifting, saturation, multitap delays, wavefolding, extreme compression and limiting, granulators, and feedback across any of those. I tend to use simple sound sources and let the effects sculpt them into something more complex, making this an extension of subtractive synthesis.

With overcomplicated effects chains, every element in the chain adds its own flavor, especially if it contains artifacts of some kind. The signal path accumulates a cascade of imperfections and complexities, resulting in wonderfully organic sounds. It can be used subtly as well as dramatically — by mixing a small amount back into the dry signal, a bit of space and character is added.

Two effects in particular, reverb and compression, are particularly powerful ingredients in this methodology. Synthetic sounds tend to be completely dry, so even a small amount of reverb can give the impression akin to the body of a real instrument. (The traditional audio mixing advice of sending all tracks at various levels to a single reverb, while useful, is less important when dealing with synthesis-heavy music.) Furthermore, reverbs create complex signals that have a butterfly effect on downstream elements in the signal path. There is, of course, such a thing as too much reverb, and I often have to suppress the habit of overapplying it to compensate for otherwise bland synthesis patches.

Compression serves a utilitarian purpose of taming unpredictable variations in volume that effects cocktails might create. However, when pushed to the extreme, compression brings out low-amplitude details from upstream effects.

I am particularly fascinated by the possibilities when applying an effect and then imperfectly reversing it. These include pitch shifting up and then down, compression followed by expansion, reverberation followed by crude dereverberation (i.e. noise gating), etc.

Modulation is also a huge part of interesting sound synthesis. I make extensive use of random LFOs created by starting with a random impulse train, using that to trigger random sample-and-hold noise, and then smoothing out the signal with a one-pole lowpass filter. The two controls for a random LFO are its rate and its smoothness. This way, I have control over a continuum from hard glitches to smooth, organic wobbles.

Multiple parallel copies of the same effect or oscillator with slightly different parameters.

4.4 Pacing

Once *FORAY* reached a level of maturity where I could create large-scale musical development, the pacing of musical works in a live environment became a concern. This is a lesson I learned when I presented *FORAY* live at a forum — in my nervousness, I burned through the options I had far too quickly, immediately reaching stasis. States of frustration where I cannot decide where to go next are detrimental to creative flow.

A timer added to the software's GUI helped with awareness, but solving pacing issues is largely a human factor. I come from a background of piano improvisation, where I can do spontaneous composition from intuition. However, since I lack experience improvising on the system that I built, I have to relearn how to pace my works. To get satisfactory development, I found I had to think several steps ahead — how will the piece change over the next five to ten minutes? What is the destination I am trying to arrive at?

Pacing is also perceived differently between the artist and audience. I have been listening and experimenting with these patches for months, but the listener has only been hearing it for 10 seconds. The use of generative sequencing and highly dynamic sound design means that the patch is still evolving quite quickly even if I am not actively changing any parameters. However, during such moments, *I* know how the patch will develop, and I reach impatience much quicker. When I realized how this distorted perspective impacted my music, I started correcting for it by simply developing slower. I lingered much longer on points that I found pleasing. I avoided immediately activating every instrument all at once, instead waiting extended periods of time to introduce them.

By working in a goal-oriented, time-aware manner rather than aimlessly wiggling knobs, and by staying patient and moving slowly, I immediately noticed improvements in the form of my music, a reduced frequency of "stuck" moments, and a much easier time improvising for an hour or more without running out of material or dragging any section out for too long.

Nevertheless, stuck states are still unavoidable. The addition of a preset system proved hugely beneficial as an easy way out of such points, provided that they are not overused. Extremely interesting results happen when crossfading the parameters of distant presets — both settings were human-designed, and intermediate settings tend to have the qualities of both, but a sound completely unpredictable to the designer.

Overall, good pacing with live generative electronics is a problem of finding a balance between interesting development and patient use of material. My personal experiences are that I move too eagerly and had to compensate by slowing down. Other artists might find they have the opposite problem. Either way, developing an awareness for pacing is an important aspect of electronic improvisation, and presents specific challenges for artists building their own live workstations.