Dan Trueman*[*] and Michael Mulshine[†]
[*]Effron Music Building
Lewis Center for the Arts
Princeton University
Princeton, New Jersey 08544, USA
[†]Center for Computer Research in Music
and Acoustics
Department of Music
Stanford University
Stanford, California 94305, USA
dtrueman@princeton.edu,
mulshine@ccrma.stanford.edu

Preparing the Digital Piano: Introducing bitKlavier

Abstract: This article describes a new kind of digital musical instrument, a novel assemblage of the familiar MIDI keyboard with custom interactive software. Inspired by John Cage's prepared piano, our instrument both takes advantage of and subverts the pianist's hard-earned, embodied training, while also inviting an extended configuration stage that "prepares" the instrument to behave in composition-specific, idiosyncratic ways. Through its flexible—though constrained—design, the instrument aims to inspire a playful approach to instrument building, composition, and performance. We outline the development history of our instrument, called bitKlavier, its current design, and some of its musical possibilities.

Of the many subversive acts that mark John Cage's career, preparing the piano is one of his most enduring. Although cramming bolts between the strings of this refined instrument might seem like vandalism, the delicate, almost spiritual act of preparation is irreverent but not sacrilegious, and it begs us to consider the piano as a work in progress, a place for play and exploration, as it was in its adolescence. Roger Moseley, in his book *Keys to Play*, argues that play "owes its existence to rigid rules or material constraints, but takes place despite—and sometimes in opposition to—them" (Moseley 2016, p. 36). Although not directed at the prepared piano explicitly, this aptly captures some of the subversive nature of Cage's approach. The subversion is both mechanical and embodied; it directly impinges on the feedback loop between player and instrument that so much instrumental training relies on. Not unlike *scordatura* with stringed instruments, preparing the piano rewires the digital fingering patterns that the thousands of hours of practice embed in our bodies, creating new and unexpected analog sonic patterns, many of which would be literally impossible with the "properly tuned" instrument. Though subversive, this rewiring still leverages the embodied and notational training of the player; Cage's Sonata V from the *Sonatas and Interludes* (1946–1948) is easy to sight-read for a trained pianist, and yet what is heard is not even recognizable as piano music. Finally, as anyone who has experienced one knows, the prepared piano is fun and inspiring to play, and it changes how we play. In the words of Jonathan De Souza, "changing the instrument, then, changes the player. Alteration illuminates everyday experiences of instruments, even as it disrupts them" (De Souza 2017, p. 23).

Like the prepared piano, but via a fundamentally different mechanism, the prepared digital piano subverts the expected feedback loop between player and instrument while still exploiting the player's musical training. Rather than placing mechanical objects between physical strings, algorithms are placed "between" the virtual strings of the digital piano; in place of analog, physical interventions, we have digital, algorithmic interventions. These interventions can take many forms, and in this article we will explore those found in bitKlavier, a software instrument developed and used extensively by the authors and others. The process of creating composition-specific, idiosyncratic preparations—be they simple or involved—is also shared by both types of prepared pianos. In the case of Cage's *Sonatas and Interludes*, Cage instructs the preparer to "have free two to eight hours, and put yourself in a frame of mind conducive to the overcoming of

Computer Music Journal, 43:2/3, pp. 48–66, Summer/Fall 2019
doi:10.1162/COMJ_a_00518
© 2020 Massachusetts Institute of Technology.

obstacles with patience." A similar patience may be required of a composer preparing the digital piano, although, as we will see, a complex set of digital "preparations" can be saved, restored, and modified instantaneously, in ways impossible with the analog piano. These two complex characteristics—a detailed preparatory process and a subverted, but still utilized, player–instrument feedback loop—are essential to both instruments, but their natures are different in crucial ways.

We see bitKlavier as an "assemblage" of hardware and software, in which both components reach beyond their typically conceived bounds into broader performance, compositional, and computer music practices. Georgina Born has developed a rich, inclusive, and unstable notion of assemblage, one that combines multiple objects, histories, and practices into a network of relationships, and she has applied it to a range of musical activities, including electronic music (Born 2005). Paul Théberge has extended this notion in the context of musical instruments, asserting in regard to instrument design that "the project is not so much to design objects, but to design relationships" (Théberge 2017, p. 66). For us, the traditional layout of the keyboard—its uneven alternation of black and white keys, and how the keys interact with our hands—is a primary, inseparable design constraint for the software, guiding how the various digital preparations are constructed and respond to a player's performance:

> The interface of the keyboard can be approached as a zone where the digital and the analog come together under the rubric of play (Moseley 2016, p. 70).

Furthermore, from Philip Alperson:

> Musical instruments are not objects divorced from performers' bodies, but rather are intimately tied to the performers' bodies, so much so that, in some cases, it is difficult to know where the body ends and where the instrument begins (Alperson 2008, p. 46).

So, on the one hand, bitKlavier engages the long history of piano performance practice, interfacing with the hard-earned, embodied technique of the pianist's body and, by extension, the centuries of associated piano repertoire. On the other hand, it contains a software program that is itself programmable, constrained, and enabled by the digital world. These two aspects come together in an instrument design project that is focused on fostering new relationships between musician and machine. We have intentionally chosen to limit the design of the software to interface with a standardized version of the MIDI keyboard, even avoiding such normal add-ons like knobs, sliders, expression wheels, and so on, and have consistently aimed to promote a kind of "embodied interaction." We have taken, following Paul Dourish, "an approach to the design and analysis of interaction that takes embodiment to be central to, even constitutive of, the whole phenomenon" (Dourish 2004, p. 102).

In what follows, we detail the history of bitKlavier, including the design priorities and development process. To help the reader understand the bitKlavier paradigm, we first describe antecedent projects that we developed. Although these lacked a keyboard-oriented interface, they introduced fundamental features that reappeared in bitKlavier. Having laid that foundation, we summarize the current design of bitKlavier—its primary components and typical workflow—and consider a concise example that takes advantage of many of its capabilities. Finally, we close with some broader thoughts on the nature of preparing the digital piano, on the ongoing potential of the keyboard as an interface between body and machines, and how it shapes musical imagination, all while inspiring play and creativity. The source code for bitKlavier is freely available open source, downloadable at bitklavier.com, and available on iOS, macOS, Windows, and Linux (we hope to support Android in the future); the reader is encouraged to work directly with the application while reading this article and the bitKlavier manual (Fishman and Trueman 2018), the only bitKlavier publication to date.

Antecedents

Neither bitKlavier nor the prepared digital piano were fully formed concepts, envisaged and then constructed. Rather, they are the ongoing, emergent products of a creative process, a process that began with a commitment to inspiring play, exploration, creativity, and composition. This decade-long compositional process has its roots in the Princeton Laptop Orchestra (PLOrk), where the building of digital musical instruments (DMIs) for others is required (Trueman 2007; Smallwood et al. 2008; Trueman 2011). Often PLOrk demands a relatively old-fashioned approach to building digital instruments, in which stability, usability, and transparency are important. This is driven by the expectation that the instruments be designed for others to use, unlike the familiar personal practice of DMI development, in which there is no expectation that anyone other than the practitioner will ever use the "system" and the system is in a constant state of flux, sometimes undergoing revision backstage during intermission. Taken to the extreme, "revision" may even continue on stage, as with live coding (cf. Collins et al. 2003). Although the PLOrk approach has many limitations (and in no way do we mean to imply any kind of superiority; it's simply different), it does encourage the development of instruments that can be shared, used by non-technical musicians in traditional music-making contexts, and composed for by—potentially—many composers. We think of this as a generous approach to DMI building (for more on DMIs in general, cf. Jordà 2002; Magnusson 2009), one with severe limitations that requires time and commitment, but one that invites others to use the instrument, and even impact its development over time.

It was in this context, long before it was so named or even associated with a keyboard interface, that bitKlavier began, with a challenge: to compose a piece for the extraordinary chamber ensemble So Percussion with laptops in the mix. We tell the story of this piece in part to introduce the primary characteristics of the instrument, but also to openly detail a creative process. This latter aspect seems particularly relevant today, when digital instrument building, composition, and performance are often intertwined, rapidly coevolving components of a larger creative practice. Finally, the nature of bitKlavier is sharply characterized by where it started, and by this development process.

120bpm

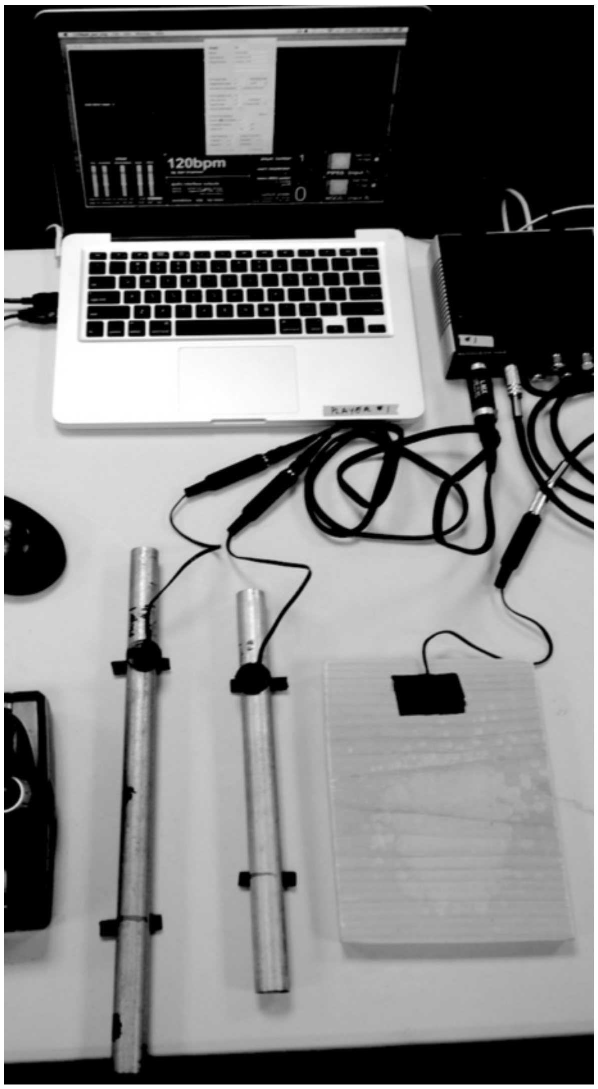
From 2008 to 2010, coauthor Trueman composed *neither Anvil nor Pulley*, a 42-minute quartet commissioned by So Percussion, which made extensive use of digital instruments created by the composer. The second of the five movements, "120bpm," relies on a simple instrument—informed by instrument-building practices in PLOrk—that ultimately led to the development of bitKlavier. One of Trueman's goals with the instrument was to challenge the core musicianship of So Percussion, as highly skilled individual players and as a consummate chamber ensemble. The instrument has two linked components: a resettable "metronome" and a reverse "tape delay" that is dependent on timing relationships between player and "metronome." Physically, this takes the form of a plank of wood and two metal pipes, all with piezoelectric pickups, connected to a software instrument, shown in Figure 1.

Resettable Metronome

Left alone, the resettable metronome will behave like a digital metronome, generating a regular click with a given tempo. Metronomes are generally not interactive; they can be turned on (at which point they begin pulsing), turned off, and their pace can be manually adjusted. The key component of this instrument is that the phase of the metronome can be reset by striking a small plank of wood; the plank has a piezoelectric pickup attached to it, which is connected to attack detection software (bonk~; see Puckette, Apel, and Zicarelli 1998), which in turn causes the metronome to restart whenever an attack is detected. (See Figure 2; video and audio examples for this article can be viewed at https://www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00518.)

This simple form of interaction is surprisingly compelling, both for an individual player and for a group of four all playing their own resettable metronomes together. While percussionists are trained to play with metronomic precision and have extensive experience playing with static metronomes and click tracks, they don't generally

Figure 1. Physical components of the 120bpm instrument. Striking the wood plank (with attached piezoelectric pickup) resets the phase of the digital metronome. Striking the metal pipes (also with attached pickups) triggers reverse delays synchronized with the digital metronome.



interact with metronomes in this way, and hearing the metronome adjust to their own subtle and variable microtiming can be disconcerting.

Consider this simple instruction: "strike the woodblock one 16th note after each click, the click interpreted as a quarter note." Players can interpret this in two ways. In the first, they maintain a consistent 16th-note duration, in spite of the fact that the time between clicks will be more than a quarter-note after its phase is reset by the woodblock, essentially transforming the meter to $\frac{5}{16}$ (see Figure 3).

So the 16th-note rest will no longer be a 16th note relative to the new (longer) click spaced "quarter-note." In addition, this $\frac{5}{16}$ measure length will be somewhat variable, depending on the timing of the player, so this will no longer be mechanically metronomic, though the pulse length following the player's strike *will* be mechanically precise.

The second (more complex) way the player might approach this is to try to have the "16th" note be accurate relative to the changing spacing of the "quarter-note" clicks; because each wood-sync strike will delay the subsequent click by a certain amount, these durations will get longer and longer, eventually converging as in Figure 4.

We can model this behavior as follows, where T is the time between clicks when the metronome is left alone, and T_i is the time between clicks after the ith attempt by the player to play one 16th-note after the click:

$$T_{i+1} = T + \frac{1}{4} \cdot T_i. \tag{1}$$

Or, more generally as in Equation 2, where F is the fractional note value (so, 1/4 in this case):

$$T_{i+1} = T + F \cdot T_i. \tag{2}$$

This function converges to Equation 3, where T_c is the time between clicks after the player has converged on a stable tempo (substitute T_c for both T_{i+1} and T_i in Equation 2 and solve for T_c):

$$T_c = \frac{T}{1 - F} \tag{3}$$

or in terms of beats-per-minute (M for metronome):

$$M_c = (1 - F)M. \tag{4}$$

So, in our example here with the quarter-beat delay, T_c will converge on $4/3T$, and metronome mark at $3/4M$. (See examples in Videos 3, 4, and 5, all illustrating this phenomenon with different F values, including 1/2, which yields an interesting convergence.)

This simple interaction can result in complex behaviors that put an intense physical focus on the relationship between musician and machine (we encourage the reader to experiment with these

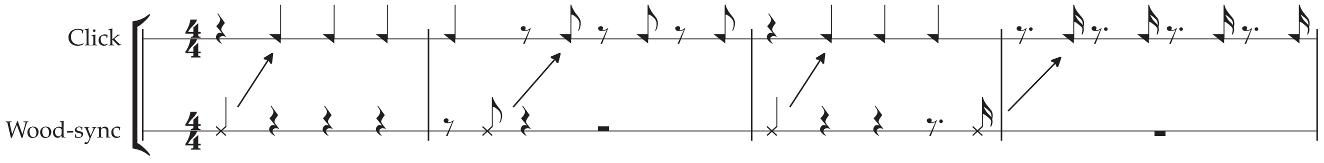


Figure 2

Figure 2. Simple rhythmic offsetting with the resettable metronome: Striking the wood plank restarts the timing of the metronome, shifting its pulse to different beats within the notated meter. (See Video 1.)

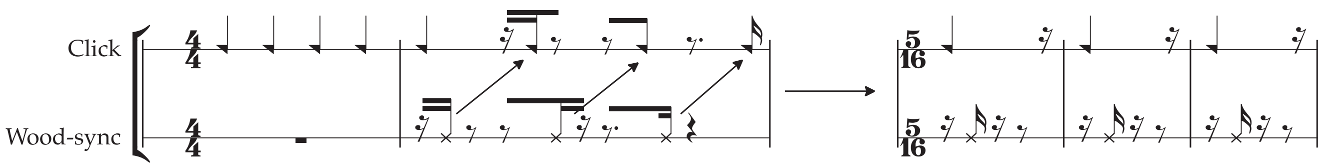


Figure 3

Figure 3. One possible outcome when a performer plays one 16th note after each click: The meter is effectively changed to $\frac{5}{16}$. (See Video 2.)

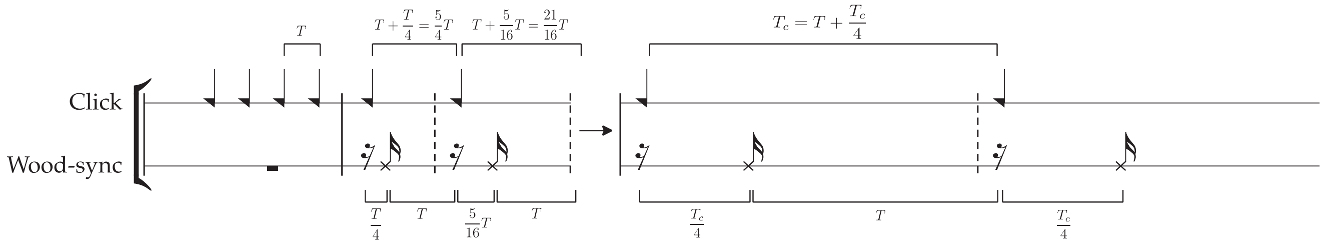

T
$T + \frac{T}{4} = \frac{5}{4}T$
$T + \frac{5}{16}T = \frac{21}{16}T$
$T_c = T + \frac{T_c}{4}$
Click
Wood-sync
$\frac{T}{4}$
T
$\frac{5}{16}T$
T
$\frac{T_c}{4}$
T
$\frac{T_c}{4}$

Figure 4

Figure 4. A second possible outcome of an instruction to "play one 16th note after each click." As opposed to Figure 3, the "beat" becomes longer after each stroke. (See text for further discussion of the equations and the example Video 3.)

examples directly). It reminds us of the process of phasing, in, for example, Steve Reich's *Drumming*, where one player remains steady and the other gradually slows down (or speeds up) until they synchronize again, in a new combination, but in this case a completely deterministic machine is embedded in the process.

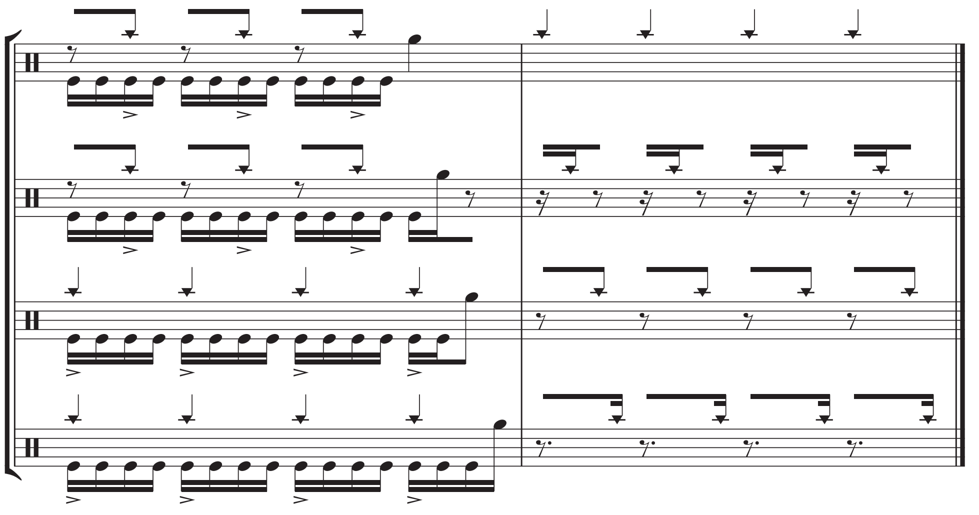
When four players gather to play, each with one of these resettable metronomes, even very simple challenges can prove engaging and revealing. In "120bpm" at the moment shown in Figure 5, each player resets the metronome a 16th note apart from one another (note above the top line of the staff), resulting in interleaved 16th-note clicks (triangular noteheads), which of course won't be perfectly metronomic, reflecting subtle variations in the ensemble timing (see Figure 5).

Although perfect accuracy is, of course, dazzling when it happens in performance, our view is that the imperfections themselves are compelling, just as they are in conventional performance practices, and players should aspire to perfection but embrace the idiosyncrasies that arise from not quite reaching it.

Timed Reverse "Tape" Delay

The second component of the "120bpm" instrument is linked to the resettable metronome and introduces a pitched element via two tuned pipes, with attached piezoelectric pickup (see Figure 1 again). When the player strikes the pipe, the algorithm does three things: (1) it determines how much time until the next click T_{nc}; (2) it samples the sound of the pipe

Figure 5. Interleaved 16th note metronome clicks in "120bpm," for percussion quartet. Triangular noteheads above the staves represent the four metronome pulses, whereas the last played stroke for each performer at the end of the first measure represents a wood-plank strike that resets the metronomes; these last strokes are then separated by a 16th note, as are metronome pulses that follow. (See Video 6.)



for half that time $T_{nc}/2$; (3) it waits $T_{nc}/2$ and then begins playing the sample in reverse, so that it peaks at T_{nc} in sync with the next click. This is illustrated in notation in Figure 6, assuming a click at the beginning of every $\frac{4}{4}$ measure.

This system is inspired by old tape delay systems, where different delay times can be set by arranging the play and record heads at different distances. The algorithm includes a number of simple variations. For instance, it can be set to skip a certain number of clicks, so that at a fast tempo a slow reverse build would be possible. It can also be set to transpose the metal pipe (via basic playback speed variation, like a tape machine), so that dyads between the pipe and the reverse pipe are possible; in this case, the sampling and playback time have to be adjusted to account for the change in playback rate. In "120bpm" each player has two pipes of different pitches and uses a foot pedal to change the transposition and skip values, affording the possibility of changing many-voiced sonorities. Consider the excerpt in Figure 7.

As in the earlier example in Figure 5, the note above the top line of the staff is the woodblock that resets the metronome and the triangular notehead above is the click. The top line and space are the two pipes, and the *x* notehead below is the foot-pedal that resets the transpositions. The small parenthetical noteheads indicate the transposition of the pipe and can be read as the sonority that will be heard (assuming treble clef). Each measure is repeated N times, as determined by the ensemble, so the foot pedal is pressed at the beginning of each section, but not for each repeat (the boxed "p" numbers above the staff are preset numbers). Because of the variable placement of the metronome resets, these loops are of varying lengths, along with the reverse sonorities.

Machine Metaphors, Interactive Impossibilities, and Repeatability

Both the resettable metronome and the variable reverse "tape" delay are simple algorithms based on familiar machines, and both include "interactive impossibilities"—ways of manipulating the virtual machines that are impossible with the actual physical devices. Instantaneously resetting the phase of a mechanical metronome is impossible, as is instantaneously moving the tape head on a tape delay system to the proper distance and then reversing at the proper time. Both can also be understood quickly by simply playing with them, and are completely deterministic; while randomization of the various parameters would of course be possible, our preference is for completely predictable systems that are revealing in how they consistently respond to human performers—the

Figure 6. Basic structure of the reverse "tape" delay in "120bpm." On beat 2, the performer strikes the metal pipe, which triggers live sampling of the pipe. The next metronome click is three beats later, so the sampling continues halfway to that point, for 1.5 beats; at this point the sample begins to playback, in reverse, ending in synchrony with the click on the downbeat. Note that in the video example (Video 7) a tuned bowl is used in place of the metal pipes.

Figure 7. Harmonic changes via transposed reverse tuned pipes in "120bpm." Treble clef for pitches. (See Video 8.)

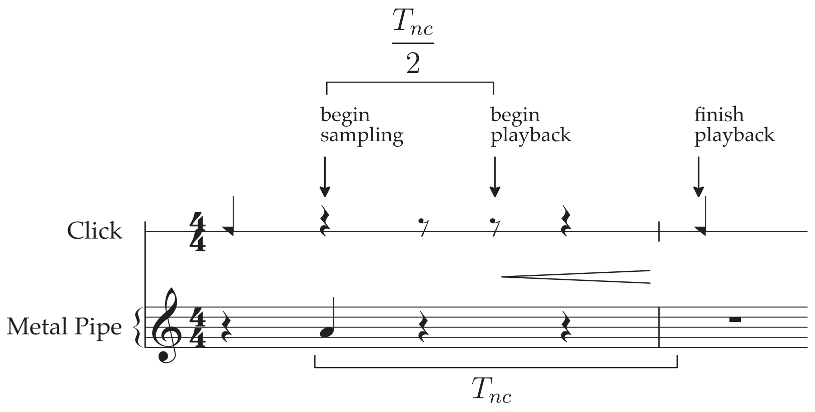


Figure 6

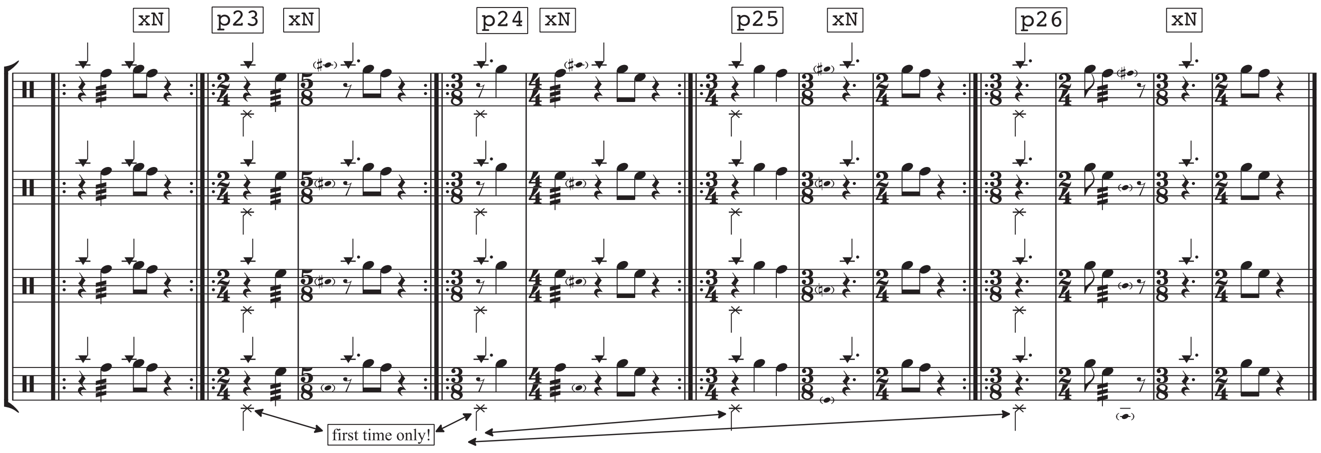


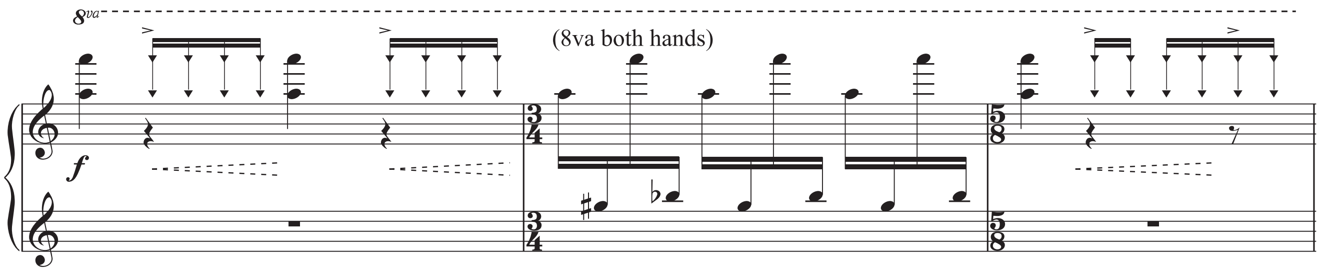
Figure 7

variations come through that relationship, not through some internal and unpredictable behavior in the machines themselves. Just as we don't want our metronomes to be unpredictable (as Beethoven's may well have been! [Forsén et al. 2013]), we don't want these virtual machines to surprise inexplicably. In spite of Cage's later association with aleatoric music, when he was working with the prepared piano in the 1940s he expressed a similar desire for repeatability, and was highly disappointed when he discovered how difficult it was to precisely reproduce his preparations on different instruments and with different materials; this frustration, in fact, was partially responsible for his later interest in chance procedures (Bunger 1973).

To the Keyboard

This is a good moment to point out that the idea of the "prepared digital piano" was a consequence of a long exploratory process that was pragmatic as much as it was creative, not so unlike the emergence of the prepared piano itself with Cage.

Figure 8. Key releases (noteOff events) launch metronomic pulses; noteOn events stop the metronome. Excerpt from Trueman's Nostalgic Synchronic, Etude No. 1 (cf. Audio Example 1).



As Cage himself said, "I'm only being practical," referring to his making the most of having only a piano in a small dance studio (Cage 1951). In this case, attaching piezoelectric pickups to a wood slab and steel pipe and then connecting them to an audio interface and some kind of attack-detection software is not trivial, and very similar information (attack time for a number of items) can be provided easily (and more reliably) with a MIDI keyboard. The first experiments with what would ultimately become bitKlavier began precisely this way, replacing the combination piezoelectric pickup, wood, and pipe with a MIDI keyboard to facilitate easy exploration of the instruments from "120bpm" and then intuitively managing the consequences. Without the sound of the wood and pipe, what should we hear? Well, we're working with a piano keyboard, and piano samples are easy to come by, so let's use those, including for the metronome (in place of a click). And now that we have ten fingers instead of two mallets, what are the performance technique consequences, and how do they impact the metronome and reverse delay algorithms? Without detailing all of these issues here, we will consider a couple of key examples.

NoteOn and NoteOff Messages: Releases and Durations as Control Types

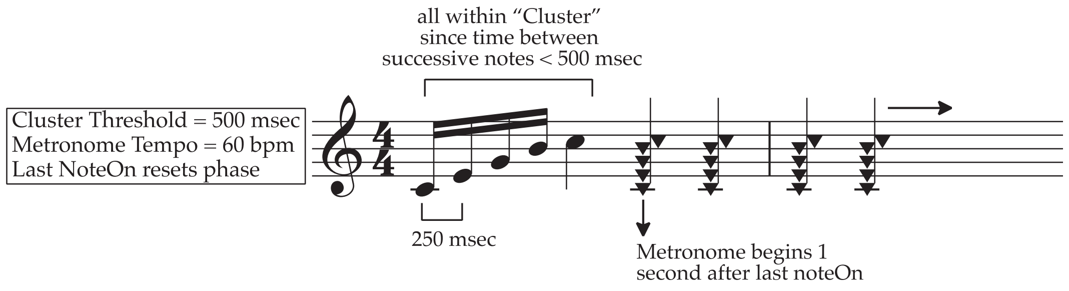
Substituting a MIDI keyboard for the combination woodblock with piezoelectric pickup most obviously means using MIDI noteOn messages to reset the metronome. Using noteOff messages is equally straightforward to implement, however, and this dramatically changes the physical interface with the instrument and the expressive nature of the gesture: Instead of striking something, we are releasing something. And if we use noteOff messsages for synchronization, do we just ignore noteOn messages, or do they have some other effect?

With bitKlavier, we decided to allow the user to choose whether noteOn or noteOff is the synchronization source, as both offer compelling possibilities. We also decided to have noteOn messages suppress the metronome when in this new noteOff mode. Consider this example, from Trueman's *Nostalgic Synchronic* (see Figure 8; bitKlavier XML files for use with the musical examples are also available with the supplemental materials for this article at https://www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00518).

Here, as long as keys are depressed, the metronome pulse is silent, and as soon as they are released, a 16th-note pulse begins, precisely when the keys are released, so in this case the quarter notes must be held for their full values for the 16th-notes to begin on beat 2. This focus on key release naturally has a significant impact on performance technique, an issue we will revisit in a companion article (Sliwinski 2020).

Having noteOff messages also affords the possibility of another way to approach the reverse tape delay; in addition to having it synchronized to the metronome, it could simply have a duration set by the length of time that the key is depressed. Note in Figure 8 the dashed crescendo lines; these represent the quarter-note length swell of the reverse notes, peaking a beat later as set by the length of the sustained first note.

Figure 9. Notes played individually can be "clustered" and treated as a single chord. With the settings used in this example, an arpeggio is treated by bitKlavier as a single event triggering a response from a metronome, which plays back the same notes as chords (cf. Audio Example 2).


all within "Cluster" since time between successive notes < 500 msec
Cluster Threshold = 500 msec
Metronome Tempo = 60 bpm
Last NoteOn resets phase
250 msec
Metronome begins 1 second after last noteOn

Chords, Clusters, and Pitched Metronomes

Since MIDI note messages are always streamed sequentially, there is no such thing as a true "chord" or simultaneity (Moore 1988); how, then, does a chord, or even a rapid arpeggio, set the timing of the reverse metronome, and what does that metronome actually sound like? We chose to allow the user to prepare the instrument so that a minimum period of time between successive notes—the "cluster threshold"—must be exceeded for a cluster of notes to be considered "complete." If this value is very small—say, 20 msec—then anything but a chord will cause a reset, but if it is somewhat larger—say, 500 msec—then it would be possible to play an arpeggio and have its notes collectively set the timing of the metronome. The user defines whether the first note or the last note of the cluster sets the timing, and whether noteOff or noteOn messages are used for in this purpose. All the notes within this cluster are then included in the metronome pulse as a single, repeating chord of pitched piano samples (see Figure 9).

Later in this article, we will look more closely at how both of these algorithms from "120bpm"—what we ultimately renamed the Synchronic (resettable metronome) and Nostalgic (variable reverse "tape" delay) preparations—transform and expand when inhabiting the physical and virtual domains of the keyboard.

There are many other variables that can be set to "prepare" these algorithms in specific ways, some of which we will address later in this article, and all of which are detailed in the bitKlavier Manual (Fishman and Trueman 2018).

Development Architecture and History

The original development of bitKlavier took place from 2011 to 2016 on macOS, using ChucK (Wang and Cook 2003) and Max (Puckette 1991), with the two environments communicating via Open Sound Control (OSC, cf. Wright and Freed 1997). ChucK handled the audio duties—loading samples, scheduling, and mixing the metronome and reverse delays—while Max provided the user interface (UI) and MIDI device and message handling. The utility Platypus (http://sveinbjorn.org/platypus) wrapped the components into a single double-clickable application, which made sharing the application with players easy and minimized the technical know-how required. This is a model that has been used by many composers with PLOrk over a number of years.

Both ChucK and Max are powerful composition and creative environments, allowing for rapid experimentation and prototyping. Indeed, the original algorithms for "120bpm" were a product of a lengthy series of creative coding sessions (see, e.g, Maeda and Burns 2004; Hugill and Yang 2013); the ideas emerged from the coding sessions themselves, rather than being fully developed a priori designs that were realized through code. Similarly, the subsequent evolution of these algorithms and the original version of bitKlavier as a whole were products of a creative process in which coding and composition progressed side by side.

ChucK and Max are not primarily intended for industrial product development, however. Max can export "standalones," but the ChucK code is always interpreted, and although the wrapping of Max and

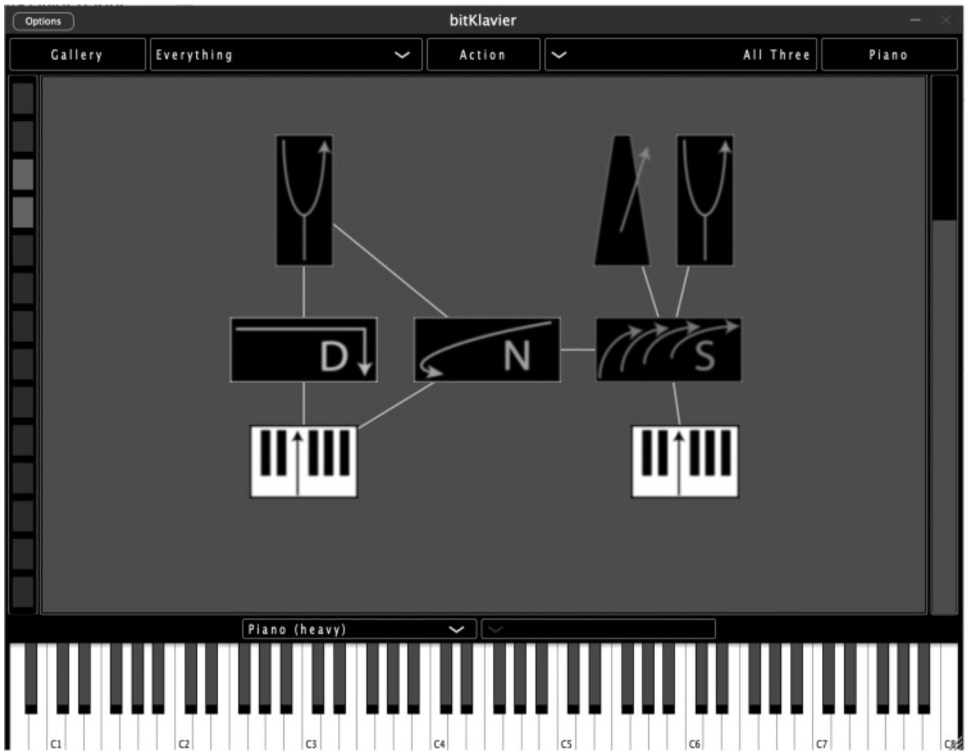
ChucK using OSC and Platypus is effective, it still poses challenges when trying to develop a stable, easy-to-use, cross-platform application for others to use. In 2016, we actually went so far as to develop an iOS version of bitKlavier, replacing the Max UI with a new one written in Objective C, but still using ChucK internally, communicating with the UI via OSC. This experience led us to reconsider future development of bitKlavier, with three primary goals in mind: (1) support for multiple platforms, including mobile devices, to increase access; (2) open source and version-controlled codebase publication, to facilitate multiple developers and ongoing development; and (3) efficiency.

Our experiences making an iOS version of bitKlavier already highlighted difficulties with the first goal, and we had run into similar difficulties trying to make a usable Windows version. Open-source Max code is possible, but version control promised to be challenging, where merging Max patches with their right-to-left order of operations (among other things) can lead to coding problems in which errors are difficult to locate. Finally, ChucK is not a highly optimized language; even just loading the sample library itself might take upwards of two minutes, which feels like an eternity when sitting on stage in front of an audience.

In 2017, we decided to redevelop bitKlavier from scratch using the C++ audio framework JUCE (http://juce.com). Initially, our intent was to exactly reproduce the look and functionality of the original bitKlavier, but we were immediately faced with questions of code architecture that invited us to rethink even some of the fundamental design of the instrument. For instance, the original bitKlavier hardwired a single version of each preparation type to a keymap that determined which keys would activate that preparation, which meant that we could only use one type of preparation, and the preparations couldn't easily share keymaps. This was, in part, a legacy of the prototyping process, in which we simply wanted to get something working quickly so that we could compose with it. But it was also partially due to the obstacles to true object-oriented programming in Max—Max's ease of use was powerful during the composition process, but the resulting code was inflexible and did not scale well. When redeveloping in C++ we had to answer fundamental questions about code design before we could do anything (quite the opposite of Max), and we carefully broke up the original code into small classes that could be recombined in various ways and that supported multiple instantiation. The result is a composition environment with far greater flexibility, one in which the new codebase informed the UI, which itself is more object-oriented as well. Ironically, the UI bears more resemblance to Max, with objects that can be connected in various ways, than the original bitKlavier did, despite the fact that it was built in Max. The new instrument is also far more efficient, which is most obvious while loading samples; what used to take more than two minutes now takes a few seconds.

Although development in C++ is more cumbersome than ChucK or Max, we are still developing bitKlavier in tandem with composing, and the codebase is actually cleaner, more flexible, and easier to read than what had it had become in the awkward combination of the two prototyping languages. Also, the focused design of the instrument requires us to ask more questions about what we're trying to accomplish with new functionality and guards against "feature creep" (cf. Mission 2018 for a discussion of this problem). Indeed, it would be a fair question to ask why we would invest so much in what is really a simple design, algorithms that—in principle—could be coded quickly in any of the many environments available (Max, Supercollider, ChucK, etc.). The answer is complex, but part of it is a commitment to a particular design, to refining that design as much as possible, and to abiding by particular choices. Ge Wang's *Artful Design* (Wang 2018) offers a rich perspective on this approach to design. We also believe in the principle of creating complexity from simplicity, as argued by Perry Cook (2001) and by Thor Magnusson (2010), and we see the commitment to simplicity in bitKlavier as an asset for performers, composers, and students alike. Other parts of the answer include sustaining a commitment to all the repertoire and performance practices that might grow with the instrument, making bitKlavier accessible to nontechnical musicians, and making it as sustainable as possible (the cleaner codebase with fewer dependencies should be

Figure 10. Main preparation window in bitKlavier, with core preparation types.



easier to maintain and extend over the years). These are all priorities that require significant investment.

Current Design

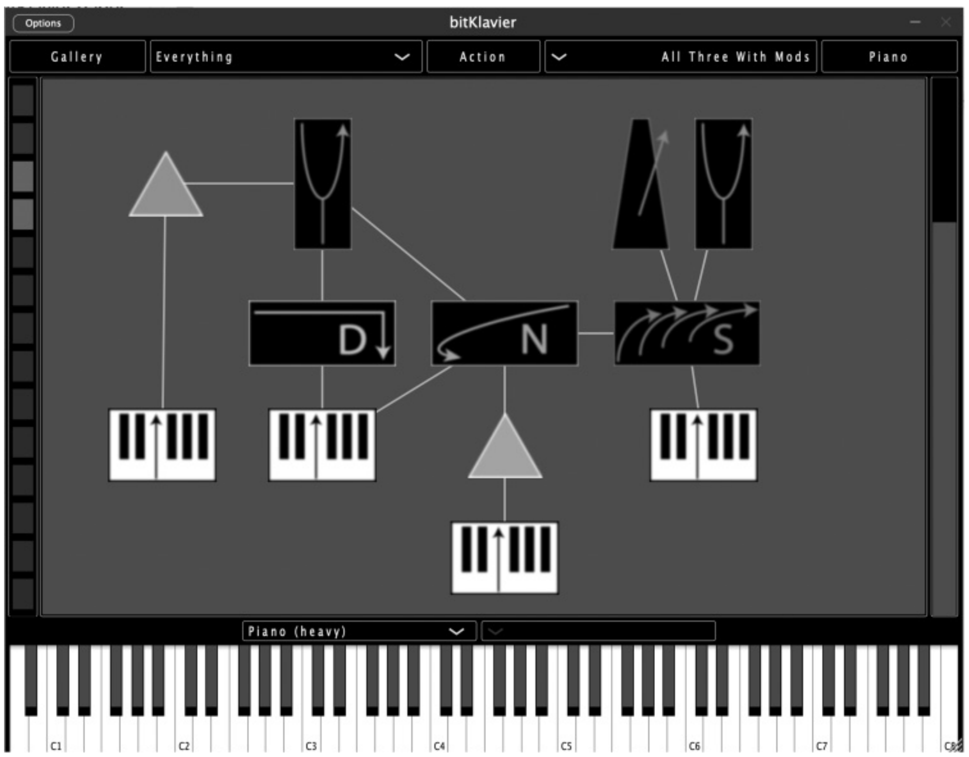
A comprehensive explanation of all the features of bitKlavier is beyond the scope of this article (for the complete manual see Fishman and Trueman 2018), but we provide an overview of the design and some closer looks at particular aspects of the instrument.

The Preparation Construction Site

The prepared digital piano in Figure 10 contains many of the key components of bitKlavier.

Across the middle we have the three primary types of preparations: Direct (the actual direct sound of the piano, which can be modified, as we will see), Nostalgic (the variable reverse "tape" delay), and Synchronic (the resettable metronome). Below those are two Keymap preparations (with keyboard icons); with these, particular keys are set to activate the preparations to which they are connected, so the Direct and Nostalgic preparations will be activated by the same keys, and the Synchronic preparation by a different set of keys. Across the top we have two Tuning preparations (with tuning fork icons) and a Tempo preparation (with a metronome icon); again, the Direct and Nostalgic preparations share a Tuning preparation while the Synchronic preparation has its own Tuning. Finally, the Nostalgic and Synchronic preparations are connected, defining which Synchronic pulse should be used by that Nostalgic preparation to set the timing of the reverse delay. These objects can be moved around, reconnected, deleted, and so on, and new objects can be added as needed, in a manner similar to a Max or Reaktor patch. Several pianos like these (this one is called "All Three" as seen in the menu second from right) can be saved in a

Figure 11. Addition of Modification preparations (triangles) that can modify other preparations in performance.



"gallery" (called "Everything", in the menu second from left).

Double-clicking on any of these preparations will reveal its inner workings, where various parameters can be set. We will take a look at each of these shortly, but we first expand this particular piano with the remaining types of preparations (Figures 11 and 12): the Modifications (triangles that take on the colors of the preparations to which they are attached), which allow the user to change (via a Keymap) any of the parameters of a particular preparation; and the Piano Switch preparation (the trapezoid to the right in Figure 12), which causes the piano to switch immediately to another in the same gallery (this one will switch to the piano called "Totally Different").

Finally, preparations can be reset to their original settings (see Figure 12, the octagonal "stop" signs), undoing any prior modifications, again via a Keymap, allowing the player to make modifications and return fluidly while playing the instrument.

The Synchronic and Nostalgic Preparations

By double-clicking on the Synchronic icon we can see its internal parameters (see Figure 13).

Again, without detailing all of the features, we point out a few salient ones. The Synchronic preparation extends the original resettable metronome in a number of ways. Our metronome is now pitched, so we can transpose it (top right), and we can also specify a sequence of transpositions. (In the example displayed in Figure 13 it will continually switch between untransposed and down a major third, as set by the consecutive vertical sliders) We can create sequences of other parameters as well, such as accent patterns. In this example we have three accent values through which we sequence, these will phase with the two transposition values. To the left are static parameters, like the "cluster threshold," which is exactly as described in the section on "120bpm."

Figure 12. Addition of Reset preparations (octagonal "stop" signs) that reset previously modified preparations to their saved values, and the trapezoidal Piano switch preparation.

Figure 13. The Synchronic preparation window. In bitKlavier, the metronome from "120bpm" becomes pitched and transposable, and can have variable beat lengths and accents. Popup menus offer options for configuring how the metronomes are launched (by noteOn or noteOff messages, for instance), and for saving and loading individual settings.

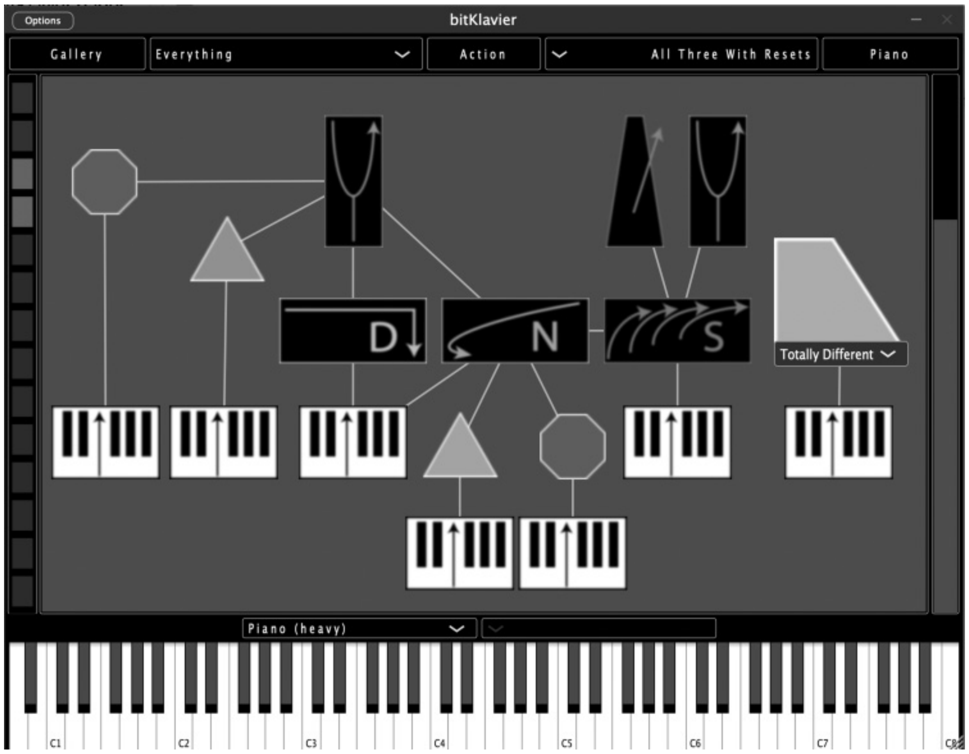
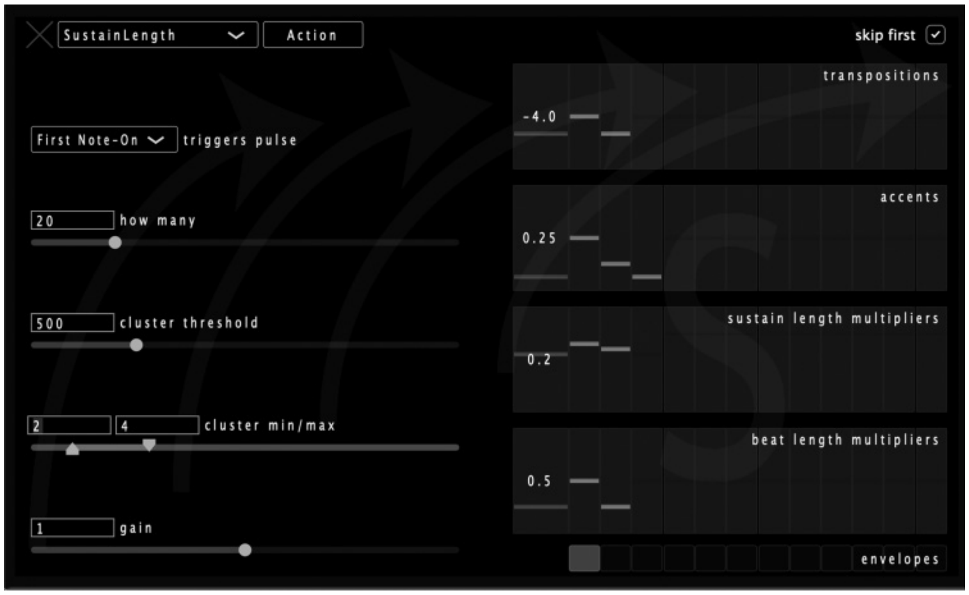


Figure 12



SustainLength Action

skip first ✓

First Note-On triggers pulse

20 how many

500 cluster threshold

2 4 cluster min/max

1 gain

transpositions -4.0

accents 0.25

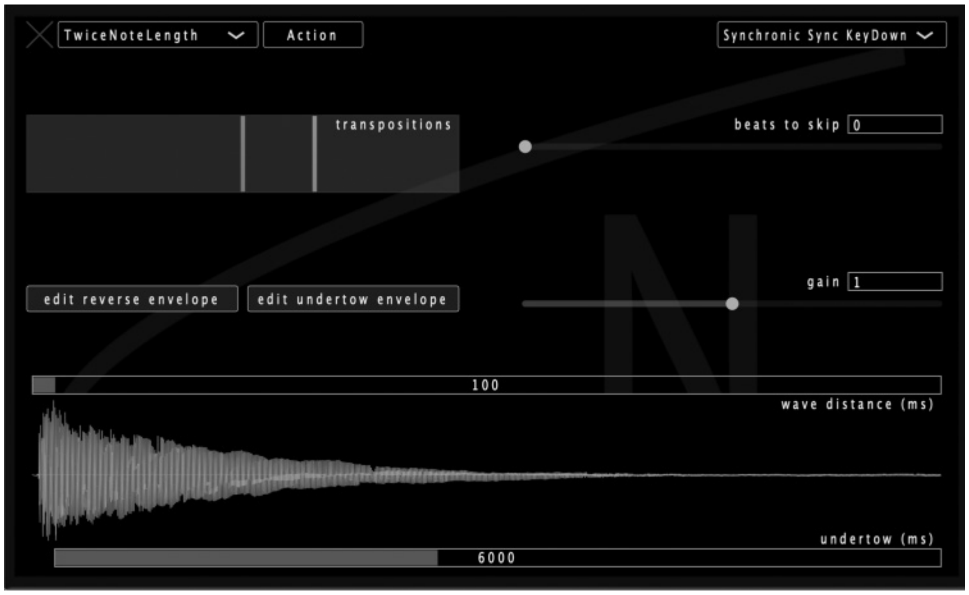
sustain length multipliers 0.2

beat length multipliers 0.5

envelopes

Figure 13

Figure 14. The Nostalgic preparation window. In bitKlavier, the reverse "tape delay" can be layered with transpositions, and can be extended with forward playback after the reverse sample has finished playback (the "undertow"). Popup menus allow for envelope shaping, saving and loading of settings, and for configuring whether the length of the notes are set by an attached Synchronic preparation (as in this image) or by the length of time the launching key is held down (note duration).



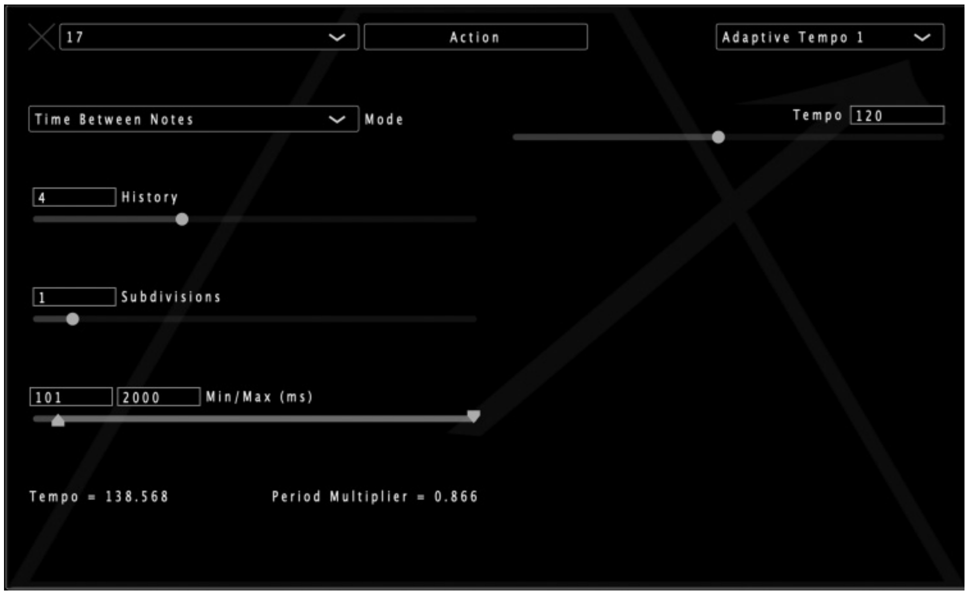
The variable reverse tape delay of the original version of bitKlavier evolved into the Nostalgic preparation (see Figure 14). As with the Synchronic preparation, the reverse notes can be transposed and layered (notice the two sliders labeled "transpositions" at the top left of Figure 14, with its two vertical lines signifying transposition values), and the number of "beats to skip" when syncronizing to its paired Synchronic can also be set. Two extensions to the original algorithm are the "wave distance" and "undertow" parameters. The reverse note can stop short of the sample attack by a given amount (wave distance) and then change directions, moving forward for a specified time (undertow). The wave distance softens the crest of the reverse wave, and the undertow acts as a kind of receding glow.

These preparations, although intentionally specific and limited, afford a remarkable range of sound design. Their envelopes can be set in various ways, and the tuning (which we will discuss shortly) can be applied uniformly across all preparations, or individually. But, most importantly, these are by definition interactive algorithms; they are not "effects" that are applied to a preexisting sound or gesture, but rather machines with which we negotiate musically, in real time, through the remarkably powerful interface of the keyboard. We play with them, and they shape us as much as we shape them—as Roger Moseley (2016, p. 271) writes, "All playing is a being-played."

The Tempo and Tuning Preparations

In the spirit of being object-oriented, and imagining multiple Synchronic objects sharing a single Tempo object (something that was impossible with the original bitKlavier) and possibly other future preparation types that might need a tempo, we decided to break tempo out into its own preparation. The Tempo preparation is just that: a variable that sets the tempo for whatever it is attached to. Of course, this can be modified in real time with a Modification, so even this basic parameter can become dynamic. But bitKlavier has become a site that invites exploration of the relationship between musician and machine, and in addition to resetting the phase of the metronome, we are interested in resetting the tempo of the metronome in various ways. At this stage, our explorations in following tempo and meter are rudimentary: We compute tempo from a

Figure 15. Basic adaptive tempo using a moving average of the last four notes (History slider), constrained by minimum and maximum interonset intervals (Min/Max slider). Alternatively, the tempo could be set by a moving average of note durations; this is an option in the Mode popup menu.



Tempo = 138.568   Period Multiplier = 0.866

constrained moving average of interonset-intervals (see Figure 15). But even this basic functionality has proven to have musical value. We are currently looking into more sophisticated techniques for following tempo and meter, including simple extensions like using an exponentially weighted moving average, as well as taking advantage of recent work in music information retrieval and machine learning.

The Tuning preparation is similarly a site for exploration of musician–machine relationships. Tuning and temperament is a historically rich—and fraught—subject, especially with regard to the keyboard and its requirement that we choose discrete, fixed pitches for each key (Barbour 2004; Duffin 2008). With digital instruments, we have the flexibility to change tuning quickly, from piece to piece, or within pieces, but surprisingly little has been done to facilitate this. In the Tuning preparation, we can liberate tuning in a number of ways, as shown in Figure 16.

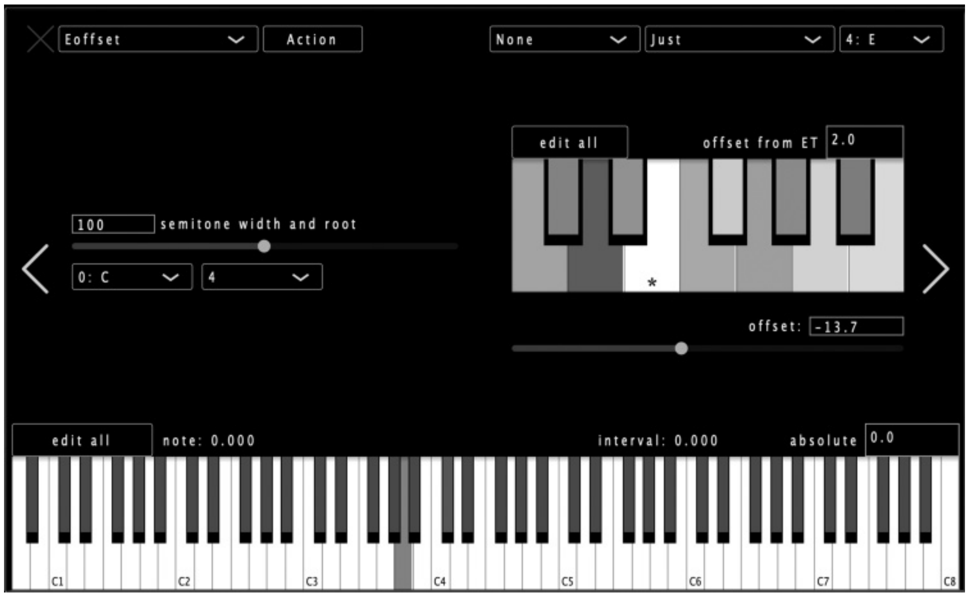
The basic tuning tools allow the user to choose from a large set of tunings or create their own (is in use in Figure 16) and to set the tuning's fundamental. (In Figure 16 one of a number of preset just tunings with the fundamental set to E is in use.) The entire tuning can be offset and individual keys can be set on the bottom keyboard to have their own offsets from the tuning system. (In Figure 16 the tuning is 13.7 cents flat, so that C matches concert tuning, and the A3 has a tuning offset, set in the bottom keyboard.) To the left we have a simple but powerful way to make the whole keyboard microtonal by setting the width of the semitone. All of these parameters can be changed on the fly with Modifications, enabling the creation of composition-specific dynamic tunings.

A rich array of tools for exploring tuning are incorporated into bitKlavier, including composed and adaptive systems. These are beyond the scope of this article but are described elsewhere (cf. the companion article in this issue of *CMJ*, Trueman et al. 2020).

An Example

The opening segment of Trueman's Etude No. 5: "Wallumrød" from *Nostalgic Synchronic* illustrates many of the core bitKlavier behaviors (see Figure 17). The opening chord launches a regular fading pulse (triangular noteheads in the bottommost staff of the

Figure 16. Tuning preparation window. The keys on both keyboards are sliders that allow for tuning of a temperament (the 12-key keyboard) or of individual notes on the 88-key keyboard. The intonation settings can also be edited in text, to facilitate copying to or from other systems. The roots of the temperaments can be set (top-right popup), as can the width of the semitone (a width of 50 would turn bitKlavier into a quarter-tone keyboard, for instance). The "None" popup includes options for three different adaptive tuning systems (for details, see Trueman et al. 2020).



first system), along with nostalgic swells (dashed crescendo markings) set to peak a measure after they are launched. Additionally, the fundamental of the just-intonation system (Doty 1993) is set to change via notes in the left hand; slashes through these notes tell the player to play these just slightly (even inaudibly) before the rest of the notes of the chord, so the sonority is tuned as desired (beginning in measure 5).

We can see these modifications in the construction site of bitKlavier (see Figure 18), where triangles attached to the one Tuning object set the tuning for all the objects in this piano; the Keymaps attached to these triangles have only the slashed notes of Figure 17 activated, so they in turn change the tuning fundamental. We can also see the Piano Switch preparation, which occurs at letter A, as triggered by middle C (with a slash through it, again indicating to the player to play that note first).

We explore more examples along with their implications for performance practices in the companion article (Sliwinski 2020) to this one. We have also been developing a number of adaptive tuning systems, which are detailed in the companion article in this issue of *CMJ*, "Intonation with a Software-based Piano," (Trueman et al. 2020).

Closing: On Preparing the Digital Piano

Cage implored preparers to put themselves in a "conducive frame of mind to overcoming obstacles with patience" (Cage 1946–1948), and although preparing the digital piano can at times require endless hours, these hours are some combination of composition and exploration time; once a set of preparations has been created, it can be saved and recalled in an instant, making their re-creation unnecessary. That said, personalizing and refining preparations is something in which even those just wanting to play existing repertoire for the instrument can engage; the preparations that Trueman creates for his work, for instance, are regarded as starting points—explanatory, but not definitive.

For the composer, preparing the instrument can be part of a rich compositional practice, and the instrument's preparation can indeed be so specific as to really define a compositional space. Prepared instruments are nearly "compositions" in their own right, and bitKlavier becomes a kind of "composed instrument," where anyone who touches it is in some ways playing the piece by the preparer. Note that we view the notion of "composed instrument" somewhat differently than has been described

Figure 17. An example utilizing many of the core features of bitKlavier: Trueman's Etude No. 5, "Wallumrød" from Nostalgic Synchronic. *(A recording of the entire movement is given in Audio Example 3, and the score is also online.)*

Figure 18. The preparation window for Trueman's Etude No. 5, "Wallumrød," used for the score and performance noted in Figure 17.

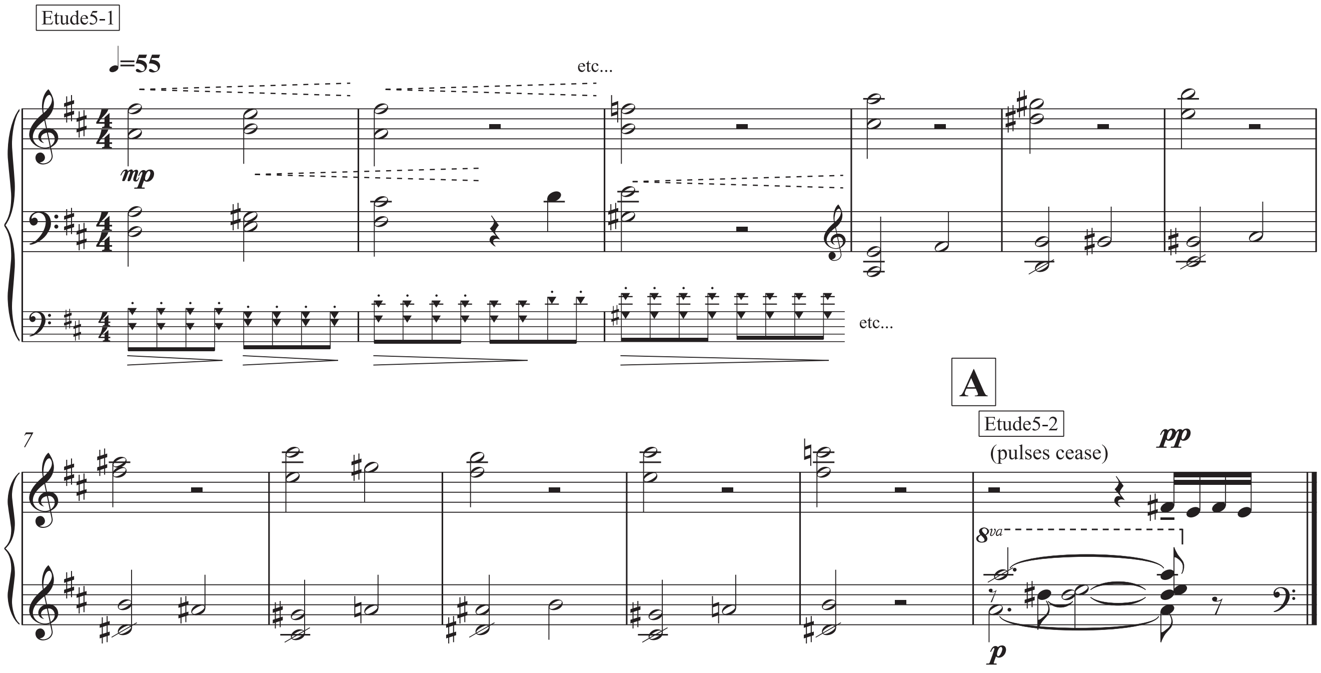


Figure 17

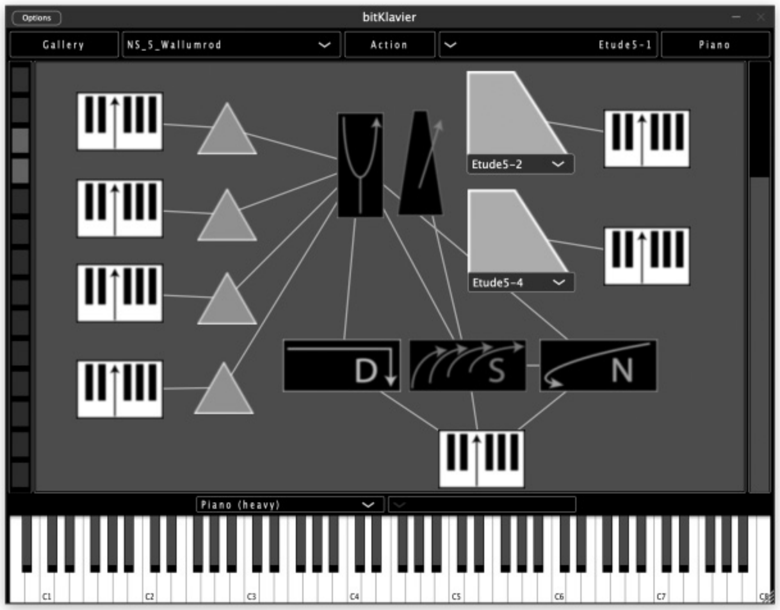


Figure 18

elsewhere (cf. Schnell and Battier 2002; Murray-Browne et. al. 2011). We prefer the more general idea of an instrument that has been created or modified so that any performance with it sounds close to a specific composition; indeed, for listeners familiar with Cage's *Sonatas and Interludes*, hearing even an improvisation on a similarly prepared piano will evoke Cage's work.

In her article "Toward a Musicology of Interfaces" musicologist Emily Dolan muses:

> Instead of thinking about music as a genus in which keyboards exist as a species, perhaps it could be productive to think of the keyboard—here standing in for all immaculately controllable instruments—as the genus, while this thing we have come to know as music as a species of keyboard (Dolan 2012, p. 12).

This remarkable inversion implies that we imagine music through our instruments, rather than creating instruments from our musical imagination, and is more than merely a provocation: Would Bach's *Well-Tempered Clavier* even be conceivable without the keyboard? The keyboard interfaces our body with sound and, in so doing, poses deep and fundamentally unsolvable questions. Further, the keyboard can interface our bodies with machines binding together digital and analog, all the while taking advantage of the embodied capabilities that musical training provides:

> The keyboard's persistence as an interface, its patterning of fixity and flexibility that has at once resisted and accommodated change, forms a shifting boundary that connects and separates worlds, joining and cleaving human and machine, player and played, the analog and the digital (Moseley 2016, p. 117).

This recalls the notion of the "centaur," where human and machine combine symbiotically to achieve behaviors impossible for them independently. The chess master Gary Kasparov realized that playing computer-assisted chess was "like learning to be a race car driver: He had to learn how to drive the computer" (Thompson 2013, p. 4). For musicians and bitKlavier, the steering wheel of the race car is analogous to the keyboard: an interface that enables a kind of real-time collaboration between human and machine.

We view bitKlavier as a tool for exploring this interface, this "shifting boundary." It is an instrument for imagining and making music through these explorations, a platform for reflectively revising and expanding its own design and capabilities, and, ultimately, it is a collaborator that inspires play, both in how it is played and played with, and how it plays us.

As Ge Wang wrote, "design for play and delight" (Wang 2018, p. 73).

Acknowledgments

We are grateful to Adam Sliwinski for his insights and dedication to the development of bitKlavier, and to Rebecca Sutton Koeser for her close read of the manuscript and her suggestions. Support for bitKlavier has come from the American Council of Learned Societies, the Center for the Digital Humanities, the Council on Science and Technology, and the Music Department at Princeton University.

References

Alperson, P. 2008. "The Instrumentality of Music." *Journal of Aesthetics and Art Criticism* 66(1):37–51.

Barbour, J. M. 2004. *Tuning and Temperament: A Historical Survey*. North Chelmsford, Massachusetts: Courier.

Born, G. 2005. "On Musical Mediation: Ontology, Technology and Creativity." *Twentieth-Century Music* 2(1):7–36.

Bunger, R. 1973. *The Well-Prepared Piano*. Colorado Springs: The Colorado College Music Press.

Cage, J. 1946–1948. *Sonatas and Interludes*. Leipzig: Peters.

Cage, J. 1951. Booklet text for "John Cage: *Sonatas and Interludes*." Composers Recordings CRI 700. (Reissued 1995 and 2007.)

Collins, N., et al. 2003. "Live Coding in Laptop Performance." *Organised Sound* 8(3):321–330.

Cook, P. R. 2001. "Principles for Designing Computer Music Controllers." In *Proceedings of the CHI Workshop on New Interfaces for Musical Expression*, pp. 3–6.

De Souza, J. 2017. “Voluntary Self-Sabotage.” From *Music at Hand: Instruments, Bodies, and Cognition*. Available online at https://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780190271114.001.0001/acprof-9780190271114. Last accessed March 2020.

Dolan, E. 2012. “Toward a Musicology of Interfaces.” *Keyboard Perspectives* 5:1–13.

Doty, D. 1993. *The Just Intonation Primer*. San Francisco: Just Intonation Network.

Dourish, P. 2004. *Where the Action Is: The Foundations of Embodied Interaction*. Cambridge, Massachusetts: MIT Press.

Duffin, R. W. 2008. *How Equal Temperament Ruined Harmony (and Why You Should Care)*. New York: Norton.

Fishman, N., and D. Trueman. 2018. “BitKlavier: Manual and Documentation.” Available online at manyarrowsmusic.com/bitKlavier/bitKlavier_Manual.pdf. Accessed April 2019.

Forsén, S., et al. 2013. “Was Something Wrong with Beethoven’s Metronome?” *American Musicological Society* 60(9):1146–1155.

Hugill, A., and H. Yang. 2013. “The Creative Turn: New Challenges for Computing.” *International Journal of Creative Computing* 1(1):4–19.

Jordà, S. 2002. “FMOL: Toward User-Friendly, Sophisticated New Musical Instruments.” *Computer Music Journal* 26(3):23–39.

Maeda, J., and R. Burns. 2004. *Creative Code: Aesthetics and Computation*. London: Thames and Hudson.

Magnusson, T. 2009. “Of Epistemic Tools: Musical Instruments as Cognitive Extensions.” *Organised Sound* 14(2):168–176.

Magnusson, T. 2010. “Designing Constraints: Composing and Performing with Digital Musical Systems.” *Computer Music Journal* 34(4):62–73.

Mission. 2018. “How to Avoid Feature Creep for Your Startup MVP.” Available online at medium.com/the-mission/6ed4e649eacd. Accessed April 2019.

Moore, F. R. 1988. “The Dysfunctions of MIDI.” *Computer Music Journal* 12(1):19–28.

Moseley, R. 2016. *Keys to Play: Music as a Ludic Medium from Apollo to Nintendo.* Oakland: University of California Press.

Murray-Browne, T., et al. 2011. “The Medium Is the Message: Composing Instruments and Performing Mappings.” In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 56–59.

Puckette, M. 1991. “Combining Event and Signal Processing in the MAX Graphical Programming Environment.” *Computer Music Journal* 15(3):41–49.

Puckette, M., T. Apel, and D. Zicarelli. 1998. “Real-Time Audio Analysis Tools for Pd and MSP.” In *Proceedings of the International Computer Music Conference*, pp. 109–112

Schnell, N., and M. Battier 2002. “Introducing Composed Instruments: Technical and Musicological Implications.” In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 156–160.

Sliwinski, A. 2020. “The BitKlavier: A Performer’s Perspective.” *Adam Sliwinski* (blog), 10 February. Available online at https://adamsliwinski.blogspot.com/2020/02/the-bitklavier-performers-perspective.html. Accessed 11 February 2020.

Smallwood S., et al. 2008. “Composing for Laptop Orchestra.” *Computer Music Journal* 32(1):9–25.

Théberge, P. 2017. “Musical Instruments as Assemblages.” In T. Bovermann et al., eds. *Musical Instruments in the 21st Century*. Berlin: Springer, pp. 59–66.

Thompson, C. 2013. *Smarter Than You Think: How Technology Is Changing Our Minds for the Better.* New York: Penguin.

Trueman, D. 2007. “Why a Laptop Orchestra?” *Organised Sound* 12(2):171–179.

Trueman, D. 2011. “Digital Instrument Building and the Laptop Orchestra.” *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2010 Symposium*. Washington, D.C.: National Academies Press, pp. 52–59.

Trueman, D., et al. 2020. “Tuning Playfully: Composed and Adaptive Tunings in bitKlavier.” *Computer Music Journal* 43(2–3):67–88.

Wang, G., and P. Cook. 2003. “ChucK: A Concurrent, On-the-Fly, Audio Programming Language.” In *Proceedings of the International Computer Music Conference*, pp. 219–226.

Wang, G. 2018. *Artful Design: Technology in Search of the Sublime.* Stanford: Stanford University Press.

Wright, M., and A. Freed. 1997. “Open Sound Control: A New Protocol for Communicating with Sound Synthesizers.” In *Proceedings of the International Computer Music Conference*, pp. 101–104.