# Beholden to our tools: negotiating with technology while sketching digital instruments

Andrew McPherson
Centre for Digital Music
Queen Mary University of London
London, UK
a.mcpherson@qmul.ac.uk

Giacomo Lepri
Centre for Digital Music
Queen Mary University of London
London, UK
g.lepri@qmul.ac.uk

## ABSTRACT

Digital musical instrument design is often presented as an open-ended creative process in which technology is adopted and adapted to serve the musical will of the designer. The real-time music programming languages powering many new instruments often provide access to audio manipulation at a low level, theoretically allowing the creation of any sonic structure from primitive operations. As a result, designers may assume that these seemingly omnipotent tools are pliable vehicles for the expression of musical ideas. We present the outcomes of a compositional game in which sound designers were invited to create simple instruments using common sensors and the Pure Data programming language. We report on the patterns and structures that often emerged during the exercise, arguing that designers respond strongly to suggestions offered by the tools they use. We discuss the idea that current music programming languages may be as culturally loaded as the communities of practice that produce and use them. Instrument making is then best viewed as a protracted negotiation between designer and tools.

## Author Keywords

Design, music programming language, aesthetic influence, idiomaticity, Pure Data

## CCS Concepts

•**Applied computing** → **Sound and music computing;** Performing arts; •**Human-centered computing** → **HCI theory, concepts and models;**

## 1. INTRODUCTION

Most contemporary music programming languages are Turing complete,[1] meaning that they are theoretically capable of representing any possible sonic outcome. In contrast to the MIDI synthesisers of the 1980's and early 1990's where limited computational power resulted in constrained sound design palettes and relatively coarse granularity of control, modern digital tools allow easy manipulation of audio at the sample level. In principle such openness of expression ought to lead to rapidly diversifying musical ideas. Why, then, do we find so many recurrent patterns in NIME practice?

Teachers of musical interaction design will be familiar with the tendency of beginning students to create theremin-like instruments [7] or to use touch sensor boards to arrange arbitrary objects into rudimentary keyboards [6]. Attendees of NIME concerts will encounter a recognisable prevailing (though by no means universal) aesthetic, often including drones, textured noise or manipulated samples. If technology opens up the possibility to produce any imaginable sonic interaction, what explains this clustering? Do NIME works of 2020 sound similar to the interactive music pieces of the 1980's and 1990's? If not, to what extent is the change explained by cultural shifts versus changes in underlying technology?

This paper examines the ways in which digital musical systems are non-neutral mediators of creative thought. Although a tool may theoretically be capable of anything, it will still have certain idiomatic patterns, making some structures and concepts easier or more obvious to the designer than others [13, 16]. While the aesthetic influence of tools has long been discussed [11], this paper seeks a more specific accounting of the idiomatic patterns of particular tools commonly used in NIME design. Specifically, we report on a compositional game in which sound designers were asked to sketch simple instruments with common electronic sensors and the Pure Data (Pd) programming language [20]. By examining the recurrent patterns of the resulting instruments, we begin to reveal the latent influence of these tools. We conclude by arguing for a fuller accounting within NIME of the central role of tools in shaping our creative processes, suggesting that instrument design should be viewed as a process of negotiation between designer and tools.

## 2. TECHNOLOGY AND VALUES

Historian of technology Melvin Kranzberg famously wrote: "Technology is neither good nor bad; nor is it neutral" [9]. The role of technology as a non-neutral mediator of human perception and actual has been explored by many philosophers [8, 14, 23, 24]. Madeleine Akrich proposes that designers embed values in their technologies as "a script out of which the future history of the object will develop" [1]. These scripts may be obvious or subtle, and the user of the tool may not even be aware of their influence [11]. Thor Magnusson writes that "instruments are actors: they teach, adapt, explain, direct, suggest, entice. Instruments are impregnated with knowledge expressed as music theory ... they explain the world" [12]. Such directivity is equally true of music notation [13] and music programming languages [18].

McPherson and Tahiroğlu consider these influences through the lens of *idiomaticity*: "patterns of instruments or lan-

[1] Turing completeness is a construct of computability theory which means that a language can be used to simulate any Turing machine, or more informally, that it can be used to represent the same set of possible computations as every other Turing-complete language.



*NIME'20,* July 21-25, 2020, Royal Birmingham Conservatoire, Birmingham City University, Birmingham, United Kingdom.

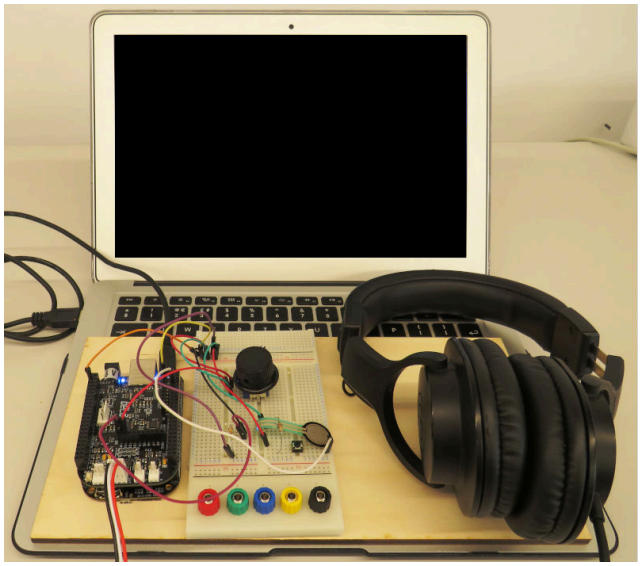

**Figure 1: Setup of the compositional game, one station per participant, facing one another. Each participant independently sketches simple instruments using common sensors, Bela and Pd.**

guages which are particularly easy or natural to execute in comparison to others" [16]. On traditional instruments, musical patterns which naturally fit the movements of the body often disproportionately appear in improvisation [5, 21]. We suggest that similarly, patterns which are idiomatic to a digital tool will also disproportionately appear in the objects and systems created with that tool. Importantly, we do not argue for technological determinism; the background and aesthetic outlook of the designer is also extremely important. In design fiction activities where no functional technology is involved, participants of different musical backgrounds express widely varied musical values through their artefacts [10]. Still, in the engagement with any tool, we should question what patterns they suggest, how we recognise them, and how to account for this influence in the design of our instruments.

Previous work [4] has considered digital musical instrument design as a form of *bricolage* practice, building on Vallgårda et al.'s account of interaction design: "the bricoleur does not plan ahead but develops the project in-situ with concerns of interaction, physical form, and behavior pattern not being hierarchically ordered a priori. Thus, the bricolage is the result of careful negotiations in the making" [22]. In this study, we aim to observe the patterns of in-situ exploration and negotiation.

# 3. INSTRUMENT DESIGN GAME

To explore the patterns that emerge through the influence of tools, we created a music-technological game involving the design of simple instruments. The study was conceived as a game in which music technology practitioners were invited to "compose" a simple instrument. Participants were given a breadboard containing 3 sensors commonly used in DMI design: a pushbutton, a potentiometer (with knob attached) and a force sensing resistor (FSR). The sensors were prewired to a Bela board [17], using a digital input for the button and (16-bit) analog inputs for the other two sensors. We provided our participants with headphones so that they could listen to their work without disturbing each other.

The activity involved making instruments using the Pure Data programming language. The workflow involved creating Pd patches on a computer and uploading them to the Bela board where they could be tested. As the focus of the activity was on sketching the code, participants used the sensors on the breadboard rather than considering how they might be integrated into physical objects.

## 3.1 Activity Structure

The compositional exercise involved two participants at a time. The activity took place in a quiet and isolated studio. Two independent workstations were prepared, one for each participant: a computer, a Bela with three connected sensors and a pair of headphones (Figure 1).

We introduced the study as a playful activity, explaining that we were not interested in testing their music programming skills but rather in observing the collaborative process. However, even if we did not aim to study collaboration, we used this pretense to both push away design pressure and compositional anxiety and emphasise the playful connotation of our study (i.e. there is no a right way to do it). The game was characterised by a fast pace: various short tasks quickly take over forcing the musician to start making without overthinking.

We borrowed this approach from the work of Andersen on Magic Machine workshop in which participants build "magical machines" starting with a "prompt" activity (in our case the collaboration excuse) and they are encouraged to "think with the hand" [3]. The activity was structure based on the following steps:

- **Step 1** (10 min) - the facilitator invites both participants to design a simple audio algorithm to be controlled by one of the sensor available sensors (drawn at random).
- **Step 2** (10 min) – participants are asked to sit on the other side of the table taking the place of the other person. They are then are asked to start a new Pd project and start a new sound algorithm to be controlled by one of the two remaining sensors available at the workstation (drawn at random).
- **Step 3** (10 min) - the participants swap places again, and the facilitator invites participants to start a new Pd project and work with the remaining sensor.
- **Step 4** (15 min) – after a last swap, participants are invited to gather in a new Pd patch the three algorithms saved in the workstation they currently are.

At this point each musician is able to play an instrument featuring the three sensors, each controlling different sound designs which have been alternatively developed by both players. The facilitator asks participants to review the overall code, eventually modifying / improving it according to either aesthetic or technical concerns. The game finishes with a "performance" in which participants briefly introduce the assembled instruments through a practical demonstration. It is important to notice that during the game, musicians are not able to listen to what the other is doing (due to the use of headphones). In the end, the participants come together in a semi-structured discussion of the musical and technical implications of their work, their design process and aesthetic choices.

We ran two different versions of the study, each following the same structure but with a difference in the first step. In the second version of the game, musicians randomly selected two sensors to start with. In the first step of second version of the exercise musicians could composed audio processes that could be controlled by two sensors (thus resulting in one fewer step overall).

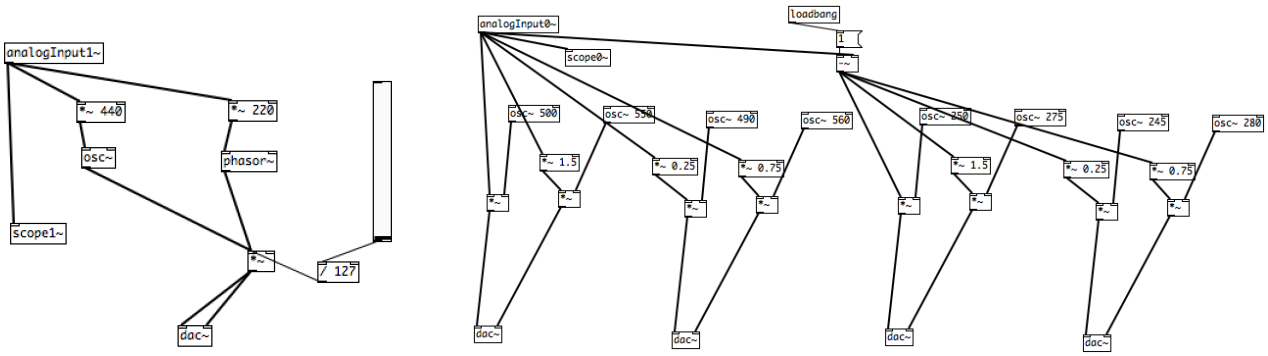


**Figure 2: Left: harmonically-related sine and sawtooth oscillators whose frequency is controlled by the potentiometer. Right: two clusters of 4 sine oscillators which are crossfaded with the FSR. Visual layout of the patches is participants' original wherever possible, with minor adjustments for space.**

## 3.2 Participants

The workshops involved 14 music technologists (7 pairs): 10M, 4F, aged 23-37. 4 pairs played the first version of the game (one sensor at a time) and 3 pairs played the second version (starting with 2 sensors). 8 participants identified as professional musicians and 6 as amateur musicians. All participants played an instrument (including 7 guitar/bass, 6 electronics/DAWs, 1 vocalist). 11 identified as active musicians for more than 15 years. 7 participants are currently active as live performers, in contexts ranging from classical to experimental electronic music; 7 are music producers, including electronic and film music.

13 of the 14 participants had formal training in sound design. 8 participants had formal training in Pd; of the remainder, 4 have beginner-level proficiency with Pd and 2 have used a similar language such as Max/MSP. Participants self-rated their Pd expertise on a 1-5 scale (5 being most proficient): level 1 (1 participant); 2 (2 participants); 3 (5 participants); 4 (5 participants); 5 (1 participant).

## 3.3 Data Collection and Analysis

At each step, the Pd patches were saved. Participants also completed a survey and interviews following the activity. For this paper, our analysis focuses on the Pd patches created in Steps 1 and 2 of the process, reflecting heir initial encounters with either one or two sensors prior to sharing and refining their ideas. For each patch, we annotated a brief summary of its function and analysed the use of common Pd objects and data processing structures. Given the short time frame of the activity and the basic materials, we expected the resulting Pd patches to be relatively simple. Our interest is in *what kind* of simple patches result, which can reveal idiomatic patterns of the language and sensors.

# 4. OUTCOMES

14 simple instruments were created in each of Steps 1 and 2 of the activity, for a total of 28 patches. The patches involved either one or two sensors each and showed a variety of design patterns. Considering number of objects as a proxy for patch complexity, the median number of Pd boxes per instrument was 14.5 (range 6-43) in Step 1 and 12.5 (range 5-30) in Step 2.[2] The sonic profile of the instruments tends toward continuous drones over intermittent or percussive events, especially when the potentiometer or FSR are used, with a tendency toward low-frequency sound sources.

## 4.1 Sound Sources

Only three Pd objects were responsible for the original sound in every case: `osc~` (sine wave oscillator); `phasor~` (simple 0-1 sawtooth oscillator with no antialiasing); and `noise~` (white noise). In Step 1, 10 patches used `osc~` as a sound source (range 1 to 3 copies), 7 used `phasor~` (range 1 to 6 copies) and 1 used `noise~`. 4 patches used more than one of these sources simultaneously. In Step 2, 6 patches used `osc~`, 5 used `phasor~` and 4 used `noise~`, with only 1 patch using two sources simultaneously. No patch in Steps 1 or 2 used samples, wavetables or any other sound sources.

Low-frequency oscillators (below 200Hz) were commonly used, especially with `phasor~`. 12 patches across the two steps used multiple oscillators (of either the same or different types) either summed into a chorus effect or multiplied together in a form of AM intermodulation. Only one of these patches used FM modulation.

Figure 2 shows two examples of the use of multiple oscillators. In Figure 2a, a sine and sawtooth are related by octaves and the frequency is controlled with the knob. This patch also contains some nonfunctional elements (such as the slider), which was fairly common in these short experimental exercises. In Figure 2b, the FSR crossfades between two clusters of 4 sine oscillators whose frequencies are fixed.

Filtering was used in 1 patch in Step 1 and 4 patches in Step 2. In 4 of 5 cases, the filter was used on a noise source; in the fifth case it was used on a 300Hz sawtooth. Filter types used included `hip~` (high pass), `lop~` (low pass), `vcf~` (band pass) and `bob~` (Moog nonlinear filter emulation).

Amplitude envelopes were occasionally used on the sound sources, most often in association with the button where the `line` object was used to extend the transitions between on and off states (Figure 3a). Another approach to enveloping is shown in Figure 3b where the knob controls the interval of high-frequency "pings" generated with the `vline~` object.

## 4.2 Use of Controls

In general, the controls had straightforward relationships to common audio parameters. The most common use of controls was as frequency adjustment. 9 patches in Step 1 and 7 patches in Step 2 included frequency control of either oscillators or filters. By contrast, amplitude control (either continuous or on/off) was included in 5 patches in Step 1 and 5 patches in Step 2. More uncommon controls included resetting the phase of a low-frequency oscillator, triggering envelopes, controlling the rate of a repetitive event, and controlling filter Q (1 patch each). The first of these (phase reset) is shown in Figure 4, where the designer comments that the output is designed to become increasingly annoying until the performer resets it with the button.

[2]This count included all boxes connected to the given sensor(s) including debugging objects, number and message boxes, but excluding any extraneous unconnected objects.

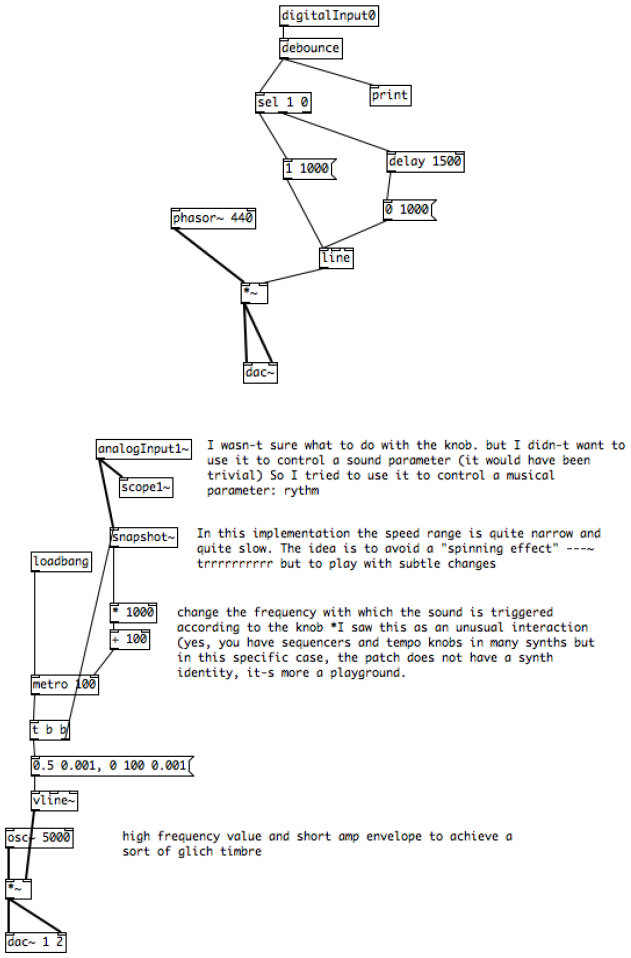


**Figure 3: Use of envelopes. Top: line segment provides gradual fade in and out with button press. Bottom: potentiometer controls rate of a repetitive envelope.**

**Potentiometer:** Of the 10 patches to use the potentiometer, 7 used it as a frequency adjustment, always as a continuous range and never as discrete steps. In 2 cases it was an amplitude control, and in the final case it controlled the rate of repetitive events.

**FSR:** Of the 12 patches to use the FSR, 6 used it to control frequency (an oscillator in 4 cases, a filter in 2 cases), 5 patches used it to control amplitude. One patch used it to control both frequency and amplitude. One further patch appears to be designed to control frequency although the input is wired to the incorrect input of the oscillator for this function. One patch compared the FSR value to a threshold and triggered a noise burst when the threshold was crossed.

**Button:** Of the 12 patches to use the button, 6 turn the sound on and off like a keyboard (though it serves an additional role in 2 of these patches). 1 further patch attempted to use the button to enable and disable all sound production although this did not work as intended. 3 patches used the button to select new random frequencies for one or more oscillators, 1 cycled through a series of harmonically-related frequencies, 1 reset the phase of a low-frequency oscillator, and 1 implemented an accumulator in which fast-paced button presses would slowly increase the Q of a filter.

### 4.2.1 *Transfer functions*

By convention the control data from each sensor is normalised between 0 and 1 at the input from Bela. With

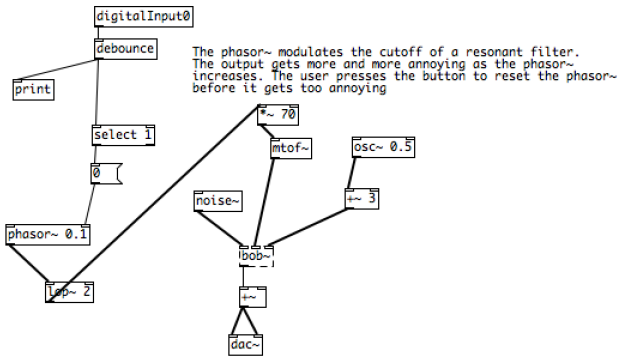


**Figure 4: Atypical use of controls: button resets a slowly-building process which produces an "annoying" sound.**

the exception of some amplitude controls, nearly all patches rescaled this range in some way. For the continuous sensors, linear relationships between sensor and sound parameter were by far the most common (12 out of 14 in Step 1, 9 out of 12 in Step 2, with 2 of these inverting the range). Only 4 patches across the two steps incorporated a logarithmic relationship, 3 of which used the `mtof~` (MIDI-to-frequency) object, even though both amplitude and frequency are typically perceived on logarithmic scales.

Nearly all patches used a memoryless control relationship where the current value of the sensor manipulated a current parameter value. Exceptions included a patch which had a slowly building sound reset by the button, a patch involving the accumulation of rapid button presses (both made by the same designer; Figure 4), one use of a delayed control signal in addition to its current value, and some use of line segments to taper the edges of on/off controls.

### 4.2.2 *Instruments with two sensors*

In Step 1, 6 of the 14 participants were assigned two sensors to work with while the remaining 8 were assigned only one. 5 of the 6 were randomly assigned the potentiometer and button. While perhaps the most conventional use of these two controls would be to control frequency with the potentiometer and use the button to control (on/off) amplitude, only one of the patches worked this way. One patch inverted this relationship by using the potentiometer to control amplitude and the button to select random frequencies; 3 others used both controls to affect the same parameter (both controlling frequency, or both controlling amplitude). The final two-sensor patch used the FSR and button; the FSR controlled frequency (with a sample-and-hold through the button) while the button activated the sound. This patch is shown in Figure 5.

## 4.3 Treatment of Pitch Material

23 of the 28 patches were based on audio-frequency oscillators. Every use of the FSR and potentiometer to control frequency was continuous; no patch quantised continuous sensor values to a musical scale. The `random` object was used in 5 patches, always to control frequency, usually updated with the button. One patch (Figure 5) stepped through harmonically-related frequencies using the button.

10 patches included more than one audio-frequency oscillator (excluding FM synthesis). In these cases, the harmonic language can be queried through the relationship between frequencies. In 2 cases, two oscillators had octave relationships to one another; in 1 case, three oscillators had a harmonic relationship (1x, 2x and 3x a base frequency).

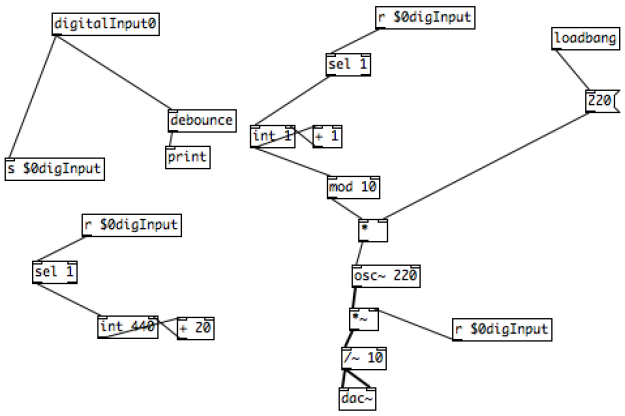


**Figure 5: The only patch involving pitch quantisation. Pressing the button activates the sound and steps through harmonic multiples of a 220Hz fundamental frequency.**

In 4 cases, the ratios between oscillators were complex (i.e. no simple harmonic relationship), including one cluster of 6 low-frequency oscillators 1Hz apart. The remaining 3 cases involved random frequencies. Overall, the treatment of pitch material shows an absence of any reference to musical scales (whether equal tempered or otherwise) though some reference to the harmonic series can be found.

# 5. DISCUSSION

## 5.1 Idiomatic Patterns of Pd

In the preceding analysis of the prevalence of different control and sound production strategies, it is important to emphasise that this was not a study of a mapping toolkit which permitted only a limited set of predefined relationship between sensors and sound. Rather, the Pd language could in principle express nearly any control relationship, but despite considerable variation in the finer details of sound design, we found that most control relationships fell into just a few categories of sensors manipulating fundamental sonic parameters (frequency, amplitude), usually in a linear, time-invariant, 1-to-1 manner. It is equally notable what we did *not* see in any of the instruments:

- Quantisation of pitch to a musical scale (other than one patch using the button to step through the harmonics of a 220Hz fundamental);
- Rhythmic patterns other than constant regular interval metronomes;
- Step sequencers or other pattern sequencing of control parameters (other than the above mentioned patch);
- Dynamic instantiation of synthesis processes, for example increasing or decreasing the number of oscillators on the fly.

The above patterns would be simple and achievable within 10 minutes in other languages. Our observed design outcomes align with the idiomatic patterns of dataflow languages like Pd and Max, as reported by the language creators in [16]. These languages are inspired by modular synthesis paradigms and patches often feature static signal processing graphs even as the data within the graph changes dynamically. Dynamic instantiation is not natural to these languages as it is to languages like SuperCollider or ChucK, and time is frequently an implicit rather than explicit quantity, perhaps explaining the lack of focus on rhythm.

In our ongoing research, we are repeating a similar exercise in SuperCollider, with the intention to identify differences in their idiomatic patterns. Our first pair of participants tended to design autonomous or ongoing sonic patterns that were not necessarily controlled with the sensors but rather influenced by them. Their focus was often on the digital side of the system, with the language seemingly privileged over the physical affordances inherent to the sensors. However, these observations might also relate to SuperCollider's different learning curve or that, compared to the visual immediacy of Pd, the first design steps might require more planning, especially with limited time.

## 5.2 Elemental Sounds

Although audio synthesis is manipulated at a much lower level in Pd and other languages than in MIDI-based equipment of previous decades, we also do not see the complexity of sound design that might have gone into those earlier tools. Instead, we see the fundamental elements of sound synthesis (sine and sawtooth oscillators, white noise, filters) appearing straightforwardly in the final output. While this partly reflects starting from a blank patch with limited time, sonically similar results can often be found in more developed NIME instruments, suggesting that the technology might be exerting an aesthetic influence.

Pd creator Miller Puckette writes that the Max/Pd family of languages "goes to great lengths to avoid a stylistic bias on the musician's output." [19]. In a 2019 email interview in [15], Puckette writes that an "important feature might be Pd's low-level primitives (phasor, but no band-limited sawtooth generator, for instance) - which is intended to encourage users to develop their own 'sound' rather than guide them." In practice, none of our participants used `phasor~` as a lookup to a wavetable, as it might have been intended, and all of them used it as a sawtooth despite the lack of band-limiting. This may show the influence of immediacy and convenience even where tools are intended to be assembled into more complex structures. We suggest that continuous pitch spaces, linear mappings and pure waveforms are just as stylistically loaded as any MIDI sequencer.

## 5.3 The Influence of Background

The 14 participants come from a variety of musical backgrounds, with widely varying experience in instrumental training and music technology. In previous work using a design fiction exercise [10], we found that a designer's musical values led to widely differing types of (non-functional) artefact. Thus the level of consistency across the simple instruments in this activity is notable. It is unlikely that each person's abstract musical intentions would lead them to simple oscillators with linear frequency controls; rather, we are seeing a strong influence from the technology.

It is beyond the scope of this paper to compare the resulting designs to the background of individual participants, but this would be interesting in future work. However, a perusal of new musical instruments on crowdfunding websites [15] and designs on YouTube from communities engaged with more popular music styles may show quite different sonic characteristics from NIME instruments even though these instruments often use some of the same digital technologies.

## 5.4 Limited Time, Increased Influence

Our musical game was deliberately time-limited and highly constrained in its physical materials. We do not propose that the instruments created reflect what any of the designers would do in a longer and more open-ended situation. It may be that certain kinds of sounds and control strategies are simply not achievable from first principles within

10 minutes. In fact, this underlines our thesis about the idiomatic patterns of the tools. In a language like SuperCollider, creating musical scales, rhythmic patterns, or dynamic numbers of sound sources might be implemented in seconds with a single line of code. That certain patterns recur in the work of several designers with varied musical backgrounds shows the aesthetic influence of the tools.

We do not endorse a technological-determinist viewpoint of instrument design. Although the tools contain hidden scripts and offer aesthetic suggestions [1, 11], personal background and aesthetic priorities still play a strong role in the outcomes. In a few cases in our exercise, designers explained to us that they chose a particular approach to deliberately avoid what they felt would be the most obvious thing to do with the technology. For example, one participant reported in a comment written within a patch: "I wasn't sure what to do with the knob. but I didn't want to use it to control a sound parameter (it would have been trivial) So I tried to use it to control a musical parameter: rhythm".

## 6. CONCLUSION

Designers sketching simple instruments using a limited palette of sensors with the Pd programming language approached the task with creativity and style. Within the diversity of individual outcomes we saw frequently recurrent patterns in musical language and the use of controls, which appear to reflect the idiomatic patterns of the tools.

We propose that instrument design is best viewed as a dialog or negotiation between designer and tool. In any negotiation, both parties give up something in order to gain something else that they want. In this case, the designer may temper some of their abstract ideas to create what is perceived to be feasible with the available tools, time and skills. Just as an improvising pianist may reach for chords that fit easily under the hand, the idiomatic patterns of the tool will suggest certain ideas which can then be accepted, rejected or modified by the designer. The influence of the language in turn puts the focus on the designer of that language, who is responsible for shaping its scripts and embedded values [1, 11, 16]. The tool creator may in turn be influenced by their own communities of practice and use of technology. In this way, the identity of any musical instrument emerges out of a process of recursive inscription from successive generations of musicians and technologists. Paraphrasing Günther Anders [2], the uses and purposes of a device are nothing more than the possibilities made available by the technology itself. Each tool generates its own purposes and scopes, and not the other way around.

On this basis, we encourage NIME tool and language creators to embrace, even accentuate, the non-neutrality of their tools. Memorable electronic instruments have often been those with notable limitations or eccentricities: consider the influence of the TR-808 on hip hop, or the Commodore 64 on chiptunes. However, just as the 1990's saw the creation of dozens of similar-sounding General MIDI sound banks, the current era features a proliferation of graphical DSP toolkits similar to Max or Pd, typically re-implementing the same canonical operations. Future designers who forego the search for an idealised all-powerful platform in favour of making deliberately limited or idiosyncratic tools could instigate exciting new creative negotiations.

## 7. ACKNOWLEDGMENTS

This work was funded by EPSRC under grants EP/N005112 (Design for Virtuosity) and EP/L01632X/1 (Centre for Doctoral Training in Media and Arts Technology).

## 8. REFERENCES

[1] M. Akrich. The de-scription of technical objects. In W. E. Bijker and J. Law, editors, *Shaping technology - Building Society: Studies in Sociotechnical Change*. MIT Press, 1992.

[2] G. Anders. *Die Antiquiertheit des Menschen: Über die Zerstörung des Lebens im Zeitalter der dritten industriellen Revolution*. 1956.

[3] K. Andersen and R. Wakkary. The magic machine workshops: making personal design knowledge. In *Proc. CHI*, 2019.

[4] J. Armitage and A. McPherson. Bricolage in a hybrid digital lutherie context: a workshop study. In *Proc. AudioMostly*, 2019.

[5] J. De Souza. *Music at hand: Instruments, bodies, and cognition*. Oxford University Press, 2017.

[6] E. I. Dolan. Toward a musicology of interfaces. *Keyboard Perspectives*, V:1–12, 2013.

[7] T. Igoe. Physical computing's greatest hits (and misses). Online: `https://www.tigoe.com/blog/category/physicalcomputing/176/`, 2008.

[8] D. Ihde. *Technology and the lifeworld: From garden to earth*. Indiana University Press, 1990.

[9] M. Kranzberg. Technology and history: "Kranzberg's laws". *Technology and culture*, 27(3):544–560, 1986.

[10] G. Lepri and A. McPherson. Making up instruments: Design fiction for value discovery in communities of musical practice. In *Proc. DIS*, 2019.

[11] T. Magnusson. Of epistemic tools: Musical instruments as cognitive extensions. *Organised Sound*, 14(2):168–176, 2009.

[12] T. Magnusson. Ergomimesis: towards a language describing instrumental transductions. In *Proc. ICLI*, 2018.

[13] T. Magnusson. *Sonic writing: technologies of material, symbolic, and signal inscriptions*. Bloomsbury Academic, 2019.

[14] M. McLuhan. *Understanding media: The extensions of man*. MIT press, 1994.

[15] A. McPherson, F. Morreale, and J. Harrison. Musical instruments for novices: Comparing nime, hci and crowdfunding approaches. In *New Directions in Music and Human-Computer Interaction*, pages 179–212. Springer, 2019.

[16] A. McPherson and K. Tahiroglu. Idiomatic patterns and aesthetic influence in computer music languages. *Organised Sound*, 25(1), 2020.

[17] A. McPherson and V. Zappi. An environment for Submillisecond-Latency audio and sensor processing on BeagleBone black. In *Proc. AES Convention*, 2015.

[18] C. Nash. The cognitive dimensions of music notations. In *Proc. TENOR*, 2015.

[19] M. Puckette. Max at seventeen. *Computer Music Journal*, 26(4):31–43, 2002.

[20] M. S. Puckette. Pure data. In *Proc. ICMC*, 1997.

[21] D. Sudnow. *Ways of the hand: The organization of improvised conduct*. MIT Press, 1993.

[22] A. Vallgårda and Y. Fernaeus. Interaction design as a bricolage practice. In *Proc. TEI*, 2015.

[23] Y. Van Den Eede. In between us: On the transparency and opacity of technological mediation. *Foundations of Science*, 16(2-3):139–159, 2011.

[24] P.-P. Verbeek. Beyond interaction: a short introduction to mediation theory. *interactions*, 22(3):26–31, 2015.