

Interfacing Sound: Visual Representation of Sound in Musical Software Instruments

Thor Magnusson

Abstract This chapter explores the role of visual representation of sound in music software. Software design often remediates older technologies, such as common music notation, the analogue tape, outboard studio equipment, as well as applying metaphors from acoustic and electric instruments. In that context, the aim here will be study particular modes in which abstract shapes, symbols and innovative notations can be applied in systems for composition and live performance. Considering the practically infinite possibilities of representation of sound in digital systems—both in terms of visual display and mapping of gestural controllers to sound—the concepts of graphic design, notation and performance will be discussed in relation to four systems created by the author: *ixi* software, *ixiQuarks*, *ixi lang*, and the *Threnoscope* live coding environment. These will be presented as examples of limited systems that frame the musician’s compositional thoughts providing a constrained palette of musical possibilities. What this software has in common is the integral use of visual elements in musical composition, equally as prescriptive and representative notation for musical processes. The chapter will present the development of musical software as a form of composition: it is an experimental activity that goes hand in hand with sound and music research, where the musician-programmer has to gain a formal understanding of diverse domains that before might have been tacit knowledge. The digital system’s requirements for abstractions of the source domain, specifications of material, and completeness of definitions are all features that inevitably require a very strong understanding of the source domain.

T. Magnusson (✉)
Department of Music, School of Media, Film and Music,
University of Sussex, Brighton, UK
e-mail: t.magnusson@sussex.ac.uk

1 Introduction

Musical instruments are amongst the earliest human technologies. Possibly preceding fire and weaponry, we could speculate how early humans used rocks or sticks to hit other materials in order to define territories, communicate, or synchronise movements. For a social animal like the human, music is clearly a sophisticated and multipurpose cohesion technique. Some of the oldest known musical instruments are flutes found in Germany (the *Hohle Fels* flute) and Slovenia (the *Divje Babe* flute), estimated to be between 35,000 and 42,000 years old. Preceding these flutes would be generations of forgotten instruments in the form of rocks and sticks that might not even “look like” musical instruments at all.

Today we talk about “music technology,” a two-word coinage that conjures up the image of plastic- or metal-surfaced equipment offering interaction through rotating knobs, sliders, or buttons, which are mapped to functionality represented on a screen of some sort. However, a quick etymology of the word “technology” clearly demonstrates that we are not discussing plastic gadgets here, but rather an embodied knowledge, skill or craft. The root of the Greek word *technē* is “wood,” but at the time of the early philosophers, it had begun to denote the craft of producing something out of something else. For Aristotle, *technē* (τέχνη) is an activity where the “origin is in the maker and not in the thing made” (Ackrill 1987, p. 419). In *Rhetorics*, Aristotle uses the word “technology” to signify the “craft of the word” (*technē* and *logos*) as used in grammar or rhetoric, which is an inverse meaning to the later use signifying the knowledge (*logy*) of craft (*technē*). The word is not used much until the 17th century, which is when it enters the English language (Mitcham 1994, p. 130). At no point did the word signify objects, but rather the skill of doing things, as evidenced in Marx’s *Das Kapital*: “Technology discloses man’s mode of dealing with Nature, the process of production by which he sustains his life, and thereby also lays bare the mode of formation of his social relations, and of the mental conceptions that flow from them” (Marx 2007, p. 402). Earlier in the same paragraph we read: “Darwin has interested us in the history of Nature’s Technology,” and it is clear that he means the ways nature goes about its business. Bernard Stiegler defines technology as “the discourse describing and explaining the evolution of specialised procedures and techniques, arts and trades” (Stiegler 1998, p. 94) and encourages us to use the word in the manner we apply the words “psychology” or “sociology”.

We *do* music technology: we don’t buy it, own it, or use it. Thinking, designing, discussing, performing and composing are all acts of music technological nature. Musical instruments are the tools of music technology and represent the musico-theoretical framework of the specific culture. However, let’s not forget the Greek origins, where the technology was about shaping something out of something else: in contemporary music technological practice, we are applying hardware, code libraries, communication protocols, and standards that become the material substance of our design explorations. We are working with *designed materials*, not wood or skin, but entities that already are of an epistemic nature

(Magnusson 2009). The new materials are semiotic in that they are part of a complex organisation of protocols and standards, which are needed for the diverse code libraries and hardware to be applied in the complex ecosystem of wired and wireless inter-software and inter-hardware communication.

When we create digital instruments we operate like a Latourian ant: busily operating as a part of a larger whole, applying actor-networks consisting of other actor-networks, or, in short, inventions that have become *blackboxed* in other technological processes, to the degree that we lose the possibility of grasping any origins. Where would a technological object originate from anyway? For this reason the instrument often appears before we know its expressive scope or indeed rationale (how, why, where, etc.).¹ The history of the saxophone provides a good example of how undefined the role of a new instrument can be, slowly gaining diverse functions amongst different musical cultures. In this context it is interesting to behold Attali's statement that in "music, the instrument often predates the expression it authorizes, which explains why a new invention has the nature of noise; a 'realized theory' (Lyotard), it contributes, through the possibilities it offers, to the birth of a new music, a renewed syntax" (Attali 1985, p. 35).

2 Digital Music Technologies—Designing with Metaphors

Music is many things to many people. If we were to attempt at a general definition, one approach might divide music into two key categories: in the first, music is performed, where an instrumentalist, or a group of them, engage in an act of generating sound, either from a score, from memory, or by improvisation. The context of co-players, the audience, and the location plays an important role here, where the liveness yields a sense of risk, excitement and a general experience of the moment's uniqueness and unrepeatability. The second category is music as stored: in people's memory, as written notation, on disks, tapes, or digital formats. The music could even be stored as an undefined structure in the form of algorithmic code for computer language interpreters. Now, in the 21st century, things are a little more complicated. New developments in digital music technologies transcend the above categories, deriving their symbolic design equally from the world of acoustic instruments, performance, notation, and electronic technologies. These new technologies further complicate the relationships between instrument makers, composers, performers, and the audience. Who is what? And the work itself ... is it an instrument? A compositional system? A piece?

¹The 160 character text message is a good example: the SMS (Short Message Service), although invented as part of the GSM cooperation in 1984, was initially implemented in Nokia phones for their engineers to test mobile networks. The technology was quickly adopted by users who began enjoying this mode of communication. This became a protocol of sorts, and as of 2016, Twitter is still respecting this 140 char limit.

There is a real sense that the technologies of music making are undergoing a drastic change by the transduction into the digital domain. This can be explored by studying the divergent natures of acoustic vs. digital instruments. The sound of a traditional musical instrument is necessarily dependent on acoustics, or the physical properties of the materials it is built of. Instrument makers are masters of materiality, with sophisticated knowledge of material properties and how sonic waves traverse in and through diverse types of matter, such as wood, metal, strings, or skin membranes. The instrumental functions of an acoustic instrument are necessarily determined by millennia old practices of material design. This is clearly not the case with digital instruments, where any interface element can be mapped to any sound. The mappings are arbitrary, and can be intelligent, non-linear, non-determined, inverse, open, and more. The design of digital interfaces ranges from being directly skeumorphic² and functional to more abstract and representational. In either case, every interface element *signifies* a function resulting in a sound rather than directly *causing* a sound. With the mapping function inserted between a gesture and the sound, the interface becomes semiotic: with this arbitrary relation, the digital instrument begins to surpass the acoustic as an epistemic entity, and at times manifests as a vehicle of a music theory or even a score.

The idea of making music with computers has existed since they were invented, and we can boldly claim that computers are the ideal media for composing, performing, storing and disseminating musical work. A quick tracing of this symbiotic relationship takes us back to early computers, with Ada Lovelace speculating about the musical potential of Babbage's Analytical Engine in 1842 (Roads 1996, p. 822). In the early days of electronic computers, we find Lejaren Hiller and Leonard Isaacson applying Markov chains in 1957 for one of the first algorithmically composed pieces, the *Illiac suite*, and Max Matthews inventing notation languages for computer generated synthetic sound. However, if we look at the history of mass produced digital musical instruments and software, we see that the computers have been used primarily as bespoke microchips integrated in instruments, for example in a synthesizer or an electronic drum kit, where the hardware design has been primarily mimetic, aiming at imitating acoustic instruments.³ In the case of music software, we are faced with multiple imitations of scores, piano rolls, magnetic tape, where the key focus has been on developing tools for the composition and production of linear music at the cost of live performance. From both business and engineering perspectives it is evident that hardware manufacturers benefited from a model where new synthesis algorithms were embedded in redesigned computer chips, and sold as new hardware.⁴ Software developers in turn addressed another

²Skeumorphic design is where necessary features in an original objects are used as ornamentation in the derivative object. Examples in graphical user interface design could be screws in screen-based instruments, leather in calendar software, the use of shadows, and so on.

³The contrasting design ideologies between Moog and Buchla are a good example of the problems at play here. It is evident that Moog's relative commercial success over Buchla's was largely due to the referencing well known historical instruments (see Pinch and Trocco 2002).

⁴There are exceptions of that model of course, such as the discontinued Nord Modular Synth.

market, applying the “studio in your bedroom” sales mantra, which sparked the imagination of a generation in the late 80s, who used Cubase on Atari computers, starting a genealogical lineage that can be traced to the current Logic or Live digital audio workstations.

Specialists in innovation studies, marketing, science and technology studies, and musicology, could explain in much more detail how technologies gain reception in culture, the social and economical conditions that shape their evolution, and the musical trends that support the development of particular technologies. From the perspective of an inventor, it is less obvious why the history of musical technologies has developed this way, although inventions ultimately have to depend on market forces in order to enter public consciousness. Here, the history of failures is as, if not more, interesting as the history of successes. (“failure” is here defined in the terms of the market, economy and sales). One such “failed” project could be Andy Hunt’s MidiGrid, a wonderful live improvising software for MIDI instruments written in the late 80s (Hunt 2003). An innovative system, ahead of its time, the focus was on performance, liveness and real-time manipulation of musical data. Written for the Atari, Hunt received some interest from Steinberg (a major software house), which, at the time, was working on the Cubase sequencing software. Only an alternative history of parallel worlds could speculate how music technologies had evolved if one of the main music software producers would be shipping two key software products: one for performance and the other for composition.⁵ At the time of writing certain digital interfaces are being produced that are not necessarily imitating the acoustic, although inspired by them. It is yet to be seen whether instruments such as the *Eigenharp* and the *Karlax*⁶ will gain the longevity required to establish a musical culture around the technology of composing and performing with them.

Since the early 2000s, developments in open source software and hardware have altered this picture. The user has become developer, and through software such as Pure Data, SuperCollider, CSound, Max, ChuckK, JavaScript, and hardware such as Arduino and Raspberry Pi, a world has opened up for the creation of new music technologies. The ease of access and low cost of these technologies, together with strong online communities that are helpful and encouraging, make such DIY approaches fun, creative and rewarding. When music software has become sophisticated to the degree that it can almost compose the music without the input

⁵Hunt’s software is of course no failure. It is a highly successful research project that has served its author and many others as musical tool, for example in education, and it has inspired various other research projects, mine included. But the context of this discussion is innovation and how a specific music technology instance might fare in the world of mass markets and sales.

⁶The manufacturers of both interfaces call them “instruments”. Some might argue that they only become instruments when coupled with a sound engine, as familiar instrumental models indicate (e.g., Wanderley 2000 or Leman 2008), but I do believe it makes sense, in terms of innovation, longevity and spread of use, to call these instruments. Will there be a day when something like the Karlax will be taught in music conservatories? How would that even work? What would the training consist in?

of the user (who becomes a “curator” of samples or a “wiggler” of knobs and buttons), many find that real creative approaches happen when music technology itself is questioned and redefined. Gordon Mumma’s ideas of “composing instruments” (see also Schnell and Battier 2002) are relevant here.

This chapter describes such questioning of music technology. Here the investigation regards interface and interaction design, i.e., how the visual elements in music software can affect musical ideas, composition and performance. Considering the practically infinite possibilities of representation of sound in digital systems—both in terms of visual display and mapping of gestural controllers to sound—the process of designing constraints will be discussed in relation to four systems developed by the author that engage with visual representation of sound in music software.

3 Interfacing Sound with Screen Interfaces

Interfacing sound in screen-based music software is no simple task: traditionally the software tends to either follow linear scoring metaphors (piano rolls, traditional notation, tape tracks) that are useful for composition, or imitate hardware (sliders, knobs, buttons, cables, screens), allowing for a real-time manipulation of sound which is eventually “bounced down” to a fixed file. There have been myriads of other, more experimental approaches, that investigate how we can perform with screen-based musical interfaces. However, designing two- or three-dimensional representation of sound, where the physical interfaces might consist of a mouse, keyboard or touch screens, comes with some complications. Some of the design patterns that we find in the material world cannot easily be abstracted and represented in the digital domain. Such translations often become a process of *transduction*, where sounds or actions are transformed in the digital. Even when we attempt mimesis and aim to be true to the original object, we lose some of the unique (non-universal) characters of the individual instrument, the entropic qualities that often manifest in its behaviour, as well as the history and use of the particular object itself. A copy of software does not have a history in the same way as an individual object.

On the other hand, elements not found in acoustic instruments present themselves as natural properties within the digital, for example the possibility of timelines, looping techniques, learning mechanisms, or diverse mechanics of mapping gesture to sound. Here, screen-based instrument designers apply techniques from computer games, interface design, HCI, apps, installations, and computer networks. The metaphors abound, but they can be found in diverse areas of development, where techniques and user skills are *reused* in the new design. Commensurate with how the maker of hardware musical instruments seeks to enact the skills developed and incorporated into the motor memory of instrumentalists over the years, the designer of screen-based interfaces will apply techniques from diverse fields, such as drag and drop, shift-click for multiple selection, swipe for new screens, right

click for menus, etc. Blackboxed design patterns are applied in the design of new instruments, consciously or not, and the user intuitively performs the system, learning its conceptual nature through interaction design that is already familiar.

3.1 *ixi Software*

The dozen or so applications that Enrike Hurtado and myself developed around 2000, and uploaded onto our *ixi audio* website (ixi 2000), were all experiments in sonic interaction design (Magnusson 2006). We wanted to create non-representational graphics that would control sound, through both real-time interaction (using the mouse and keyboard) and automation. The user would create visual objects that had associated sound which could be manipulated by moving them around, changing the shape, connecting them with other objects, and so on. We explored diverse design patterns, each represented by a small “app” (as we called the software—this was before the days of mobile media). Examples of the mapping of visuals to sound include: size for amplitude, vertical location for pitch, horizontal location for panning, shape or colour for timbre, blinking or rotating for automatic triggering of sounds, movement as a type of panning (perhaps a moving microphone that randomly navigates a space of sound), and so on (Fig. 1).

This software is now “abandonware,” as we have no time to translate it to new operating systems, indeed it is a good example of how transient digital systems for musical production can be. However, what *is* of lasting value are the ideas developed, the use of metaphors, the interaction design, the idea of automation, computational creativity, and real-time playfulness. These ideas become a design language, a set of interface and interaction patterns that are learned, embodied, and easily implemented in new software. Clearly not unique to *ixi* software, they are design discoveries, often of personal—as opposed to historical—nature,⁷ that have been reapplied in later software by us, and, indeed, inspired other software.

3.2 *ixiQuarks*

The *ixiQuarks* (Magnusson 2007) continued the research of *ixi* software, but here with a more coherent research agenda. They were developed in SuperCollider between 2004 and 2006 as an investigation into alternative screen-based interfaces, where non-linearity, performativity and real-time control of sound were the key design considerations. Discarding common concepts like timelines or linear

⁷See Boden (1990) on creativity - although her P-creativity and H-creativity stand for psychological and historical creativity (where the former is always included in the latter), in this case we use the term personal creativity.

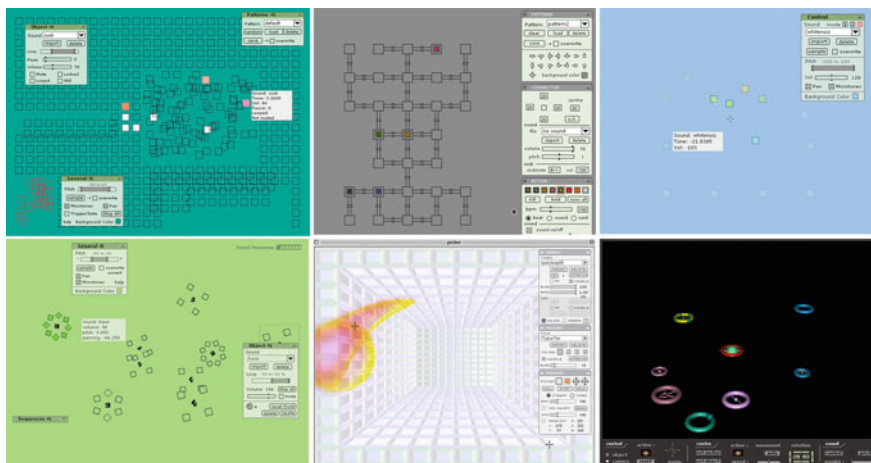


Fig. 1 A screenshot of six individual ixi software applications. Each of them served as an investigation into a different mode of interactive design

notation, most of the instruments developed as part of the ixiQuarks package were aimed at direct control of sound, where the mouse, the keyboard, pressure tablets and other cheap and common interface devices are used for control.

At the time of development, there were no multi-touch screens or trackpads, which resulted in more limited design decisions. However, it is not clear that multi-touch would be of a drastic benefit here since the lack of tangible interface elements makes the instrument less embodied and the user focus becomes more conscious on particular visual elements (think selecting a bespoke element on the screen with the mouse arrow). Furthermore, touching a visual element with your finger on a screen hides it (under the finger), prevents overlapping elements (as fingers can't be in the same space at the same time), and the anatomy of the hand also provides some expressive limitations. These would be interesting constraints to design around, but they simply didn't exist at the time (Fig. 2).

The ixiQuarks interfaces are non-representational, or, at least, they do not primarily derive their metaphors from physical instruments or music technological hardware. The interface and interaction metaphors were rather influenced by traditional HCI, computer games and web design. Creative audiovisual coding was a much more inspirational context than acoustic or digital music technologies. A central question to be explored was how visual interfaces and alternative interaction design would result in different music. We were equally interested in how the design itself inspired musicians, and also how the limitations of expressive actions would provide affordances and delineate constraints that would be navigated by users through a process of exploration (Magnusson 2010). For this reason, there were no manuals written, no demo videos created, no sound banks provided.

An element of ixiQuarks was that the user would be able to redefine the sound, create new sounds and change the function of the interfaces during performance.

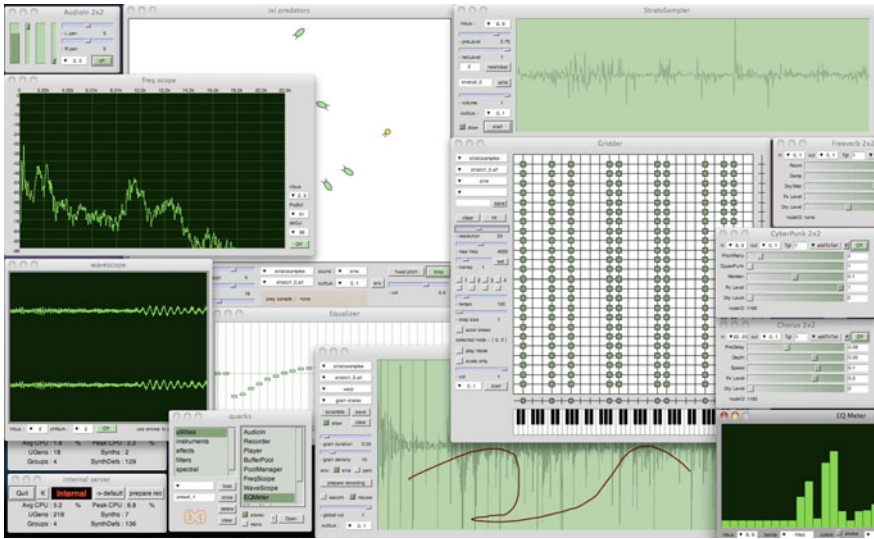


Fig. 2 A screenshot of ixiQuarks. Each of the instruments and widgets are independent from the other but work well together. The sounds from one can be used as input into another. Some of the instruments can be live coded and changed in real-time

This was an early version of live coding, where the interface could be altered in real-time. As a performer in improv ensembles I became more interested in the live coded aspects of musical performance, sometimes only using the graphical interfaces to trigger patterns, whilst changing the SuperCollider synth definitions. For this reason I decided, in 2009, to attempt at creating a live coding system that would continue some of the explorations of the early ixi work, but here through the use of language or code as opposed to graphical design.

3.3 *ixi Lang*

SuperCollider is an ideal platform for live coding. It is a real-time system where synths can be created and stopped without affecting other running synths, their parameters changed, and musical patterns can be written to control the synths. This is ideal for live exploration of sound synthesis and electronic music. Indeed, rumour has it that the term “live coding” was first used on the SuperCollider mailing list by Fabrice Mogini when describing his compositional process, sometime in the late 20th century. However, when performing live, speed and simplicity of syntax becomes important, as the performer ideally wants to be focusing on the music and not the code. With *ixi lang* some of the main design goals were: to create a simple, fast, and forgiving live coding system with a syntax that makes mistakes unlikely (no commas, brackets, or semicolons); a language that was easy to learn, and

```

oo -> morph|S B |
obb -> |o o |
cbb -> |o xxo o |

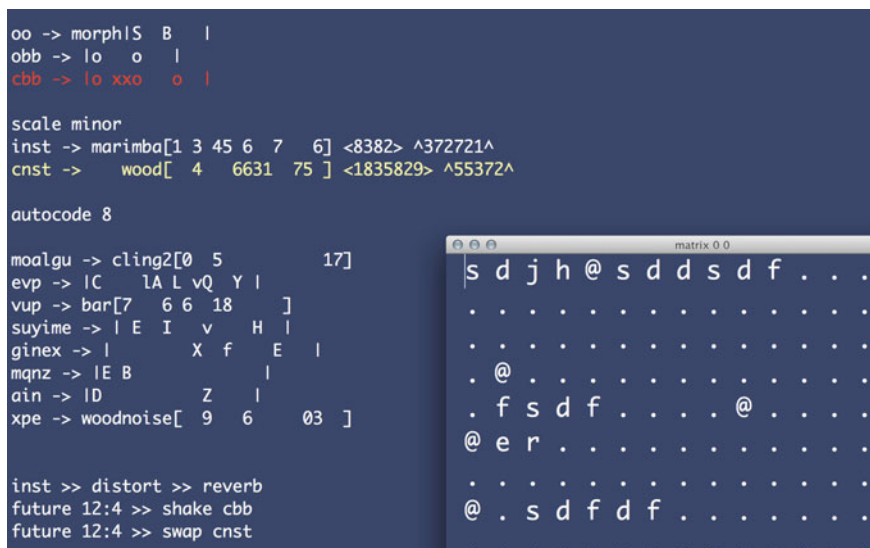
scale minor
inst -> marimba[1 3 45 6 7 6] <8382> ^372721^
cnst -> wood[ 4 6631 75 ] <1835829> ^55372^

autocode 8

moalgu -> cling2[0 5 17]
evp -> |C |A L vQ Y |
vup -> bar[7 6 6 18 ]
suyime -> | E I v H |
ginex -> | X f E |
mqnz -> |E B |
ain -> |D Z |
xpe -> woodnoise[ 9 6 03 ]

inst >> distort >> reverb
future 12:4 >> shake cbb
future 12:4 >> swap cnst

```



The screenshot shows a terminal window on the left with code for defining instruments and applying effects. On the right, a window titled 'matrix 0.0' displays a grid of characters representing musical events. The grid is a 10x10 matrix where characters like 's', 'd', 'j', 'h', '@', and '.' are placed at specific grid coordinates, representing a polyrhythmic structure.

Fig. 3 A screenshot of ixi lang, with the Matrix grid-based event system

understandable by the audience, and where the use of visual elements was part of the language syntax (Fig. 3).

In many ways, ixi lang continues ideas from the graphical explorations but here using what I call a CUI (Code User Interface). Even though ixi lang is a textual interface, it perhaps more shamelessly applies design patterns that ixiQuarks tried to ignore: there is a temporal score where characters represent sounds and spaces silence. There is an underlying temporal grid, which syncs the musical events to a default tempo clock. There is a clear timeline-based design in ixi lang, but the flexibility of the language and specific features make both polyrhythmic and polymetric explorations easy. In terms of graphic design, we find diverse instances of visual elements in the language, for example where effects are applied with symbols such as “>>” and “<<”, a visual reference to how guitar pedals are connected with jack cables (Fig. 4).

ixi lang has been described in detail elsewhere (Magnusson 2011), but in retrospect one could characterise it as a system of language elements that try to move away from typical programming language commands embracing playfulness and simplicity. For this reason the system has strong limitations and it does not extend very well. However, it is used for teaching children from the age of seven live coding and music, and anyone can learn it in about 30 min. The language has various hidden features that can be discovered by eager users, it is quirky, and it contains an autocoder, where the language writes its own code, often resulting in fine music.

```

ubu -> guitar[1 5 2 3 4 ]

ubu >> distort >> reverb

ubu << distort

shake ubu

ubu + 7

```

Fig. 4 This is a typical *ixi lang* score. First we create an agent called “ubu” and give him a guitar instrument. The guitar is the name of a SuperCollider synth definition, and the user can use any of their definitions. Ubu’s score ([1 5 2 3 4]) are the notes in the selected scale, and the spaces between the notes are silences. This is effectively Rousseau’s system of notation from the 18th century. Ubu is then given two effects (distortion and reverb), but in the next line the performer removes the distortion from the effect chain. Then ubu is shaken, which scrambles the numbers in the score, leaving it the same length, but with the notes in different places. Finally ubu’s score is transposed up by a fifth. One of the system’s innovations is to update the code in the text document when a function has been applied to it (such as “shake” or +7)

3.4 The Threnoscope

More recently I have been developing an instrument I call *Threnoscope* (Magnusson 2014). As the other software, it is a live coding system, split into three views: a notational view, a code view, and a console for system output. The notational view and the code view serve as a dual visual representation of the music: the former is a visual description of the sound, whilst the code is a form of prescriptive notation. Both aid the audience in understand the sonic events but they also serve as interfaces for further control by the performer.

The visual system contains circles that represent the harmonics of a fundamental frequency, often tuned to a 55 or 54 Hz A note. In the former case, the second harmonic is then 110 Hz, the third 165 Hz, fourth 220 Hz, and so on. Notes (or drones) can be created on these harmonic circles or anywhere in between through a structure of tonic (the harmonic, ratio, degree, or frequency). The interface has crossing lines that represent the loudspeakers. Any combination from two to eight speakers can be used. The speakers serve as static timelines where notes travel across the circular space. This creates an unusual looping structure, heavily influencing the music created with the system (Fig. 5).

The Threnoscope was initially planned as a musical piece, to be performed by the author and anyone who wanted to play the piece.⁸ After two decades of interest

⁸However, further development and user experience shows that the system is more of a compositional tool, an instrument, and not a musical piece. Admittedly, the boundaries are not very clear here and the author has had interesting discussions with users who are of different opinions of what might constitute a musical piece.

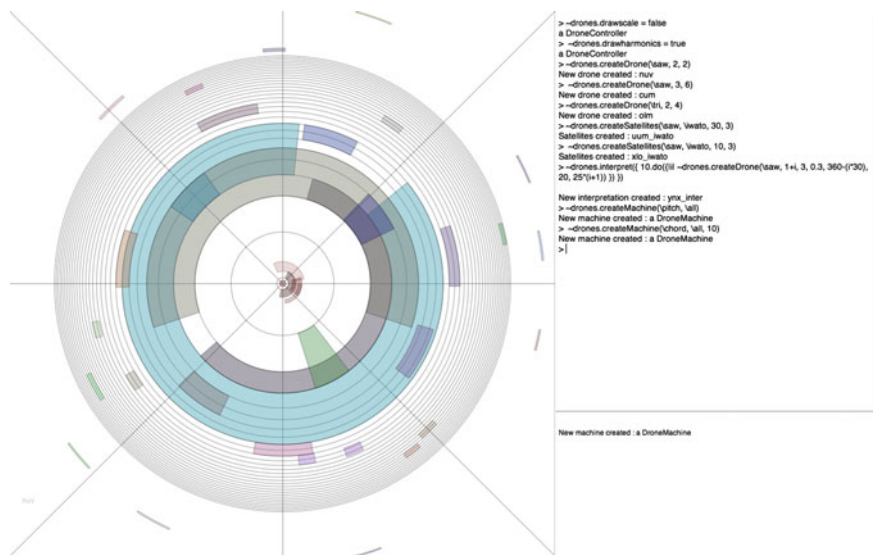


Fig. 5 A screenshot of the Threnoscope microtonal live coding system. The circles represent the harmonics of a fundamental tone. The crosshair lines represent the speakers, here an eight-channel surround system. Notes are the *coloured* wedges that can move around the space, or stay static. In the middle we see a machine that can affect the drones in different ways, supplementing the performer's actions

in microtonal music and alternative tunings and scales, I felt the need to study these areas more formally and the best way to do that is to develop a digital instrument implementing these concepts. My aim was also to move away from notes as events that happen in time, but rather conceive of them as spatial phenomena that move around without a set duration. Although the system can be used to play staccato notes that disappear immediately, the system is designed as an encouragement for long duration musical events, where Morton Feldman might be considered a rest-less character, but La Monte Young and Phill Niblock on a similar wavelength.

The Threnoscope is created for live improvisation, as a live coding instrument. The textual interface is considered the most expressive and free interface for this compositional system. A key problem with graphical user interfaces is that their elements take up screen estate, where the biggest elements call for the attention of the user. Designing a graphical user interface for musical software therefore resembles the writing of a musical score. With text, on the other hand, it is only the imagination and vocabulary of the performer that sets the limitations of what could be possible within the language framework. As an example, at the spur of the moment, some performer might want to create 100 sine waves randomly on the first six harmonics at different degrees in a chromatic scale. This could easily be written as following:

```
100.do({ ~drones.createDrone(\sine, rrand(2, 7), degree: rrand(1, 9))}).
```

Obviously, it would not make sense to create an interface for such musical acrobatics. Here the coding language is a much more appropriate and simple interface than buttons, drop down menus or sliders would ever be.

There is a score format for the Threnoscope, which enables composers to write linear or non-linear pieces. This format is a timed array with code instructions in it. The code score can be visually represented and manipulated during its playback. This can be useful when composing, but often the scores are short scores that are played during an improvisation, almost like a “lick” or an incorporated musical phrase in jazz or other improvised music.

4 Conclusion

The development of musical software has been described above as an experimental activity that goes hand in hand with sound and music research. A redefined boundary between the instrument maker and the musician is forged, a practice which requires a strong knowledge of both music and materiality. Musical composition at this level requires music technological research. We find that, in order to develop the music or instrument that one works towards, one has to understand key concepts of the source domain—such as human gestural patterns, or the resonant properties of physical materials. Just as a composer for acoustic instruments needs to understand the acoustics of the instruments, music theory, harmony, and rhythm, the composer of digital systems will need to comprehend the physics of sound, digital signal processing, software engineering, human-computer interaction, as well as music theory. For composers and software developers the question is how to represent these new features of sonic control. A new notational language is needed, and it is in that context that the work above is presented.

For some musicians, it might feel convenient to buy off-the-shelf products that perform many of the things we might want to do, but at some stage the software will limit compositional ideas and performance options. This inspires musicians to conduct their own compositional work through research and development of their own music technologies, designing affordances and setting constraints relevant to the particular musical work (Magnusson 2010). The best practitioners in this field are the ones who can manage their time on instrument building, software development and other engineering tasks, whilst still keeping a focus on their compositional intention. People who work this way report that time spent on learning, researching and experimenting will result in novel musical output that is unique, personal and of a strong musical and technical identity.

References

- Ackrill, J. L. (1987). *Nicomachean ethics*, book 6, chapter 4. In A. New (Ed.), *Aristotle reader*. Oxford: Clarendon Press.
- Attali, J. (1985). *Noise: The political economy of music*. Minneapolis: University of Minnesota Press.
- Boden, M. A. (1990). *The creative mind: Myths and mechanisms*. London: Weidenfeld and Nicolson.
- Hunt, A. (2003). MidiGrid: Past, present and future. In *Proceedings of new interfaces for musical expression 2003*. Montreal, Canada.
- ixi. (2000). <http://www.ixi-audio.net>. Accessed September 8, 2016.
- Leman, M. (2008). *Embodied music cognition and mediation technology*. Cambridge, Massachusetts: MIT Press.
- Magnusson, T. (2006). Screen-based musical instruments as semiotic machines. In *Proceedings of new interfaces for musical expression*. Paris: IRCAM.
- Magnusson, T. (2007). The ixiQuarks: merging code and gui in one creative space. In *Immersed music: The ICMC 2007 conference proceedings*. Copenhagen: Re:New.
- 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.
- Magnusson, T. (2011). ixi lang: A SuperCollider parasite for live coding. In *International computer music conference*. Huddersfield: University of Huddersfield.
- Magnusson, T. (2014). Improvising with the threnoscope: Integrating code, hardware, GUI, network, and graphic scores. In *Proceedings of new interfaces for musical expression 2014*. London: Goldsmiths College.
- Marx, K. (2007). *Capital: A critique of political economy—The process of capitalist production*. New York: Cosimo.
- Mitcham, C. (1994). *Thinking through technology: The path between engineering and philosophy*. Chicago: University of Chicago Press.
- Pinch, T. J., & Trocco, F. (2002). *Analog days: The invention and impact of the moog synthesizer*. Cambridge, MA: Harvard University Press.
- Roads, C. (1996). *The computer music tutorial*. Cambridge, MA: The MIT Press.
- Schnell, N., & Battier, M. (2002). Introducing composed instruments, technical and musicological implications. In *Proceedings of new interfaces for musical expression 2002*. Limerick: University of Limerick.
- Stiegler, B. (1998). *Technics and time, 1: The fault of epimetheus*. Stanford: Meridian. Stanford University Press.
- Wanderley, M. M. (2000). Gestural control of music. Paris: Ircam. <http://recherche.ircam.fr/equipements/analyse-synthese/wanderle/Gestes/Externe/kassel.pdf>. Accessed February 2016.