Chapter 8
Live Music-Making: A Rich Open Task Requires a Rich Open Interface

Dan Stowell and Alex McLean

Abstract In live human-computer music-making, how can interfaces successfully support the openness, reinterpretation and rich signification often important in live (especially improvised) musical performance? We argue that the use of design metaphors can lead to interfaces which constrain interactions and militate against reinterpretation, while consistent, grammatical interfaces empower the user to create and apply their own metaphors in developing their performance. These metaphors can be transitory and disposable, yet do not represent wasted learning since the underlying grammar is retained. We illustrate this move with reflections from live coding practice, from recent visual and two-dimensional programming language interfaces, and from musical voice mapping research. We consider the integration of the symbolic and the continuous in the human-computer interaction. We also describe how our perspective is reflected in approaches to system evaluation.

8.1 Introduction

In this chapter we discuss themes of our research, strands of which reflect our title's assertion in various ways. Our focus here is on live music-making, in particular improvised or part-improvised performances which incorporate digital technologies.

D. Stowell (✉)
Centre for Digital Music, Queen Mary University of London, London, UK
e-mail: dan.stowell@eecs.qmul.ac.uk

A. McLean (✉)
Interdisciplinary Centre for Scientific Research in Music, University of Leeds, Leeds, UK
e-mail: a.mclean@leeds.ac.uk

Is music-making rich and open? Rich, yes, as evident from the many varieties of emotional and social content that a listener can draw out of music, meaningful from many (although not all) perspectives (Cross and Tolbert 2008). Even unusually constrained music styles such as minimalism often convey rich signification, their sonic simplicity having a social meaning within wider musical culture. Music is particularly rich in its inner relationships, with musical themes passed between musicians both consciously and subconsciously, weaving a complex tapestry of influence. And open, yes: the generative/composable nature of musical units means a practically unbounded range of possible music performances. While musical genres place constraints on music-making, such constraints are often seen as points of departure, with artists celebrated for pushing boundaries and drawing a wide range of cultural and meta-musical references into their work. As a result, new musical genres spring up all the time.

We will start by considering the role of computers in music-making and the interfaces that facilitate this, before focusing on live music-making explored through approaches such as gestural interaction and live coding, and strategies that may combine their advantages. Finally, we will consider the evaluation question in relation to our position, and with respect to evaluation approaches we have developed.

8.2 Rich Interfaces

Against the rich and open background of music as a whole, we examine the use of computers in live music making. Famously, computers do nothing unless we tell them to. We can think of them as lumps of silicon, passively waiting for discontinuities amongst the electric and magnetic signals, which provide the on/off states of digital representation. Computers operate somewhat like a mechanical music box, where pins on a cylinder are read by tuned teeth of a steel comb. In particular, computers are controlled by performative language, where describing something causes it to happen. Beyond this, the magic of computation comes when such sequences of events describe operations upon themselves, in other words perform higher order, abstract operations. We can describe computers then as providing an active system of formal language for humans to explore. Musicians may choose to work directly in this system of discrete language, on the musical level of notes and other discrete events such as percussive strikes. Alternatively, they may use computer language to describe analogue systems such as traditional musical instruments, and expressive movements thereof. As such, computers allow us to engage with music either on the level of digital events or analogue movements, but we contend that the greatest potential for a musically rich experience lies in engaging with both levels, simultaneously.

8.2.1 Interfaces and Metaphors

Many computer music interfaces are based on pre-existing music technology: the mixing desk, the step-sequencer grid, the modular synth patchbay. These are sometimes called *design metaphors* (Carroll et al. 1988), though they may often behave more like similes, especially in interfaces which inherit features from multiple prior technologies. Design metaphors have an advantage of providing a good leg-up for those users who are familiar with the original technologies, but can lead to problems: such users may get unpleasant surprises when the analogy is incomplete, while unfamiliar users may face what seems like inexplicably-motivated design decisions. In particular, these interfaces are full of *skeuomorphic* design elements, originating in physical constraints which no longer apply. The user may be left feeling as though they are dealing with a nonsensical metaphor, which induces unneeded limitations (such as running out of display space), and embeds now-irrelevant design decisions (e.g. based on wiring considerations).

Rigorous attempts to base computer interface design around coherent metaphors have consistently met with failure (Blackwell 2006b). From this it would seem that structuring software design around fixed metaphors does not hold cognitive advantage, beyond helping users adjust to software interfaces in the short term. It seems likely that this is because such metaphors typically reflect neither the "problem space" (the target music domain) nor the breadth of possibilities provided by the computer. The target music domain is in any case hard to specify and may vary from genre to genre or track to track. If we assume that everyone has their own systems of metaphor (Cognitive Semantics; Lakoff and Johnson 1980), then we should instead develop interfaces that let people apply their *own* metaphors. This is an important part of our definition of an open interface.

Runciman and Thimbleby (1986) state a slightly more general version of this requirement: "premature design commitments must be avoided" (p. 440). They give examples from programming language design and from spreadsheet interfaces, arguing in particular that interfaces should allow the user to assign input and output roles to parts of the interface at will, rather than having the roles pre-defined by the designer. (It is interesting to consider how that idea might be incorporated into music programming interfaces.) The cells of a spreadsheet can indeed be flexibly allocated as input or output, and this flexibility and freedom of spatial arrangement is one aspect of spreadsheets which allows users to construct their own metaphors on top of the basic architecture (Hendry and Green 1994). Spreadsheets also provide a "low-viscosity" interface particularly suited to incremental or exploratory working (ibid.). Spreadsheets are a central focus of research in the field of *end-user programming*, essentially being programming environments targetted at those writing code for their own use. Spreadsheets are not designed for professional programmers, but are widely used by professionals in a wide range of other domains (Blackwell 2006a), and so must be taken seriously as programming environments.

Spreadsheet interfaces were originally motivated by a design metaphor for paper worksheets, but they now have standard features (such as absolute and relative cross-referencing) which are not constrained by the metaphor, instead providing a reusable toolset for users to build and edit their own structures. They are however oriented towards static data representations rather than temporal forms such as music. In the following subsections we describe some of our work which strives towards similar openness in a sound/music context, toward musical interfaces which allow the user to define and repurpose their own metaphors.

8.2.2 Mapping Vocal Gestures

A specific musical example comes from one of our (DS's) research into interfaces based on extended vocal techniques – in particular, beatboxing, which is a genre of vocal percussion (Stowell 2010, Section 2.2). Vocal expression is potentially a very rich source of information that could be used by musical interactive systems – pitch, loudness, timbre, linguistic content (if any) – and traditions such as beatboxing demonstrate a wide variety of timbral gestures, rich in signification.

One way to connect such a vocal signal to a computer system is by detecting vocal "events" in real time and classifying them (Stowell and Plumbley 2010, and citations within). However, vocal events can be robustly classified into only a small number of classes (especially in real time, where only the start of the event's sound is known), so the end result is a system which can learn to trigger one of a small number of options – like playing a drum machine with only three or four buttons. Much of the richness of the audio input is discarded. This can create a system which is accessible for immediate use by amateurs, but does not lead to long-term engagement as a tool for musical expression, certainly not for experienced beatboxers.

A more expressive way to make use of the vocal source is to treat the timbral input data as a continuous space, and to try to recreate some of the nuance of performers' continuous timbral gestures; for example by controlling a synthesiser such that it performs analogous gestures (Stowell and Plumbley 2011). Any "classification" is deferred to the perception of the listener, who can understand nuances and significations in a way that is beyond at least current technology. In a user study with beatboxers (discussed further below), this approach was found useful. Further, we informally observed the interface's openness in the fact that the participating beatboxers found new sounds they could get out of the system (e.g. by whistling, trilling) that had not been designed in to it!

The classification approach is not exactly enforcing a metaphor but pre-judging what variation in the input is musically salient, which has a similar channeling, constraining effect on creative options (a premature design decision). This research is one example in which designing for an open musical interaction can allow for richer musical possibilities. Similar lessons might be drawn about physical gesture interfaces, for example – should one discretise the gestures or map them continuously? The continuous-mapping approach has been used by some performers to achieve detailed expressive performance, even when the mappings are simple.[1]

8.2.3 Shape in Notation

When we fix gesture as marks on a two dimensional surface, we call it a notation. In Western culture these marks generally represent discrete events, but may also represent continuous mappings. Indeed, precursors to staff notation include cheironomic neumes, which notate the articulation of melody with curved lines. In both discrete and continuous notations, one dimension is generally dedicated to time, and the other to a quality, classically pitch. The time dimension, or timeline, is also present in notation interfaces in much music software, from the commercial offerings of sequencers to the experimental drawn sound interfaces such as Xenakis's UPIC. In effect this software constrains users to considering just one dimension at a time to describe the state of a musical moment. This allows the evolution of a musical quality in detail, but makes cross-parameter relationships difficult to describe.

The *Acid Sketching* system was developed by one of the present authors (AM) to explore the use of geometric forms and relationships in a music interface beyond standard dimensional mappings. The Acid Sketching interface consists of an ordinary pen and paper, where the latter also acts as a projection surface. When shapes are drawn on the paper with an ink pen, they are identified and analysed using computer vision. These shapes are translated to sound synthesis parameters, and their relative positions translated into a polyphonic sequence, using a minimum spanning tree algorithm. This use of a minimum spanning tree turns a visual arrangement into a linear sequence of events, a kind of dimensionality reduction. The path along the minimum spanning tree is traced, as shown in Fig. 8.1. We claim for a richness from using this graph structure, as it is built from the highly salient perceptual measure of relative distance in 2D space. After all when we view a picture, our eyes do not generally read from left to right, but instead jump around multiple fixation points, influenced by the structure of the scene (Henderson 2003).

Each shape that is drawn on the page represents a particular sound event. The nature of each sound event is given by morphological measurements of its corresponding shape, where each measurement is mapped to sound synthesis parameters. Specifically, *roundness* is calculated as the ratio of a shape's perimeter length to its area, and maps to envelope modulation; *angle* is that of the shape's central axis relative to the scene, and maps to resonance; and finally, the shape's *area* maps to pitch, with larger shapes giving lower pitched sounds.

[1] http://ataut.net/site/Adam-Atau-4-Hands-iPhone.

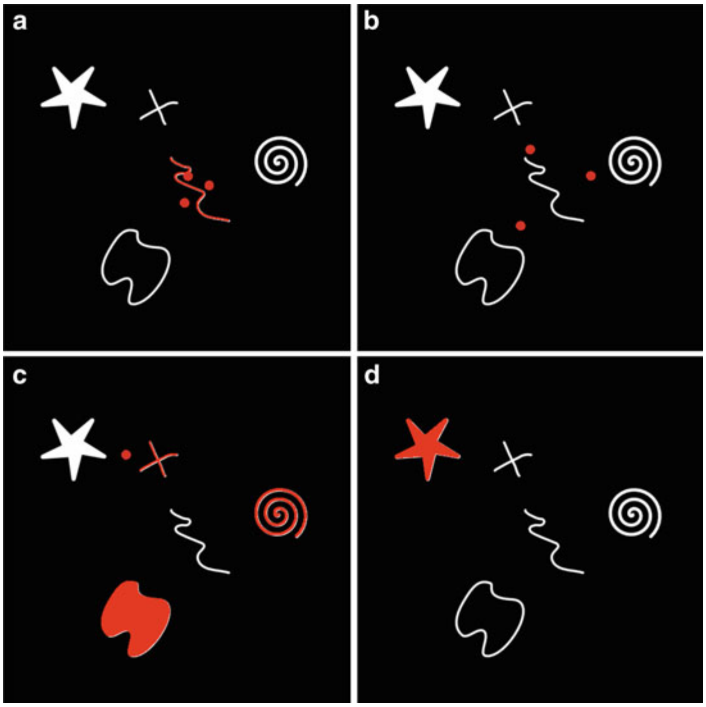

Fig. 8.1 Sequencing of shapes in Acid Sketching. Each shape is drawn to represent a different sound, given by its size, angle and roundness. Shapes are shown with a darker hue when they are 'played'. (**a**) Central shape is 'played', *circles* shown moving towards nearest neighbours. (**b**) *Circles* continue to trace line towards nearest neighbours. (**c**) Nearest neighbour shapes are 'played', with *circle* moving between the cross shape and the final neighbouring shape. (**d**) The final shape is 'played', and the process repeated (Color figure online)

Visual feedback is projected on to the paper using a standard data projector. This feedback takes the form of moving dots, tracing the path from one shape to the next along the edges of the minimum spanning tree, flood-filling each shape as its corresponding sound event is triggered. The geometric relationships employed by the Acid Sketching prototype are not formally tested, but we assert them to be perceptually salient. The correspondence between shape and timbre appear to be straightforwardly learnable, suggesting that this prototype system could be developed further into an engaging interface for live music.

Acid sketching demonstrates a use of analogue symbols which have morphological properties continuously mapped from those of what is represented. Further work by Stead (2011) demonstrates a similar system which allows mappings to be user controlled in a straightforward manner, in effect allowing the user to create their own vision-sound metaphors. A further challenge is to integrate analogue and discrete symbols in a mutually supporting manner. In particular, we look forward to the development of programming languages which are enriched with analogue symbols, in much the same way that language is enriched with prosodic speech.

8.3 Programming Languages

How would one design a computer music interface that can allow for a rich, structured yet open-ended musical expression? One answer is to design for an interaction pattern that makes use of human abilities to represent ideas in a structured but unbounded way, to abstract, to make meta-references – all well-represented in the linguistic faculty. A grammatical interface is a consistent and so learnable interface. Many different levels of musical expression find a representation in formalised language, from music-theoretic abstractions, through pattern manipulations such as modulations and L-systems, down to sample-by-sample structures for audio synthesis and effects. Hence the grammatical interface represented by programming/scripting languages is used in some of the more open sound-and-music systems (SuperCollider, CMusic, ChucK, SAOL).

8.3.1 The Skeuomorph vs. The Abstract Grammar

A grammatical interface is consistent, and so both learnable and powerful, but perhaps it achieves this at some cost. It sits well with our linguistic faculty but may be in tension with some other faculties: an abstract grammar lacks anchoring in familiar relations such as physical spatial arrangement, relationship to external entities, or signification by reference to common understandings. This is the converse of the situation with the skeuomorph, whose cognitive benefit is anchored in some experiential memory which provides signification and is useful for recall – but only if the user shares this prior experience, and if they are unlikely to be overly constrained by habits learned from it.

Hence our proposal that interfaces should let people apply their own metaphors, providing a more transitory use of metaphor than that often baked in to an interface by its designers. The signification and accumulated knowledge assumed by a skeuomorph becomes outdated over time, from year to year or even gig to gig, yet the relationship between performer and audience is rich in signification. One question is then how to support this without requiring users to build their own user interface afresh for each new situation, and without discarding the learning they have already acquired. A grammatical interface goes a long way toward satisfying this, since most of the learning is invested in the underlying composable structure; but there is work still to be done on managing the relationship between the language and the referential aspects of a music performance (both for the performer and the audience).

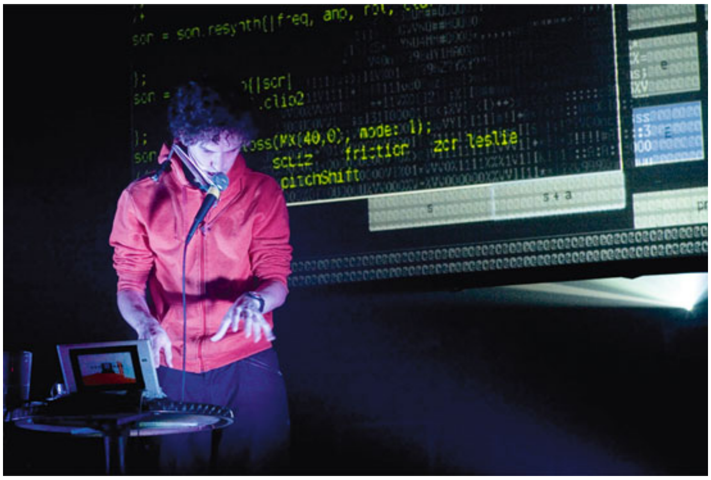

Fig. 8.2 Solo performance by Stowell as MCLD, combining beatboxing with live coded audio processing

8.3.2 Live Coding

Live coding (Collins et al. 2004) is an emerging practice of creative public programming, used to make music and other art forms in a live context. It renders digital performance in some sense more transparent to the audience, allowing them to share in the creative process.

Improvised performance with a tightly-constrained system (such as a simple drum machine) can be expressive; but improvised performance with an open system (such as an audio-oriented programming environment) allows for an interaction that coherently gives access to many of the different levels of music-making in the digital system, from high-level structure to phrasing to sound design and effects.

Most current livecoders are using performance systems that have not been around for long enough to reach the well-publicised figure of 10,000 h of practice for mastery; the virtuoso livecoder might not yet have been encountered. However, many people spend many hours typing and/or programming in their daily lives, and one advantage of a programming interface over a skeuomorphic interface might be that it can recruit skills developed in neighbouring arenas.

A livecoder, like any improvising performer, is under pressure to do something interesting in the moment. Livecoders can use abstraction and scheduling so the notion of "in the moment" may be a little different to that for more traditional instrumental improvisers. It can lead to a lack of immediacy in how the performer's actions relate to the music, which can sometimes deny the more raw physiological expressionism that some people seek in music. Hence it may be useful to combine the symbolic interaction of livecoding with open gesture-based expression; one of us (AM) has been doing this in collaboration with vocalists, guitarists, drummers and banjo players. The other (DS) has taken both roles in solo live performance by combining livecoding with beatboxing (Fig. 8.2). The cognitive load for one performer carrying out both roles is high, but the combination of continuous organic expression with symbolic abstraction helps to provide immediate multimodal access to the multiple levels of musical ideas. Compare this situation with dual-task experiments such as that of Dromey and Bates (2005) which found evidence of cognitive competition between manual and speech tasks performed concurrently. In the beatboxing-and-livecoding scenario, the vocal and typing tasks are not independent since they both work towards a common goal, but are only loosely coupled since they tend to operate on different timescales. We argue (from experience not experiment) that the skill gained from learning such a multimodal interaction allows confluence rather than competition of the modalities.

Connecting livecoding with performers' sounds and gestures is one way to integrate the organic and continuous into the programming interaction. As we will see in the following section, it is possible to cast programming itself in terms such that the symbolic and the continuous are both part of the ontology of the language.

8.3.3 Visual Programming Notation

Programming language source code is generally considered as discrete, one dimensional text constrained by syntactical rules. Myers (1990, p. 2) contrasts *visual* programming languages as "any system that allows the user to specify a program in a two (or more) dimensional fashion." This definition is highly problematic, for a number of reasons. First, several text based languages, such as Haskell and Python have two dimensional syntax, where vertical alignment is significant. Second, amongst those systems known as visual programming languages, it is rare that 2D arrangement has any real syntactical significance, including Patcher languages such as Puredata and Max/MSP (Puckette 1988). Indeed lines and boxes in visual programming languages allow non-visual, high dimensional syntax graphs of hypercubes and up.

The significance of visual notation then is generally as secondary notation (Blackwell and Green 2002), in that it is not syntactical but still of key importance to human readability. We can relate this to Dual Coding theory (Paivio 1990), in treating visuospatial and linguistic representations as not being in opposition, but rather supporting one another, with humans able to attend to both simultaneously, experiencing an integrated whole. Visual layout therefore not only supports readability, but supplements code with meaningful expression that is in general ignored by the software interpreter.

Some computer music interfaces, such as the ReacTable by Jordà et al. (2007), Nodal by Mcilwain et al. (2006) and Al-Jazari by Dave Griffiths (McLean et al. 2010) use visual layout in primary syntax. In the case of the ReacTable, Euclidean proximity is used to connect functions in a dataflow graph, and proximity and relative orientation are used as function parameters. In the case of Nodal and Al-Jazari, city block distance maps to time, in terms of the movements of agents across a grid. Inspired by the ReacTable in particular, one of us (AM) is developing a visual, pure functional programming notation based on Haskell, designed for live coding of musical pattern. This is a work in progress, but feedback from preliminary workshops with non-programmers has already been highly encouraging (McLean and Wiggins 2011). All four of the aforementioned visual programming languages allow, and with the exception of Nodal are designed primarily for live coding. Whereas research from the live coding field has mainly been concerned with time, we can think of this research as extending computer music notation into space.

We assert that integration between linguistic and spatial representations is what makes a musical experience rich. We can relate this to beatboxing, which is experienced both in terms of discrete instrumental events via categoral perception, and continuous expression within spatial experience. This is much like the relationship between the perception of words and prosody, respectively discrete and continuous, but both symbolic and integrated into a whole experience. By considering a programming notation as necessarily having both linguistic and visuospatial significance, we look to find ways of including both forms of representation in the human-computer interaction.

8.4 Rich and Open Evaluation

This attitude towards musical interface design must be reflected in our approach to evaluation. Much development of new musical interfaces happens without an explicit connection to HCI research, and without systematic evaluation. Of course this can be a good thing, but it can often lead to systems being built which have a rhetoric of generality yet are used for only one performer or one situation. With a systematic approach to HCI-type issues one can learn from previous experience and move towards designs that incorporate digital technologies with broader application – e.g. enabling people who are not themselves digital tool designers.

Wanderley and Orio (2002) made a useful contribution to the field by applying experimental HCI techniques to music-related tasks. While useful, their approach was derived from the "second wave" task-oriented approach to HCI, using simplified tasks to evaluate musical interfaces, using analogies to Fitts' Law to support evaluation through simple quantifiable tests. This approach leads to some achievements, but has notable limitations. In particular, the experimental setups are so highly reduced as to be unmusical, leading to concerns about the validity of the test. Further, such approaches do not provide for creative interactions between human and machine.

For live music-making, what is needed is more of a "third wave" approach which finds ways to study human-computer interaction in more musical contexts in which real-time creative interactions can occur. And live music-making can feed back into HCI more generally, developing HCI for expressive and ludic settings and for open interactions.

One of us (DS) developed a structured qualitative evaluation method using discourse analysis (DA) (Stowell et al. 2009). DA originates in linguistics and sociology, and means different things to different people: at its core, it is a detailed analysis of texts (here, transcribed participant interviews) to elucidate the structured worlds represented in those texts. In the context of a user of a new interface, it can be used to explore how they integrate that interface into their conceptual world(s), which gives a detailed impression of affordances relatively uncontaminated by the investigator's perspective.

This approach is useful and could benefit from further exploration, perhaps in different contexts of interface use. The approach bears an interesting comparison against that of Wilkie et al. (2010), who analyse musicians' language using an embodied cognition approach, which differs in that it decomposes text using a set of simple metaphors claimed to be generally used in abstract thought. As in any exploratory domain, the approach which attempts to infer structure "directly" from data and the approach which applies *a priori* structural units each have their advantages.

Such rich and open evaluation approaches sit well with the nature of creative musical situations. Alternative approaches may be worthwhile in some cases, such as controlled experimental comparisons, but often risk compromising the musical situation. As one example from a slightly different domain, Dubnov et al. (2006) conduct a numerical analysis of an audience's self-evaluated response to a composed piece of music over time. The study went to great lengths to numerically explore audience response in an authentic musical context – commissioning a composed piece whose structure can take two configurations, attracting a concert audience willing to be wired up to continuous rating system, etc. Their results were fairly inconclusive, demonstrating mainly that such a scientistic approach is at least possible if logistically difficult. (Simpler approaches in the same mould exist in the computer-games literature, where the audience can often be only one person (Mandryk and Atkins 2007).) In evaluating systems for music-makers we have the added complication that gathering concurrent data is generally not possible: self-reports would distract from the music-making process, while physiological measures (including brain activity sensors) are generally disrupted by the muscle movement impulses (and sweating) that occur in most music-making. Thus we see little prospect in the immediate future for concurrent protocols, hence the use of retrospective protocols in e.g. Stowell et al. (2009).

Music-making HCI evaluation is still very much an unfinished business: there is plenty of scope for development of methodologies and methods. Evaluation of music-making, like that of computer games, fits well with the developments in the HCI field that are called "third paradigm" (non-task-focused, experiential, ludic). But further: music-making is a key area in which the development of rich and open, yet structured, HCI approaches are crucial to the development of the field.

8.5 Rich and Open Questions

In relating the above themes we have provided a particular view on music interaction, extrapolating from existing analog and digital interactions to try to look towards what could be possible. We conclude then by outlining the themes we have touched upon, and proposing directions we might take in our search for new, rich and open music interactions.

Firstly, in a research field in which the study of embodied interaction is beginning to mature (e.g. through gestural and tablet interfaces), we highlight the role of computer languages and grammatical interfaces. This aspect is represented by the small but growing community of live coding researchers, and we argue it allows for a productive, creative repurposing for musical expression. This focus is not mainstream in musical interface research: in presenting their paper for the New Interfaces for Musical Expression conference, Aaron et al. (2011) noted that the word *language* was conspicuously missing from the top 200 keywords for the conference. In sympathy we take the view that programming languages give us an opportunity to explore music that we have only begun to comprehend. We add to the argument well made by Patel (2007), that music and natural language share close family resemblances, by considering computer language as a third category with its own unique properties. The research theme, ongoing for many years in many guises, is in how to address the properties of computer language to music, towards expressive, higher-order music notations.

An issue we have not discussed is that of learning to program, and whether the strategies we propose could enable (or hinder) the use of programming-type musical systems more widely. We have argued that a grammatical interface is flexible and learnable, and also that such a music interface can productively combine symbolic and continuous aspects; but we have also noted the tension with the anchored accessibility offered by more skeuomorphic designs.

However our argument for rich and open interfaces does not rest on computer languages alone, but in greater integration between abstract language, and embodied gesture. The expressive power of natural language lies in its close integration with prosodic gesture when it is communicated, and accordingly the full potential for computer language can only be unlocked when it is fully integrated with visual and gestural interfaces.

References

Aaron, S., Blackwell, A. F., Hoadley, R., & Regan, T. (2011). A principled approach to developing new languages for live coding. *Proceedings of New Interfaces for Musical Expression, 2011*, 381–386.

Blackwell, A. F. (2006a). Psychological issues in end-user programming. In H. Lieberman, F. Paternò, & V. Wulf (Eds.), *End user development* (Human-computer interaction series 9th ed., pp. 9–30). Dordrecht: Springer. doi:10.1007/1-4020-5386-X_2. chap 2.

Blackwell, A. F. (2006b). The reification of metaphor as a design tool. *ACM Transactions on Computer-Human Interaction, 13*(4), 490–530. doi:10.1145/1188816.1188820.

Blackwell, A., & Green, T. (2002). Notational systems – the cognitive dimensions of notations framework. In J. M. Carroll (Ed.), *HCI models, theories, and frameworks: Toward a multidisciplinary science* (pp. 103–134). San Francisco: Morgan Kaufmann.

Carroll, J. M., Mack, R. L., & Kellogg, W. A. (1988). Interface metaphors and user interface design. In M. Helander (Ed.), *Handbook of human-computer interaction* (pp. 67–85). New York: Elsevier.

Collins, N., McLean, A., Rohrhuber, J., & Ward, A. (2004). Live coding in laptop performance. *Organised Sound, 8*(03), 321–330. doi:10.1017/S135577180300030X.

Cross, I., & Tolbert, E. (2008). Music and meaning. In *The Oxford handbook of music psychology*. Oxford: Oxford University Press.

Dromey, C., & Bates, E. (2005). Speech interactions with linguistic, cognitive, and visuomotor tasks. *Journal of Speech, Language, and Hearing Research, 48*(2), 295–305.

Dubnov, S., McAdams, S., & Reynolds, R. (2006). Structural and affective aspects of music from statistical audio signal analysis: Special topic section on computational analysis of style. *Journal of the American Society for Information Science and Technology, 57*(11), 1526–1536. doi:10.1002/asi.v57:11.

Henderson, J. (2003). Human gaze control during real-world scene perception. *Trends in Cognitive Sciences, 7*(11), 498–504. doi:10.1016/j.tics.2003.09.006.

Hendry, D. G., & Green, T. R. G. (1994). Creating, comprehending and explaining spreadsheets: A cognitive interpretation of what discretionary users think of the spreadsheet model. *International Journal of Human Computer Studies, 40*(6), 1033–1066. doi:10.1006/ijhc.1994.1047.

Jordà, S., Geiger, G., Alonso, M., & Kaltenbrunner, M. (2007). The reacTable: Exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of tangible and embedded interaction* (pp. 139–146). doi:10.1145/1226969.1226998.

Lakoff, G., & Johnson, M. (1980). *Metaphors we live by* (2nd ed.). Chicago: University of Chicago Press.

Mandryk, R. L., & Atkins, M. S. (2007). A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. *International Journal of Human Computer Studies, 65*(4), 329–347. doi:10.1016/j.ijhcs.2006.11.011.

Mcilwain, P., Mccormack, J., Lane, A., & Dorin, A. (2006). Composing with nodal networks. In T. Opie & A. Brown (Eds.), *Proceedings of the Australasian Computer Music Conference 2006* (pp. 101–107).

McLean, A., & Wiggins, G. (2011). Texture: Visual notation for the live coding of pattern. In *Proceedings of the International Computer Music Conference 2011* (pp. 612–628). Huddersfield.

McLean, A., Griffiths, D., Collins, N., & Wiggins, G. (2010, July 5–7). Visualisation of live code. In *Proceedings of Electronic Visualisation and the Arts London 2010* (pp. 26–30). London.

Myers, B. (1990). Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing, 1*(1), 97–123. doi:10.1016/S1045-926X(05)80036-9.

Paivio, A. (1990). *Mental representations: A dual coding approach (Oxford psychology series)*. USA: Oxford University Press, New York.

Patel, A. D. (2007). *Music, language, and the brain* (1st ed.). USA: Oxford University Press, New York.

Puckette, M. (1988). The patcher. In C. Lischka & J. Fritsch (Eds.), *Proceedings of the 1988 International Computer Music Conference*, Cologne, September 20–25, 1988 (pp. 420–429).

Runciman, C., & Thimbleby, H. (1986). Equal opportunity interactive systems. *International Journal of Man–machine Studies, 25*(4), 439–451. doi:10.1016/S0020-7373(86)80070-0.

Stead, A. G. (2011). User configurable machine vision for mobiles. In *Proceedings of psychology of programming interest group*. York: University of York. http://www.ppig.org/papers/23/15%20Stead.pdf.

Stowell, D. (2010). *Making music through real-time voice timbre analysis: machine learning and timbral control*. PhD thesis, School of Electronic Engineering and Computer Science, Queen Mary University of London. http://www.mcld.co.uk/thesis/.

Stowell, D., & Plumbley, M. D. (2010). Delayed decision-making in real-time beatbox percussion classification. *Journal of New Music Research, 39*(3), 203–213. doi:10.1080/09298215.2010.512979.

Stowell, D., & Plumbley, M. D. (2011). Learning timbre analogies from unlabelled data by multivariate tree regression. *Journal of New Music Research*. doi:10.1080/09298215.2011.596938.

Stowell, D., Robertson, A., Bryan-Kinns, N., & Plumbley, M. D. (2009). Evaluation of live human-computer music-making: Quantitative and qualitative approaches. *International Journal of Human Computer Studies, 67*(11), 960–975. doi:10.1016/j.ijhcs.2009.05.007.

Wanderley, M. M., & Orio, N. (2002). Evaluation of input devices for musical expression: Borrowing tools from HCI. *Computer Music Journal, 26*(3), 62–76. doi:10.1162/014892602320582981.

Wilkie, K., Holland, S., & Mulholland, P. (2010). What can the language of musicians tell us about music interaction design? *Computer Music Journal, 34*(4), 34–48.