http://dx.doi.org/10.14236/ewic/EVA2016.8

Live Coding, Live Notation, Live Performance

Richard Hoadley
Digital Performance Laboratory, Anglia
Ruskin University
Cambridge, UK
research@rhoadley.net

This paper/demonstration explores relationships between code, notation including representation, visualisation and performance. Performative aspects of live coding activities are increasingly being investigated as the live coding movement continues to grow and develop. Although live instrumental performance is sometimes included as an accompaniment to live coding, it is often not a fully integrated part of the performance, relying on improvisation and/or basic indicative forms of notation with varying levels of sophistication and universality. Technologies are developing which enable the use of fully explicit music notations as well as more graphic ones, allowing more fully integrated systems of code in and as performance which can also include notations of arbitrary complexity. This itself allows the full skills of instrumental musicians to be utilised and synchronised in the process.

This presentation/demonstration presents work and performances already undertaken with these technologies, including technologies for body sensing and data acquisition in the translation of the movements of dancers and musicians into synchronously performable notation, integrated by live and prepared coding. The author together with clarinetist Ian Mitchell present a short live performance utilising these techniques, discuss methods for the dissemination and interpretation of live generated notations and investigate how they take advantage of instrumental musicians' training-related neuroplasticity skills.

Physical computing, algorithms, live symbolic notation, cross-domain expression.

1. INTRODUCTION

Live coding of live notations seeks to join together two disparate traditions, one new, one somewhat older. As is likely when attempting to merge such different techniques or technologies, inconsistencies, mismappings and other difficulties are revealed. This paper will review relevant aspects of music technology and notation, live coding, data projection and the computer interface before considering how joining these techniques and technologies might work as well as suggesting what the fruits of such a union might be.
In his 2001 article *Algorithms as Scores* Thor Magnusson (2011) relates live coding to traditional forms of composing and performing:

> The live coder is primarily a composer, writing a score for the computer to perform. It is therefore appropriate that the computer science term for the system in which the live coder evaluates code is normally 'interpreter.' (Magnusson 2011, p. 22)

Is it really accurate to compare the notation of code to music notation, or text for that matter (see Section 3.1)? And what about the comparison of interpretation of code by a computer with a performer's interpretation of notation?

2. MUSIC AND TECHNOLOGY

This research assumes that music is by nature technological. Musical performance with acoustic instruments involves a human controlling a separate object – a piece of technology (even vocal cords!) – in order to create music. It is hardly surprising, then, that electronic and digital technology have had such profound impacts on music and its creation. Equally important is the fact that there are still few, if any, independent electronic musical instruments (for instance the theremin and the Ondes Martenot), apart from those based on existing acoustic instruments such as the electric guitar and the ubiquitous and yet undefinable *synthesiser*.

Music itself can therefore be seen as being defined by the technology that is used to create it: the use of a piano places creates certain bounds around the music that it can create, although experimentation has ensured that these boundaries are constantly being pressurised (for instance by Cage's preparations).

3. MUSIC AND NOTATION

Worldwide, there are many types of music – formal and informal. Many of these use limited notation or no notation at all. Those that do use notation frequently use non-specific types such as text (for example folk songs and hymns, whose vernacular medium is text) or numbers (such as gamelan notation (see Figure 1).

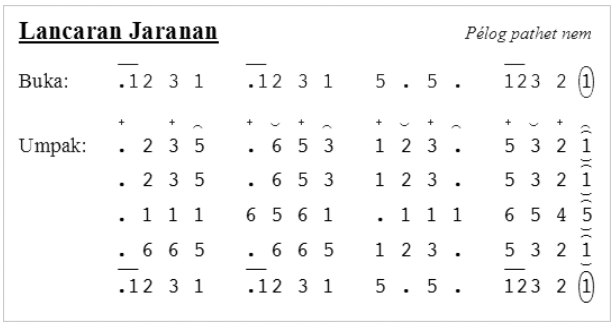

Figure 1: *An example of gamelan notation*

An example of a highly contrasting notational paradigm is Guqin (Sun *et al*. 2010; Leman and Maes 2014), (see Figure 2), explained below:

> The "D" part denotes the finger used to press the string, and the "7.6" part denotes the pressing position. The "Z" part denotes the sound decoration method, the "T" part denotes the plucking method, and the "7" part denotes the string being played. The overall meaning of this symbol is that the player uses left thumb to press on the 7.6 position of the 7th string, uses the right forefinger to pluck the same string outward, and kneads the string while pressing. Notice that other symbols may contain [a different number of] parts. The right side shows a piece of tablature that consists of many different symbols. (Sun *et al*., 2010)

Because of its convoluted history and the complexity of the resulting music itself, the use of common practice notation (CPN) is far from a precise science, as is illustrated by the following description by a computer scientist:

> Traditional notation is biased towards music that is humanly performable. This ... is an obstacle when trying to notate music intended for computer performance, where the notation is often found to be deficient, inconsistent, and redundant. (Hudak *et al*. 1993)

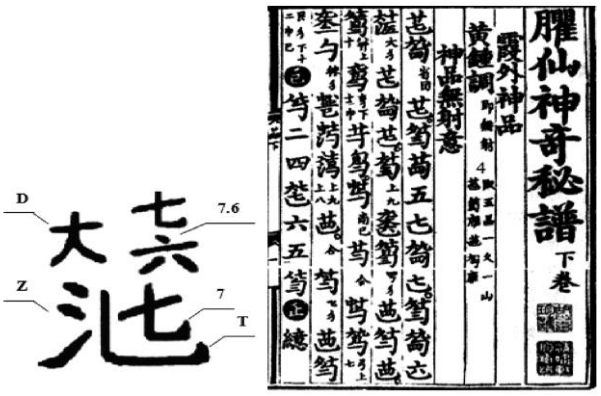

Figure 2: *Guqin tablature notation*

Don Byrd (2016) provides research showing the complex, rich and anomalous features of CPN in his collections of extreme and impossible notations, a part of which includes 'examples of published music ... intended to counter the view that [CPN], while it may have many complex details, is in principle easily mechanizable' – such as Figure 3. In this example the problem lies in the 'floating' bass clefs at the bottom of the left hand staff. These are musically nonsensical but are likely to be understood by a competent pianist. Byrd's collection demonstrates the very real anomalies that can arise from the musicians' occasional practice of developing notations which extend graphic, semantic or combination forms (often intentionally, such as Cardew's monumental *Treatise*). In this case, programming a computer to interpret or understand such notation would present particular problems.

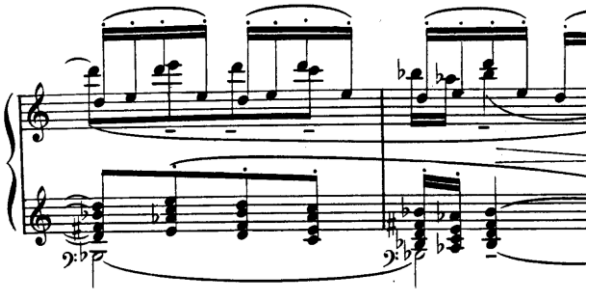

Figure 3: *'Impossible' notation in Debussy's 'Violes'*

In spite of, or maybe because of this, notation remains vital to much music making, including those supporting traditionally non-notated forms (pop and rock backing tracks, etc.). The primary value of notation in this situation is the fact that it is readable and therefore interpretable by humans. Performers from many cultures are able to read, perform and improvise based on notation, the form of which they know intimately – a phenomenon which will be discussed further in Section 8.

In spite of the current interest in specifically musical notations, this research aims to include as many notation types as possible, including graphics, images, text, and dance notations.

3.1 Related work in live notations

Over the last few years there has been a gradually increasing interest in many forms related to live, dynamic, automatic and animated notation, depending on the nuance of the work. Some (Smith 2015) are more based on animated work whereas others (Baird 2005) are more interactive. Some utilise more abstract and graphic work (Rebelo 2015) whereas others attempt to use common practice notation (Kim-Boyle 2010).

There have been many excellent studies of music notations, for instance Cole and Gould (Cole 1974; Gould 2011), and in his recent book *Extended Notation* Christian Dimpker (2013) classifies a number of standard and extended notations. He suggests notations should be exact, simple, not contradictory of traditional systems and as far as is possible, compatible with those systems.

He also identifies *qualitative*, *approximate* and *graphic* which he believes are not suitable for extended and consistent practice, leaving, he says, *action*, *symbolic*, *diagrammatic* and *schematic* notations. In particular, notations are considered in terms of their *consistency* and *reproducability*. They are also described as either *prescriptive*, meaning that they describe the actions the performer should take rather than the sounds they are to produce (such as CPN or tablature) or *descriptive notation* which is used to describe the sounds a performer or ensemble produced in a specific performance. These are reflections of Blackwell and Green's Cognitive dimensions of notations applied to music notation:

> CPN with pencil and paper has high viscosity (editing requires an eraser), a few abstractions (such as key signature, ornaments, and tablature), a few hidden dependencies (changing a key signature changes the notes), some premature commitment (a new instrument cannot be added in its logical position unless a blank stave was left), and is quite diffuse (no way to write 'play broken chords in a simple sequence until the singer starts'). (Blackwell *et al*. 2000)

Cognitive dimensions provide an objective method of comparing 'pencil and paper' methods of writing notation with computer-based ones (and others), from GUI-based visual editors to more code-based software. It is an expansion in the use of the latter by a particular group of users that has developed into the phenomenon called live coding.

4. LIVE CODING

Code, like mathematics, has often been described as a form of notation (Aaron *et al*. 2011; Magnusson 2011) and (Blackwell and Green 2003, pp. 103–134). Exciting though Magnusson's comparison between live coding and the processes of composing, interpreting and performing might be, it is possible that his idea misses a fundamental issue: that the music played by humans on physical instruments is very much defined by the qualities of those instruments and the relationships between performer and instrument. While it may be possible to replace interpretation and performance with a computer's 'interpretation' of code, this will have an equally fundamental effect on the nature of that music (Collins *et al.* 2003; Magnusson 2014). Moreover, acoustic musical instruments are exceptionally limited in their physical characteristics and it may well be these limitations themselves which enable musicians to develop such intimate, specialised relationships.

Instead, as the dominant element in performance, interpretation seems to have been replaced at least in part by the new and distinctly code-focused idea of 'changing the rules as part of performance':

> Live coding is making changes to algorithms as they run, with the possibility for both live feedback and a live audience. (See http://iclc.livecodenetwork. org/2016/)

During a talk given in 2013 Alex McLean presents a pragmatic view of 'changing rules while they are followed'. In response to the observation that 'n a live situation you have a certain range of settings and you change things within the settings rather than actually changing the rules', McLean suggests that:

> Improvisation is not quite as free as it might appear because there are rules. I generally make techno and that has quite fixed rules about what you can and can't do ... In terms of the algorithm I have lots of code that I use and don't change during a performance but I suppose I could if I wanted to I might have an idea for something and try it out. It's Turing complete so in theory anything can happen but in practice it's probably quite a limited exploration. (McLean 2013, 35:16)

Others emphasise different aspects of live coding, such as providing alternatives to conventional means of expression (Wilson *et al.* 2014); improvisation in computer music (McCallum 2011, 2 minutes); the hazards of performance (McLean *et al.* 2004), and accessibility (McLean *et al.* 2004)

Some more negative aspects have been identified as well, for instance 'the corporeal difference between direct control of a physical instrument and

indirect control via program code,' (Collins and Brown 2009, p. 2) and the 'computer head' problem during performance where the performers are, heads down inside their laptops, significantly less animated than the audience (McLean 2013).

As has been pointed out, live coding is in fact still very much a 'fuzzy concept', with 'no specific tools, practices, or musical aesthetics at play' (Magnusson 2014). As such it is an ideal sandbox to experiment in, perhaps enabling composers to feel as if they are avoiding the formality of western compositional practices. Ultimately, live coding is another piece of computer technology and the interface is just another interface, but perhaps the greatest freedom is that with learning and practice, it allows a significant degree of freedom from the strictures of major developers of audio (or any other type of) software. Whether the software uses code, or boxes, or cords (or chords), it's the ability to understand and create the (probably heterogeneous) environment in which you wish to work that is important.

5. PROJECTION AND DISSEMINATION

If the first item of the Toplap draft manifesto is 'Give us access to the performer's mind, to the whole human instrument', the second seeks to provide a practical method of achieving this: 'Obscurantism is dangerous. Show us your screens.' http: //toplap.org/wiki/ManifestoDraft. The Toplap website also announces that "All code manipulation is projected for your pleasure." (McLean et al. 2004), also, presumably to discourage any nefarious use of functions or software.

My notations are commonly displayed, but these are, of course, the *result* of other code, so should the *original* code be 'exposed', or the notation, or both? Does it matter if the functions used are opaque? In fact, in order to perform the generated notations the musicians must have access to them whether the audience wants to see them or not. In my piece *How to Play the Piano in 88 Notes* (see video at https://vimeo.com/131433884), the best solution has always been for the pianist to play directly from the main audience projection. Although some performers have mentioned a slight feeling of loss of intimacy with this arrangement, it seems to be more than compensated for by the increased sense of common experience felt between audience and performer.

The issue of projection and/or dissemination of live notations includes a number of other issues and features. It is not so practical, for instance, if using live dance notations for them to be solely projected onto a fixed screen as dancers need to be able to move freely. Wearable technologies such as Google Glasses might provide a way forward.

6. QWERTYS AND MUSICAL INSTRUMENTS

The third item of Toplap draft manifesto reads: 'Programs are instruments that can change themselves' (McLean *et al*. 2004, page: ManifestoDraft). This reveals a possible gap in the understanding of the nature of computers as musical instruments as opposed to the nature of acoustic instruments. While it is surprisingly difficult to define what a musical instrument actually is, one thing that does seem to be the case is that for better or worse an acoustic musical instrument's identity is very much bound up with how consistently it matches understood standards, and its relative permanence thereafter. The history of the pianoforte is long and complex and involves many developments and evolutionary changes. This has resulted in a few standardised versions of the piano being generally accepted throughout the world, with a consequent effect on the music written for and performed on the instrument (Ripin *et al*. 2016).

For obvious reasons the QWERTY keyboard plays an very important role in live coding, primarily because it is difficult to construct complex, or even simple code, without the precision that comes with those symbols:

> It's the first performance art form that's come out of computer music in a way that's truly computer oriented, so rather than trying to turn the computer into a more traditional instrument by plugging a keyboard in ... you're just instructing the machine. – Matthew Yee-King, (McCallum 2011, 3 minutes, https://vimeo.com/20241649).

However, QWERTY keyboards are not musical instruments according to most definitions. Like computers, they are general purpose devices. As has been pointed out by Nick Collins, (a.k.a. live coder Sick Lincoln): 'typing is hardly the most visually exciting interfacing method' (Collins *et al*. 2003, p. 322).

7. STYLE

In the same way that the nature of an acoustic instrument directly affects what music can be written for it, all of the above factors play a part in determining the style of music that can be generated through a live coded interface.

The issue of the 'blank slate' has been discussed in relation to live coding and in particular its feasibility when considered at different levels of interpretation. In general, however, if the method – that is, starting to code from a blank screen with no

pre-prepared material – is to be used at all then by definition there will have to be allowances made during early parts of the music while musical ideas and textures are developed. And of course, doing this live with code via a QWERTY keyboard (whether one has the ability to touch type or not) is not the quickest nor the most expressive method physically (compare the effort and speed involved in generating a complex chord directly at a piano, for instance).

To accommodate this, many live coding systems emphasise looping – particularly rhythm and arpeggiation based loops (for instance Tidal, Sonic Pi and Impromptu (McLean and Wiggins 2010; Aaron and Blackwell 2013; Sorensen and Gardner 2010)). Magnusson's *Threnoscope* emphasises drones: 'Any music can be played with the system, but it is deliberately designed to encourage a certain way of thinking, where tunings, harmonics, harmonic beating, scales, and slow movements are emphasized' (see https://vimeo.com/63335988).

8. LIVE CODING FOR LIVE NOTATION AND PERFORMANCE

The above discussions suggest that the role of the human musical interpreter is of fundamental importance to the music created, and that in notated music this comes about through the intimate links between interpreting written notation and the resulting motor actions developed by musicians over a lifetime of practice and experience. Live coding (as with some other forms of electroacoustic music) omits this process, replacing the human performer with a computer.

A model proposed by Andrew Sorenson (Sorensen and Gardner 2010) does provide an interesting example of interactive live coding, where a live coder and a violinist improvise together – each taking material from the other in their unwinding narratives.
Below are presented some concrete examples of notations generated by these techniques. They demonstrate the levels and type of live coding used and discuss the levels of flexibility. These examples are from an early version of the research and use adapted versions of algorithms originally used in *Calder's Violin* (Hoadley 2012). The notation is generated in the software INScore by code evaluated in the SuperCollider audio programming environment McCartney (2002). INScore receives the notation in the form of Open Sound Control messages constructed in SuperCollider's native language via a utility class currently still in development by the author. SuperCollider is also used to generate and manipulate audio. In spite of the occasional use of MIDI values these are instruments neither rendered nor controlled by MIDI.

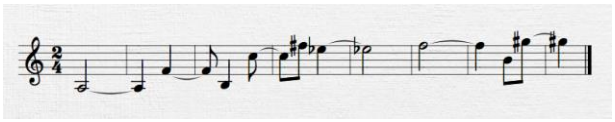

Figure 4: *An algorithmically generated melody*

Figure 4 shows one version of the melody notation generated by the following code:

```
doMelBit.value(0.15)
```

In SuperCollider, evaluation occurs by placing the cursor on the code's line and pressing the *enter* key. The value, here 0.15, is in this case the only argument to the function and represents the overall likelihood of rests occurring. In this case, the value is low and, as can be seen in Figure 4, there are indeed no rests in the melody. It would certainly be feasible to alter this value live and reevaluate the line, particularly as the algorithm generates music that would take a significant time to play.

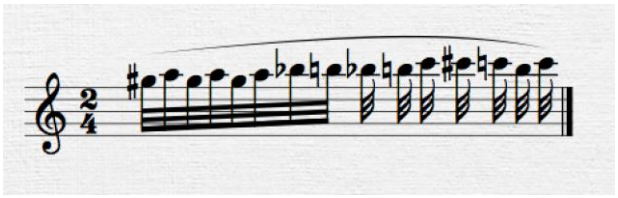

Figure 5: *An algorithmically generated gesture*

Figure 5 is generated by the following code (some parameters have been omitted for clarity):

```
multiPattDigiLines.value(1,
loopWait:0.001,
pitchInc: [1], loNote: rrand(50, 65),
hiNote:90,
loVel:10, hiVel:60);
```

This function will generate one 'swirl' with notes each one semitone apart. The lowest note will be somewhere between 50 and 65 (MIDI values), the highest 90 and velocity will run (similarly smoothly) betwen 10 and 60.

The code that generated Figure 6 is slightly more complex. Here's the top-level:

```
makeChordStream-cs.value(i, inf,
[(40+(i*12))], [0, 2, 4, 1, 1, 3, 2,
4].scramble, chordStreamsDurArray,
[10, 20, 30, 40], 0.75, 0.0)
```

This also generates a series of rhythmically repeated chords. The array [0, 2, 4, 1, 1, 3, 2, 4] is used to generate these notes. The notes from the

phrase shown in the figure are taken from these chords.

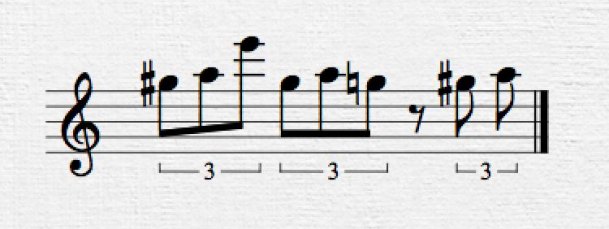

Figure 6: *An algorithmically generated rhythmic figure*

These are examples taken from an early version of the demonstration that accompanies this paper. The notation will be generated from these and similar code snippets. It can be seen that all three code examples can be used as is, the values changed and the altered snippets reevaluated. It is more likely, though, that I will prepare in advance as follows:

```
doMelBit.value(0.15)
doMelBit.value(0.35)
doMelBit.value(0.55)
```

Then, in a performance situation, it is easier to concentrate on exactly when it is more appropriate to evaluate each line. Of course, these values are only parameters because I have decided to make these part of the function. If I decided that I wanted to be able to control the pitch content of the generated melodies, I could add to the function arguments to define the highest and lowest notes to be used in each phrase:

```
doMelBit.value([60, 70], 0.15)
doMelBit.value([65, 75], 0.35)
doMelBit.value([70, 80], 0.55)
```

in which case the melody would gradually become more disjointed while increasing in tessitura – a very particular musical gesture. This can be seen as simply 'changing settings', but at this level of algorithmic composition these are the meaningful parameters. These methods very much reflect the practice of performing a new piece on an acoustic instrument – you practice the 'algorithms' necessary to perform the piece – the interpretation then involves choosing the 'settings' made available via those 'functions'. While a metaphor, (and any metaphor has its limits), I think it holds true in these circumstances.
The following, then, is a mock up of a series of pre-arranged mid and high level commands that might be used in a particular live coding, live notation event:

```
doMelBit.value(0.15)
doMelBit.value(0.35)
makeChordStream-cs.value(i,
inf, [(40+(i*12))], [0, 2,
4, 1, 1, 3, 2, 4].scramble,
chordStreamsDurArray, [10, 20,
30, 40], 0.75, 0.0)
doMelBit.value(0.55)
makeChordStream-cs.value(i,
inf, [(50+(i*12))], [1, 3,
5, 2, 2, 4, 3, 5].scramble,
chordStreamsDurArray, [15, 24,
33, 42], 0.75, 0.0)
multiPattDigiLines.value(1,
loopWait:0.001, pitchInc: [1],
loNote: rrand(50, 65), hiNote:90,
loVel:10, hiVel:60);
```

Then it is possible to:

- evaluate these lines sequentially (or in any order) without any editing – requiring a minimum number of key/mouse actions
- edit the arguments in each line and then evaluate them separately
- edit the functions on which these statements are based, evaluate those functions and then evaluate the code again – requiring the maximum number of key/mouse actions

Performing any of these actions is unlike the performance of any type of acoustic musical instrument but through experience the option first proves to be most musical, allowing for more 'expressive' performance because one is required to have the minimum contact with the keyboard. How feasible or necessary it is to type code expressively is one of the most 'fuzzy' aspects of live coding.

Developing the idea further, if the instrumentalist is presented with Figure 5, it might be considered a legitimate response to answer with a more custom piece of code such as:

```
multiPattDigiLines.value(1,
loopWait:0.001, pitchInc: [1],
loNote: rrand(40, 50), hiNote:60,
loVel:10, hiVel:60);
```

which will generate a much line of notation much lower in tessitura (Figure 7). A technique like this, using somewhat limited live coding, can be used to engage in genuinely improvisatory music making between live coder and instrumental musician.

***Figure 7:** An algorithmically generated gesture lower in tessitura*

A comment sometimes made by reviewers about these techniques is that they relegate the instrumentalist to a sight-reader with a *lower* than might be expected involvement in interpretation. What this seems to miss is that there is no compulsion on the instrumentalist to play the generated notation at all, let alone precisely. By definition, in most of these pieces there is no such thing as a wrong note (although there could be). Most performers balance practicality with musicality and the need to make a performance; they understand that if the notation is available to the public, they need to consider how the public balances what they see with what they hear.

9. CONCLUSIONS AND FUTURE WORK

One of the most exciting things about the software INScore is that it implements such a wide variety of formats controllable by OSC all of which can make a significant contribution to performance notations. In addition to Guido and MusicXML, these include various raster and vector graphics formats, text and html.

Interactivity is built into both INScore and SuperCollider and other interesting experiments involve simple hardware control of algorithmic generation by the performer. As an example, the performer might have access to two footswitches. Depressing one might trigger one of the algorithms described above. Multiple pressings might generate more material – both audio and notation based. If too much material is being generated, the second pedal can be used to cancel or reverse the process.

Similarly, use of data acquisition devices such as the Microsoft Kinect can be used for both control and/or interpretation (if I can use that term!) by code – work already partially undertaken (Hoadley 2014, 2015).

Overall, live coding provides an interesting and powerful toolbox to the technologist-composer. Although on first glance the techniques of live coding may appear somewhat incompatible with music notation, in fact they suit each other very well and will, no doubt provide ample opportunities for further expressive investigation for some time to come.

10. REFERENCES

Aaron, S. and A. F. Blackwell (2013) From sonic pi to overtone- creative musical experiences with domain-specific and functional languages. In ACM (Ed.), *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling and design*, Volume 2013, pp. 35–36.

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

Baird, K. C. (2005) Real-time generation of music notation via audience interaction using python and gnu lilypond. In *Proceedings of New Interfaces for Musical Expression*, Vancouver, BC, Canada, pp. 240–241.

Blackwell, A. F. and T. R. Green (2003) *Notational Systems – the Cognitive Dimensions of Notations framework*. Morgan Kaufmann.

Blackwell, A. F., T. R. Green, and D. J. Nunn (2000) Cognitive dimensions and musical notation systems. In ICMA (Ed.), *Proceedings of International Computer Music Conference*, Berlin, Germany. ICMA.

Byrd, D. (2016) Extremes of Conventional Music Notation. Indiana University Bloomington, USA. http://homes.soic.indiana.edu/donbyrd/CMNExtremes.htm

Cole, H. (1974) *Sounds and Signs*. London: Oxford University Press.

Collins, N. and A. R. Brown (2009) Generative music editorial. *Contemporary Music Review 28*(1), 1–4, February.

Collins, N., A. McLean, J. Rohrhuber, and A. Ward (2003) Live coding in laptop performance. *Organised Sound 8*(3), 321–330.

Dimpker, C. (2013) *Extended Notation*. Zurich: Lit Verlag.

Gould, E. (2011) *Behind bars: the definitive guide to music notation / Elaine Gould*. London: Faber Music Ltd. (Includes bibliographical references and index.)

Hoadley, R. (2012) Calder's violin: Real-time notation and performance through musically expressive algorithms. In ICMA (Ed.), *Proceedings of International Computer Music Conference*, pp. 188–193. ICMA.

Hoadley, R. (2014) Dynamic music notation in quantum canticorum. In R. K. *et al*. (Ed.), *Proceedings of the 50th Artificial Intelligence and Simulation of Behaviour Conference*, Goldsmiths, University of London, UK.

Hoadley, R. (2015) Semaphore: cross-domain expressive mapping with live notation. In M. Battier (Ed.), *Proceedings of the International Conference on Technologies for Music Notation and Representation*, Paris, France, pp. 48–57. TENOR: IReMus.

Hudak, P., T. Makucevich, S. Gadde, and B. Whong (1993) Haskore music notation – an algebra of music. *Journal of Functional Programming 1*(1), 1–18.

Kim-Boyle, D. (2010) Real-time score generation for extensible open forms. *Contemporary Music Review 29*(1), 3–15.

Leman, M. and P.-J. Maes (2014) The role of embodiment in the perception of music. *Empirical Musicology Review 9*(3-4), 236–246.

Magnusson, T. (2011) Algorithms as scores: Coding live music. *Leonardo Music Journal 21*, 19–23.

Magnusson, T. (2014) Herding cats: Observing live coding in the wild. *Computer Music Journal 38*(1), 8–16, Spring.

McCallum, L. (2011) Show us your screens. Video.

McCartney, J. (2002) Rethinking the computer music language: Supercollider. *Computer Music Journal 26*(4).

McLean, A. (2013) Changing rules while they are followed: Live coding the embodied loop. https://vimeo.com/69687342

McLean, A., D. della Casa, S. Knotts, S. Aaron, A. Brown, and J. A. Romero (2004) Toplap website. Electronic.

McLean, A. and G. Wiggins (2010) Tidal – pattern language for the live coding of music. In *Proceedings of the 7th Sound and Music Computing Conference*.

Rebelo, P. (2015) Composing with graphics: Revealing the compositional process through performance. In *Proceedings of the New Technologies for Music Notation and Representation Conference*, Paris, France. TENOR, May.

Ripin, E. M., S. Pollens, P. R. Belt, M. Meisel, A. Huber, M. Cole, G. Hecher, B. K. de Pascual, C. A. Hoover, C. Ehrlich, E. M. Good, R. Winter, and J. B. Robinson (2016) *Grove Music Online*, Volume Grove Music Online, Chapter Pianoforte. Oxford University Press.

Smith, R. R. (2015) An atomic approach to animated music notation. In M. Battier, J. Bresson, P. Couprie, C. Davy-Rigaux, D. Fober, Y. Geslin, H. Genevois, F. Picard, and A. Tacaille (Eds.), *Proceedings of the International Conference on Technologies for Music Notation and Representation*, pp. 39–47. TENOR.

Sorensen, A. and H. Gardner (2010) Programming with time: cyber-physical programming with impromptu. *ACM Sigplan Notices 45*(10), 822–834.

Sun, Q., D. Zhang, Y. Fan, K. Zhang, and B. Ma (2010) Ancient Chinese zither (guqin) music recovery with support vector machine. *ACM Journal on Computing and Cultural Heritage 3*(2).

Wilson, S., N. Lorway, R. Coull, K. Vasilakos, and T. Moyers (2014) Free as in beer: Some explorations into structured improvisation using networked live-coding systems. *Computer Music Journal 38*(1), 54–64, Spring.