Algorithms as scores: coding live music

Article (Published Version)

Algorithms as Scores: Coding Live Music

Thor Magnusson

ABSTRACT

The author discusses live coding as a new path in the evolution of the musical score. Live-coding practice accentuates the score, and whilst it is the perfect vehicle for the performance of algorithmic music it also transforms the compositional process itself into a live event. As a continuation of 20th-century artistic developments of the musical score, live-coding systems often embrace graphical elements and language syntaxes foreign to standard programming languages. The author presents live coding as a highly technologized artistic practice, shedding light on how non-linearity, play and generativity will become prominent in future creative media productions.

Traditionally the score is thought of as a message from a composer to an instrumentalist who interprets the given information. Although the emphasis varies, the musical score can be conceived as both a *description* of music in the form of written marks and a *prescription* of gestures for an instrumentalist. It has served as what we now call a "file format," where the paper (or parchment) stores the music for later realization or consumption. However, the score is more than encoded music. It is also a compositional tool through which composers are able to externalize their thoughts onto a medium that visually represents the sonic data. The score in its various forms is a mnemonic device that enables more complex compositional thinking patterns than those we find in purely oral traditions.

A careful investigation into its history will illustrate that the score is not a simple object whose nature can be easily defined. It has had multiple functions in various traditions at different time periods. This article considers live coding as a new evolutionary branch of the musical score. It will investigate the background of diverse scoring practices as applied in live coding, where the score is written in the form of an algorithm, either graphically or textually, yet always encoded in the functionality of a programming language. I present my live-coding language ixi lang as a case study.

A Note on the Score

> *Unless sounds are remembered by man, they perish, for they cannot be written down.*
>
> —Isidore of Seville, 7th century

Although musical marks have been written for millennia, the invention of the musical score as we know it today is normally attributed to Guido d'Arezzo (b. 991). A magnificent scholar of music, known for the creation of the solfège, he also invented systems for algorithmic composition, for example, where pitch values would be systematically derived from syllables in the text. Furthermore, d'Arezzo is known for the Guidonian hand [1] (Fig. 1), a system of prescriptive instructions for conducting music, where each part of the hand's digits represents a musical note for the performers. In the work of this medieval scholar we find the roots of many important ideas engaged here.

In the centuries after d'Arezzo, Western culture has exhibited a strong desire to capture music: to represent it in the silence of the written marks and invoke it again through the interpretation of signs. It is a tradition interlocked in a strongly formalistic attitude toward encoding and decoding musical data, resulting in the establishment of an industry of composing and performing music and, importantly, the technology and infrastructure that supports recording, distributing and selling music aimed for machine playback. Since its origins in the 11th century, the traditional score has evolved into a highly sophisticated language for encoding musical expression with various idiosyncrasies or personal styles. Currently, the boundaries of what can be defined as a musical score are fuzzy, as the diverse musical compositions, performances and digital systems require different representations of the musical data. Nevertheless, the great success of the traditional score as a musical technology has established it as the fundament on which our musical education is built.

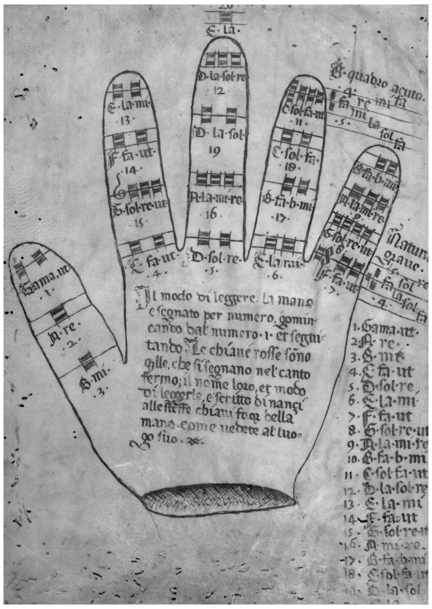


Fig. 1. Guidonian hand with somization syllables. An early technique for music instruction, as well as a mnemotechnic device for musical thinking. (Photo © Jean Gray Hargrove Music Library, University of California, Berkeley)

Thor Magnusson (educator), School of Arts and Media, University of Brighton, Brighton, BN2 0JY, U.K. E-mail: <t.magnusson@brighton.ac.uk>.

See <mitpressjournals.org/toc/lmj/-/21> for supplemental files (such as audio and video) related to this issue of LMJ and accompanying CD. See also <www.ixi-audio.net> for materials related to this article.

Player Pianos and Punch Cards

The musical parameters that have been important in the traditional score are primarily the rough denotations of pitch, velocity and duration. These are features that could easily be encoded onto rolls or cylinders for musical automata, and there are records of early such machines dating back to the 9th century. In the 19th century, punch cards were a common engineering solution, for example, in textile looms, and they were used to automate the popular player pianos of the early 20th century [2]. As instantiated through these technologies, musical notation became notably static and linear. An ideal had been fulfilled: perfect performances to be written for perfect instruments.

Not surprisingly, computers were initially given the same notation system as player pianos, namely punch cards, and the piano roll has become the key visual metaphor in musical software designed for composition and arrangement. It is interesting to observe in this context that this capacity of machines—such as pianolas or early computers—to perfectly render musical scores coincided with a development in which composers increasingly began experimenting with the scores themselves, incorporating elements that emphasized performer engagement and interpretation. This is analogous to developments in painting when, with the advent of the photograph, the machine liberated painters from realism, resulting in impressionism and various other isms such as Cubism and Surrealism [3].

If algorithmic music can be traced to the same 11th-century origins as the modern score in d'Arezzo, we may question its apparent absence through the ages. On closer scrutiny it is clear that there has never been any lack of algorithmic music: It merely presents an interesting problem of encoding. As an example, every group of people improvising or playing together succumbs to implicit rule sets that could easily be reduced to explicit algorithmic steps. However, since algorithms can be generative, resulting in diverse outcomes, it has proved difficult to find the ideal format to write them as musical scores. Although various mechanical automata, such as Winkel's Componium of 1821, were capable of algorithmic music [4], and people such as Ada Lovelace in 1842 speculated about the computational creation of music [5], it is the advent of the digital computer that establishes the ideal medium for writing music in the form of algorithms.

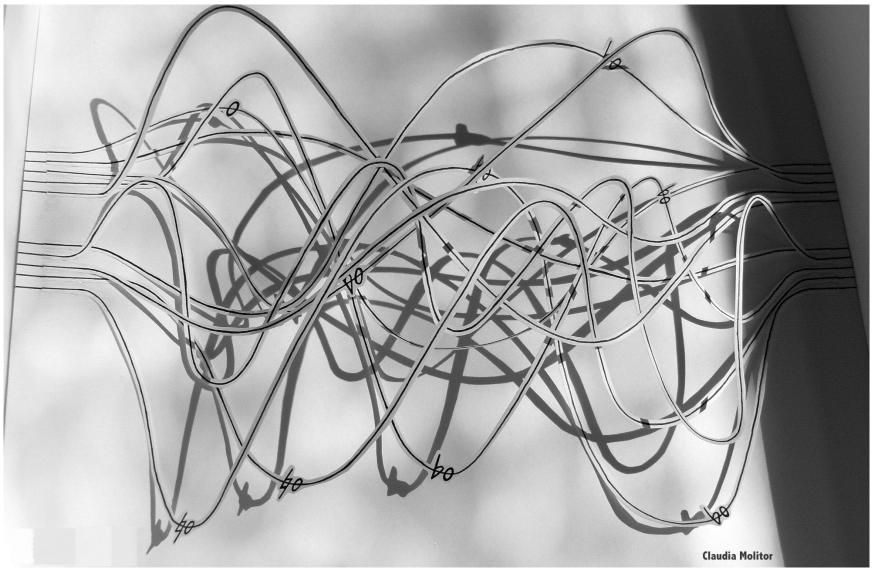
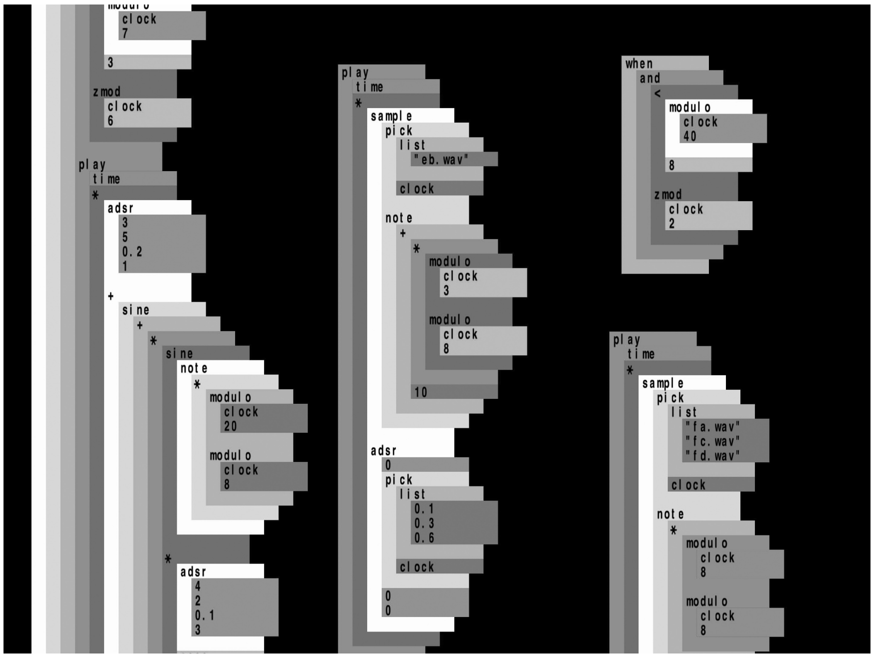


Fig. 2. A snapshot of Claudia Molitor's *3D Score Series.* (© Claudia Molitor)



```
modulo
  clock
  7
3
zmod
  clock
  6
play
  time
  *
    adsr
      3
      5
      0.2
      1
    +
      sine
        +
          *
            sine
              note
                *
                  modulo
                    clock
                    20
                  modulo
                    clock
                    8
            *
              adsr
                4
                2
                0.1
                3
```

```
play
  time
  *
    sample
      pick
        list
          "eb.wav"
        clock
      note
        +
          *
            modulo
              clock
              3
            modulo
              clock
              8
          10
    adsr
      0
      pick
        list
          0.1
          0.3
          0.6
        clock
      0
      0
```

```
when
  and
    <
      modulo
        clock
        40
      8
    zmod
      clock
      2
```

```
play
  time
  *
    sample
      pick
        list
          "fa.wav"
          "fc.wav"
          "fd.wav"
        clock
      note
        *
          modulo
            clock
            8
          modulo
            clock
            8
```

Fig. 3. David Griffiths, *Scheme Bricks*, 2008. Here the functional programming language Scheme is represented graphically as bricks that can be plugged into each other, thus building complex functional graphs out of simple ingredients. (© Dave Griffiths)

Currently the main obstacles for such musical productions are conceptual, not technical, but new media technologies are transforming this situation very rapidly, with composers and software developers increasingly grasping the potential for generative music.

Algorithms as Graphic Scores

> *What is an algorithm if not the conceptual embodiment of instrumental rationality within real machines?*
>
> —Andrew Goffey

The mid-20th century saw many intermedia experiments involving visual media and sound. Artists such as Kandinsky and Klee experimented with how a synchronic medium such as painting could be represented as a diachronic process (such as music). These experiments continued with the Fluxus movement of the 1960s introducing a novel approach, namely the celebration of the algorithm as an art form. Good examples are LaMonte Young with works such as "Draw a straight line and follow it" (*Composition 1960 No. 10*) or Yoko Ono with her instruction pieces, e.g. "Drill a hole in the sky. Cut out a paper the same size as the hole. Burn the paper. The sky should be pure blue" [6].

Graphic scores can represent a special form of algorithm. They are instructions with which the composer has found a reason to go beyond traditional notation. Many composers exploring the open work, such as Christian Wolff, Karlheinz Stockhausen, John Cage, Cornelius Cardew and Iannis Xenakis, have resolved to use the graphic score to extend the musical language [7]. This is often done with the aim of enabling non-linearity, increasing performer interpretation and presenting elements of surprise [8]. The graphic score opens new dimensions in the ontology of music by rejecting linearity in musical notation, thus paving the way for encoded generative or algorithmic music. The emphasis is put on improvisation or the performers' role in the realization of the piece, often using aleatoric or rule-based techniques such as the casting of dice or formalizing rule sets that the performers have to follow. Since the divide between composition and performance becomes vague, it naturally follows that composers using these techniques participate regularly in the performance of their work. The above examples manifest how contemporary music has established a strong basis for live-coding practice.

These 20th-century experiments with the score transformed it into an object of art in itself. In musical performance, the function of graphic scores is not limited to that of instructions for performers. Often the audience is given a chance to view and engage with the score as well. Claudia Molitor's *3D Score Series* (Fig. 2) is a good example of this approach. The score as an object of art was a path of investigation pursued by John Cage in editing the book *Notations*, where, interestingly, Xenakis's Fortran code for *Stochastic Music* is presented as a graphic score. Indeed, graphic scores are often not written for human performers but rather as instructions for the computer. The UPIC software system designed by Xenakis is a good example of an early such system [9]. Furthermore, graphic scores can be descriptive, i.e. used as visualizations of musical works, enabling listeners to engage with the piece through another modality. Rainer Wehinger's 1970 visual rendition of Ligeti's *Artikulation* is a fine specimen of such visualization [10].

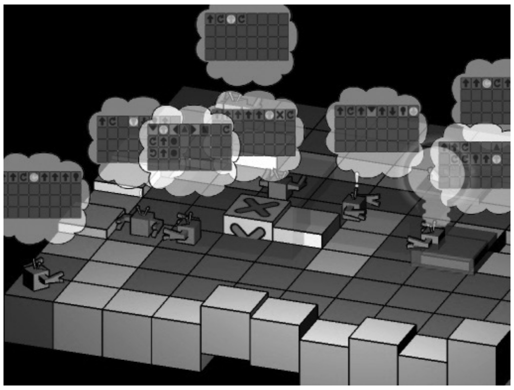
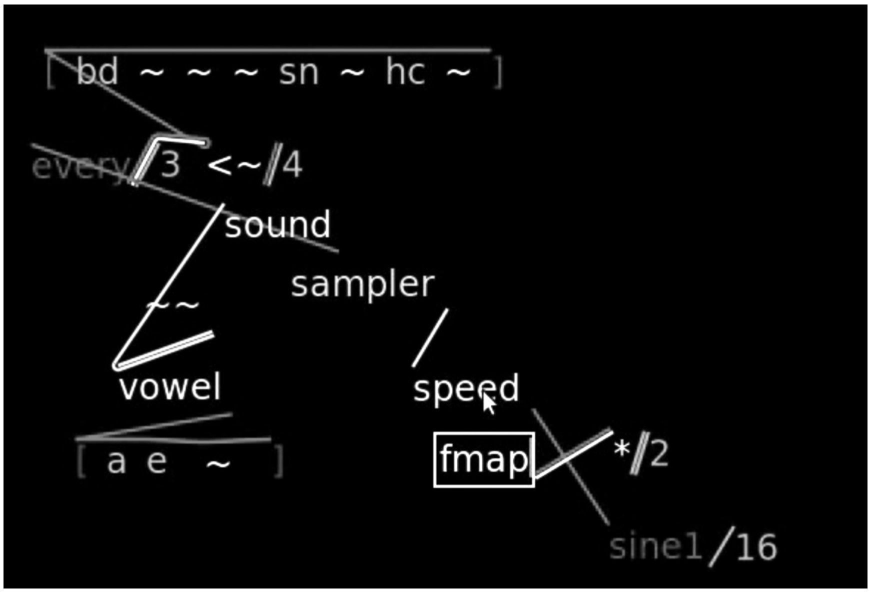


Fig. 4. David Griffiths, *Al-Jazari*, 2007. A live-coding environment that engages with the world of gaming. The robots are programmable, and the interface used to program them is the typical gamepad used in video games. (© Dave Griffiths)

Fig. 5. A screenshot of Alex McLean's Text programming language (2011), where spatial locations of the elements have syntactic relevance, as opposed to many graphical languages such as Pure Data. (© Alex McLean)



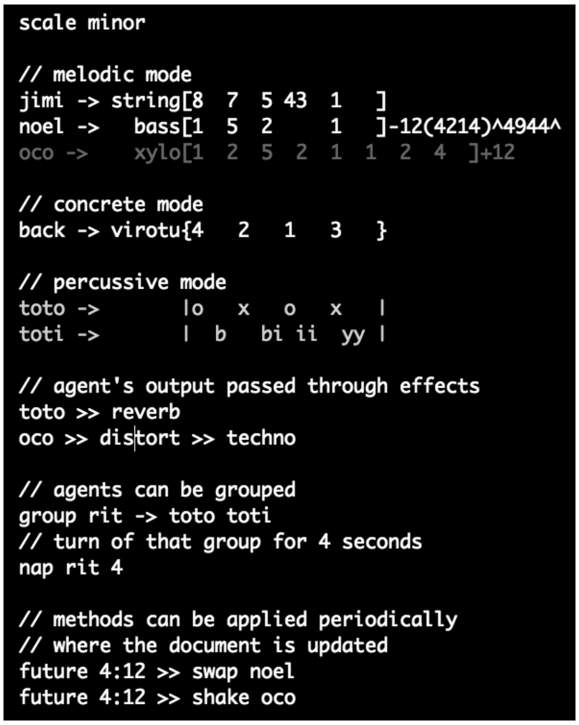
```
[ bd ~ ~ ~ sn ~ hc ~ ]

every 3 <~ 4
            sound
   ~~            sampler

vowel                speed

[ a e ~ ]            fmap   */2

                          sine1/16
```

Composing Algorithms

Live coding is the offspring of the two strong traditions described above: the formalization and encoding of music, often for machine realization, on the one hand, and the open work resisting traditional forms of encoding on the other. Live coding is a form of musical performance that involves the real-time composition of music by means of writing code. This is done in front of an audience, which follows the proceedings on a projected screen. Typically performers start with a clean sheet, a tabula rasa, and build their compositions from scratch. The compositions evolve through the writing of new code, changing code, pausing code or copying a large block in order to transform it into something entirely different. The code is in constant change, often modifying itself. For this reason McLean et al. [11] talk about "codeomorphology," since the code and the music evolve together in an interweaved process observed by the audience.

In addition to tracing the compositional process, the code is also itself a representation of what occurs in the sonic domain. Without it, diverse musical patterns could appear to be one, or a single process could be perceived as many. By inspecting the code one can learn about the instruments, the voices and the form of the musical piece. This representational aspect of code enables performers to externalize their thoughts, thereby freeing some cognitive load and enabling other explorations. A shift has taken place in that the composer is not thinking with a pencil and staff-lined paper but writing instructions in a text document by manipulating graphical forms. Such externalization is an important feature of composition [12], and the different coding environments enable the composer to think differently about music and musical problems.

Even if computer science is the foundation of all live-coding environments,



```
scale minor

// melodic mode
jimi -> string[8  7  5 43  1   ]
noel ->   bass[1  5  2     1   ]-12(4214)^4944^
oco ->    xylo[1  2  5  2  1  1  2  4  ]+12

// concrete mode
back -> virotu{4   2   1   3   }

// percussive mode
toto ->       |o   x   o   x   |
toti ->       |  b   bi ii  yy |

// agent's output passed through effects
toto >> reverb
oco >> distort >> techno

// agents can be grouped
group rit -> toto toti
// turn of that group for 4 seconds
nap rit 4

// methods can be applied periodically
// where the document is updated
future 4:12 >> swap noel
future 4:12 >> shake oco
```

Fig. 6. ixi lang exists at the border of the graphical and the textual, as spaces are important functions in the scores and iconicity is applied in the way methods look. It also introduces the use of a code document that is updated when the code modifies itself (2010). (© Thor Magnusson)

these environments are typically innovative systems that can be defined as art themselves. The goal is to explore musical representation, interaction and thinking through systems that have other criteria than those of traditional programming languages. The works of Dave Griffiths, such as *Scheme Bricks* (Fig. 3) and *Al-Jazari* (Fig. 4), are good examples of graphical representation of algorithmic music scores. As another example, the veteran live coder Alex McLean has recently performed with his new Text live-coding language (Fig. 5), in which, even if it is text-based language, spatial relations define the meaning of the code. It is yet to be seen if this language can be useful for more production-based tasks, but it works well as an artistic tool.

IXI LANG

ixi lang is a live-coding system designed with the criteria that it be fast (maximum 5-second wait before some sound is heard), be understandable by the audience and allow the user to be relaxed whilst performing. The engagement should be primarily at the level of musical composition rather than computer science. The language is high-level and simple, focusing on the arrangement and manipulation of musical events in time. Three main metaphors are utilized: agent, instrument and score. The agent is given an instrument and a score through a simple instruction such as this:

```
jimi -> string[8 7 5 43 1 ]
```

where the agent is called *jimi*, playing a string instrument with a score containing indexes to the notes in a chosen scale. Spaces between the notes represent silence, spatial organization therefore becoming a primary syntax of the language. When other agents are written below *jimi* on the document, this spatiality becomes helpful in arranging events in time. We also see (Fig. 6) that there are three different modes available for scoring: melodic, percussive and concrète. The melodic mode takes numbers from a chosen scale, the percussive works with samples that are mapped to the letters of the alphabet and the concrète mode is intended for the looping of longer sound files. Each of the modes can take suffixes in the form of note duration, note accent, silence between loops, speed and transposition. These can be seen in the case of agent *noel*, where ^ represents accents, (is note lengths, + is transposition and ! is silence before the melody is played again.

ixi lang includes actions (methods) that can be applied to the agents: The performer can instruct *jimi* to reverse, shake up, shift or swap his or her score. When an action is performed on an agent from another location in the document, the agent's code is updated. Furthermore, these actions can be automated and scheduled repeatedly in the future, resulting in a code document in constant change. The coding environment thus serves as a score with a double function (for the computer to interpret and for the audience to follow the musical events), updated in real time, not only by the performer but also recursively and algorithmically by the very score itself. Judging from user feedback and personal experience, the language serves as a constrained musical tool that affords certain compositional practices but prevents others. It is within the limits of the language that the musical exploration takes place, and users report that these limitations encourage creativity [13]. As such it is not clear whether the language is a musical work in itself or a musical tool. This is an example of how many of the old distinctions, such as work and tool or composer and performer, blur in current media practices.

REAL-TIME COMPOSITION AND PERFORMANCE

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." The novelty of live coding is not simply that composition has become a real-time activity [14] but also that the compositional tool is brought forth to the degree that it is seen as a musical composition in and of itself. One could argue that this entails taking live performance of computer music to its logical conclusion, or in the words of a *Wired* journalist, "what music made with computers could—and should—be" [15].

Current content production in new media, in particular for mobile media devices, indicates that the mp3 is being superseded by the app. Here musical scores, compositional systems or instruments for performance are often inherent with so much music that they should be considered musical pieces themselves. In this context, live coding can play an important role in the investigation of algorithmic composition and real-time music performance for these devices. This is especially important as music technologies become increasingly complex and dominant, strongly influencing our musical creativity. As an art form that plays with the core of the tools used for artistic creation, live-coding systems provide a level of self-reflexivity that can be inspiring and educative for designers and composers of other new media art.

CONCLUSION

Live coding has become a widely practiced and prominent art form, as manifested by this year's International Computer Music Conference call for papers and musical performances, both including a live-coding submission category [16]. Thus, the conference organizers acknowledge the status and function live coding has achieved as one solution to the problem of live performance of computer music and audience engagement. Live coding emphasizes the algorithmic nature of music and explores the different conceptual and technical possibilities of representing such work. Although proven successful in finding novel ways of scoring and performing algorithmic music, live coding is still highly experimental, and a vast space of investigation and exploration will take place before it matures through the development of its tools and techniques.

In this paper I have explored how live coding continues the 20th-century tradition of experimentation with the musical score. Live coding presents the ideal set of tools and performance context for algorithms to be written in the form of instructions for machines or people [17]. It shows how these instructions need not necessarily be limited to the textual but can also include the visual and the gestural. Hence, the musical score is in yet another phase of transformation as it proliferates and morphs into diverse functions in accordance with the practices afforded by new mobile, locative, interconnected and computational media. From static scores and deterministic storage formats, we move to scores that are written or executed in real time, resulting in a new type of live music: generative music.

References and Notes

[1] K. Berger, "The Guidonian Hand," in M. Carruthers and J.M. Ziolkowski, *The Medieval Craft of Memory: An Anthology of Texts and Pictures* (Philadelphia, PA: Univ. of Pennsylvania Press, 2004) p. 70.

[2] C.B. Fowler, "The Museum of Music: A History of Mechanical Instruments," *Music Educators Journal* **54**, No. 2, 45–49 (October 1967).

[3] N. Mirzoeff, *An Introduction to Visual Culture* (London: Routledge, 1999).

[4] Fowler [2].

[5] C. Roads, *The Computer Music Tutorial* (Cambridge, MA: MIT Press, 1996) p. 822.

[6] Y. Ono, *Grapefruit: A Book of Instructions and Drawings* (London: Simon & Schuster, 2000, originally published in Japan in 1964).

[7] Examples of such open work include Cage's *Fontana Mix* (1958), Stockhausen's *Plus-Minus* (1963), Pouissieur's *Scambi* (1957) and Cardew's *Treatise* (1963–1967).

[8] J. Cage, *Notations* (New York: Something Else Press, 1969).

[9] I. Xenakis, *Formalized Music: Thought and Mathematics in Composition* (Hillsdale, NY: Pendragon Press, 1992) p. 329.

[10] See <www.youtube.com/watch?v=71hNl_skTZQ>.

[11] A. McLean, D. Griffiths, N. Collins and G. Wiggins, "Visualisation of Live Code," *EVA London Conference Proceedings* (London, 2010), <www.bcs.org/server.php?show=nav.14043>.

[12] And indeed in any cognitive activity—see A. Noë, *Out of Our Heads: Why You Are Not Your Brain, and Other Lessons from the Biology of Consciousness* (New York: Farrar, Straus and Giroux, 2010); A. Clark, *Supersizing the Mind: Embodiment, Action, and Cognitive Extension* (Oxford: Oxford Univ. Press, 2008).

[13] T. Magnusson, "ixi lang: a Constraint System for Live Coding," *Proceedings of ICMC 2011* (Huddersfield, U.K., 2011).

[14] Research and experimental practice is already strong in the field of real-time composition; see, for example, *Contemporary Music Review* **29**, No. 1, and a panel at ISEA 2011 in Istanbul, where the practice of real-time notation was discussed by practitioners. <isea2011.sabanciuniv.edu/panel/chasing-ghosts>.

[15] T. Armitage, "Making Music with Live Computer Code," *Wired* (September 2009), <www.wired.co.uk/news/archive/2009-09/25/making-music-with-live-computer-code-.aspx>, accessed December 2010.

[16] See ICMC 2011 web site: <www.icmc2011.org.uk/>.

[17] N. Collins, "Live Coding of Consequence," *Leonardo* **44**, No. 3, 207–211 (2011).

Manuscript received 1 January 2011.

Thor Magnusson is a senior lecturer at the School of Arts and Media, University of Brighton, U.K. His research is primarily concerned with computer music interfaces and performance and he is the co-founder of the ixi-audio collective. Further information to be found at: <artsresearch.brighton.ac.uk/research/academic/magnusson>; <www.ixi-audio.net/thor/>.

CALL FOR PAPERS

Environment 2.0

Guest Editor: Drew Hemment

A special call for papers seeking cross-disciplinary thinking on environmental sustainability and participatory observation on the environment, climate and biodiversity.

Leonardo is soliciting texts that document the works of artists, researchers and scholars involved in the exploration of environmental sustainability and participatory observation. Themes and issues may include:

- Participatory mass observation of the environment, climate and biodiversity
- Citizen science and issues of participation
- Everyday creativity and social innovation in environmental sustainability
- Ubiquitous, pervasive, locative and mobile communication technology.

In urban environments in particular we are separated from the consequences of our actions as surely as the tarmac of the road cuts us off from the earth beneath. This physical boundary encourages a phenomenological separation. Innovative approaches to participatory observation and mapping can overcome this separation, when combined with the way the internet and digital media have enabled individuals to produce and share information globally and instantly. Cracks in the pavement can be widened by creative intervention or social innovation, including artworks, social entrepreneurship, scientific intervention or innovations that harness everyday creativity.

Linked activities exploring the Environment 2.0 theme have been led by FutureEverything (Futuresonic) and Lancaster University (U.K.).

Authors are encouraged to submit manuscripts or proposals to <leonardomanuscripts@gmail.com>. Leonardo submission guidelines can be found at Leonardo On-Line: <www.leonardo.info>.