GENERAL ARTICLE

Live Coding of Consequence

Nick Collins

ABSTRACT

A live coding movement has arisen from everyday use of interpreted programming environments, where the results of new code can be immediately established. Running algorithms can be modified as they progress. In the context of arts computing, live coding has become an intriguing movement in the field of real-time performance. It directly confronts the role of computer programmers in new media work by placing their actions, and the consequences of their actions, centrally within a work's setting. This article covers historical precedents, theoretical perspectives and recent practice. Although the contemporary exploration of live coding is associated with the rise of laptop music and visuals, there are many further links to uncover throughout rule-based art. A central issue is the role of a human being within computable structures; it is possible to find examples of live coding that do not require the use of a (digital) computer at all.

0 Introduction

The fast pace of computerized life has provoked equally dynamic artistic responses. In one current of contemporary performance practice, usually termed "live coding," the actors seem to take their relationship with computers to a natural conclusion. In intimate portraits of human and machine, they accept the challenge of programming on the spot, typically for an audience in a concert setting. Although there are gentler slopes away from the hot lights, where live coding is just interactive code prototyping on interpreted systems or a networked chatter, many liberating stresses and radical joys have come out of the explicit live scene. Usually, the programming is carried out within some sort of arts programming environment, such that the program output affects audio and visuals and the operator projects his/her screen during performance to make the process (in principle) transparent. From being intently hunched within the creation and manipulation of computer code to acting and dancing through algorithmic rules away from computer screens, the human operator's role is critical. This article will explore many rich veins of work within this scene, acknowledge many historical precedents, touch upon the inter-relation of human and machine rules and, possibly, not proceed in quite the manner expected of it.

(1)2,3,4 Definitions/Declarations

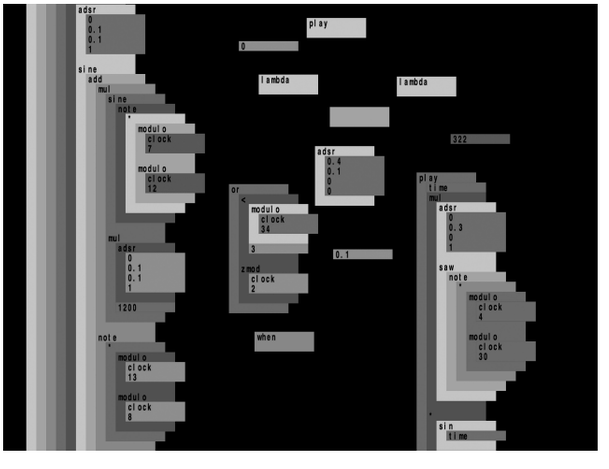
A definition of live coding might preempt the creative exploration of its potential and sets itself up to be re-written in midflow by the more zealous live coders. However, to help the reader less familiar with existing work, let me quickly declare two existing definitions, three perhaps helpful analogies and four real examples. Computers remain a primary touching point, but let us also keep in sight examples that center on human rule-making independent of digital computability.

1. "Live computer music and visual performance can now involve interactive control of algorithmic processes. In normal practice, the interface for such activity is determined before the concert. In a new discipline of live coding or on-the-fly programming the control structures of the algorithms themselves are malleable at run-time" [1].
2. "Digital content (music and/or visuals predominantly) is created through computer programming as a performance" [2].

The two definitions above, reproduced from previous academic papers, concentrate on the audio and visual modalities, primarily focusing on live concert settings for digital multimedia [3]. Let us consider more general extensions as we analogize live coding, exploiting such notions as:

1. following a recipe, but deciding halfway through cooking to substitute one ingredient for another
2. reading a Choose Your Own Adventure book and deciding to change the rules or indeed to rewrite the book because the restrictions and paths did not please you
3. dramatizing a computer algorithm: for example, let people place themselves into order of height, using an agreed-upon sorting algorithm [4]; a live-coding twist would be to perturb the algorithm halfway through, perhaps when one participant wanders off to fetch a drink or an external agent rugby-tackles someone mid-move.

Fig. 1. Scheme Bricks interpreter visualization. (© Dave Griffiths)



Nick Collins (composer-programmer), University of Sussex, Falmer, Brighton, BN1 9QJ, U.K. E-mail: <N.Collins@sussex.ac.uk>.

In the main, recent activity in live coding has centered on computers, although there are other paths, from circuit bending to algorithmic choreography. We shall also, further down this supposedly linear text, explore the artistic compulsion to change one's mind mid-progress.

Four representative projects that demonstrate live coding and some of its issues are:

1. The long-running laptop ensemble slub, a London-based sometime-duo, sometime-trio who have been projecting their laptop screens since the year 2000. The projections reveal their homemade software, much of which establishes live codable environments where running programs can be modified in real time. Alex McLean has even gone so far as to write programs that modify their own source code [5].
2. Craig Latta's Quoth environment, which, like a 1980s adventure-game text engine, uses natural language commands to manipulate musical objects and thus demonstrates that not all live coding relies on syntax that is impenetrable to a general audience (many other accessible systems have been constructed, from graphical interfaces to self-documenting object names) [6].
3. Max/MSP battles carried out from a blank starting page within fixed time limits and judged by audience acclaim, for example at the instigation of Marcel Wierckx with his students [7]. Within this visual patching system, participants build up their graphs of objects over time from a large number of types of object and associated parameters available within Max/MSP [8].
4. Taking the Unit Generator Paradigm of computer music software back to hardware: live patching of circuits, as explored by the live electronics ensemble Loud Objects, who wire up circuitry with silhouettes of the action visible via an overhead projector [9].

Many computer environments have been built to support live coding, including systems that have gone on to attract larger user bases, such as ChucK, Impromptu and Fluxus. Any interpreted language can be used for live coding; for instance, the audio programming language SuperCollider has been a popular choice and was a platform for many early experiments. Originally, a few hardy souls were involved in coding with FORTH in the 1980s, including a prototypical onstage live-coding experiment (Ron Kuivila, 1985, working without projection) [10]. However, as examples 3 and 4 above demonstrate, purely textual programming languages are not the only option. Live patching, whether through graphical programming languages or live circuit-building, is often of a less abstract form, although of course the profundity of abstraction is neither the central point of all performances nor the only criterion of success.

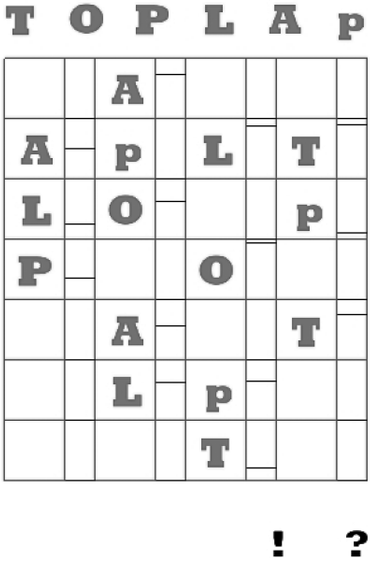
Parallels to live coding (outside of the act of concertizing) can be found in various board and on-line games, where the rule sets themselves become an object of debate and modification. The most famous include Nomic [11], invented by Peter Suber, and card games such as 1000 Blank White Cards or Dvorak (in variants where self-modification of rules is actively allowed) [12]. Such manipulations have also formed the basis of real political action. The People Speak (founded by Mikey Weinkove and Saul Albert) have run town hall–style meetings with computer vision–based voting systems, where the audience must negotiate their own political process to decide issues [13]. Existing large-scale political systems, such as those of entire countries, allow often highly indirect means of eventually adjusting the laws of the land, but with sufficient lag to make their status as "live" coding somewhat more tenuous.

No algorithms, from a computer-science point of view, have to be involved in politicking. However, to give a software analogy, a similar experiment has taken place in the on-line game spring_alpha, which exists in a permanent "alpha state" of re-development, although the live coding is here more a non-real-time infinite design cycle [14]. Google Wave and other real-time collaborative text-editing tools are of interest here as means of keeping rewrites in flow and multiple simultaneous writers empowered. The ensemble PowerBooks UnPlugged [15] code from within an audience, using laptops and their built-in speakers, spatialized by the distribution of band members sharing code via wireless network. Multi-user live coding is a prominent strand that takes the pressure off a single performer and eases algorithmic complexity in a task shared.

In all this, the very division of human decision-making from the rigidity of machine instructions is a continual trope (with human participation under potential erosion in more complex artificial intelligence scenarios). As in the generative arts, the rules allowable to humans are often more ambiguous than anything allowed within computability [16]. If anything, however, live coding's combination of humans acting against their computer programs makes this even more tangled—and excitingly human!

Live coding's precedents intersect with the history of the algorithmic arts, from early algorithmic composition (Guido d'Arezzo in 1026, Athanasius Kircher in 1645, Johann Kirnberger in 1776 and many more [17]) to text pieces and conceptual art to parlor games and conversation. Ancient Greek dialectics [18] and Renaissance mathematical competitions (lasting 40–50 days) have also previously been implicated [19], although a direct anticipation of laptop live coding relies on 20th-century technology. Live coders are highly interested in creative links to their practice [20] and recognize that manifestations of algorithmic structuring occur throughout the arts. Literary influences might include Stéphane Mallarmé, Julio Cortázar's novel *Hopskotch* (1963) or particularly Hermann Hesse's *Glass Bead Game* (1943), which describes a future civilization wherein a complex live game highly influenced by musical theory forms the basis for society. Live coding has its jocular side, even potentially live comedy elements; certainly, musical live coding has been observed to contain many extramusical events (for example, by David Wessel in witnessing a

Fig. 2. The TOPLApapp interpreted instruction sound synthesizer, for iPhone/iPod Touch. (© Nick Collins)



duel at the International Computer Music Conference in 2005) [21]. The movement has also been allied with software art, where a consciousness of the social role of computers and the ubiquity of aspects of programming in modern life underlie some theorizing [22].

Inasmuch as there might be said to be a self-conscious live coding movement primarily motivated by real-time interpreted audiovisual programming environments, activities are traced back to around 2000, when (some) laptop musicians became aware of the need to project their computer screens to avoid the perception that nothing was taking place. Gradually, consciousness of the potential and pitfalls of live coding grew. For instance, slub have always made their own software, but at first Alex McLean more often invoked command-line programs live rather than live-programming an algorithm's internal state itself [23]. The Transnational Organisation for the Promotion of Live Art Programming (TOPLAP, to give one of its interpretative abbreviations) was founded in 2004 at the close of the "changing grammars" meeting organized in Hamburg by Julian Rohrhuber and has been a focal point for much subsequent activity.

Acknowledging such prior existences, which can be pursued in various papers already endnoted, we stumble now into consequential territory.

Fig. 3. Human live coding; Click Nilson follows Matthew Yee-King's coded instructions, in a practice session in front of a BBC camera crew. Nilson attacked the camera within seconds of this shot. (© Nick Collins)

5 Consequences

Type :

Make a sine wave

—or at least its translation, within your favorite computer music environment. Please run your manifestation of this pseudocode. Now consider your next action; what is already running is a sine wave, the parameters of which you may want to tweak. Whether your environment gives you an easy opportunity to do this or not, or whether you anticipated such a need, is a key challenge in practical live coding.

Is it better to have a bird in the hand than two in the cliché? Immediate rewards are not necessarily as compelling as future magnificence, but a trade-off of patience exists in the mind of any normal audience. The more profound the live coding, the more a performer must confront the running algorithm, and the more significant the intervention in the works, the deeper the coding act. In programming, small changes can have grand repercussions; it is not impossible to imagine changing a single character or connection to achieve substantial consequences (for instance, substituting one data array for another via similar names at a key juncture, moving from "for" to "fork" or "and" to "rand," making a single rewiring between waiting complex processes).

Most live coding performances fail to live up to this promise, because sometimes it is hard enough just to get a program running at all let alone somehow to have anticipated enough about the likely program flow to make significant moves. A primary difficulty of much live coding is also common across all live performance: maintaining enough variety to keep an audience engaged. Much live coding involves an element of improvisation rather than exact rote playback, which plays to the strengths of potentiality inherent in any coding moment. Still, as with any improvisation, intensive practice makes for a much better repertoire of gestures for arising situations. An enhanced awareness of the rigor of practice and the demands of concertizing is filtering individuals in through the live coding community as they assess the quality of their performances [24].

Audience expectation has increased as the shock factor (and perhaps the additional leeway) of the early days has diminished; people are now often aware of live coding in advance. Tensions can be palpable as an audience hangs on every keystroke, adding to the tightrope effect for a performer. Such tensions are not reduced by the recurring paradigm of the blank slate, wherein performers attempt to start from a blank page. Mentioned above in the context of graphical patching in Max/MSP, it is a stalwart of all text environments, too, where a blank document forms a strong starting point for a work [25].

6 Change of Mind

Time for a sudden change in essay structure? In one example of self-referential writing, with a host of literary precedents from Shakespeare and Molière to Sterne, John Fowles intervenes in his own book in *The French Lieutenant's Woman* (1969).

7 Change of Mind

Or back again; from one avenue of thought, retreat once more to the originating stream, though perhaps energized anew by the course you found yourself deviating along [26].

8 New Developments from Visualized Techno Compilers to Algorithmic Choreography

Live coding has proliferated, extending many new tendrils. From mobile live coding to projects somewhere between performance art and contemporary dance, the algorithmic literacy of digital arts practitioners matches technology in culture. In this section, some new examples are discussed.

Beginning with some recent software-linked developments, Dave Griffiths has been responsible for some of the most cutting-edge employments of game industry graphical engines for live coding [27]. His various pieces typically involve a limited (yet powerful) control set of operations, often selected via a gamepad, which dynamically complicate the lives of graphical robot agents (*Al Jazari,* 2007), the growth of a plant (*Daisy Chain,* 2008) or the traffic in a virtual city (*JamCity,* 2009). In *Missile Command* (2008), a remake of the computer game of the same name is abstracted from within, heading from video game to audiovisual art piece as the source code is rewritten [28]. In performance with slub, one of his impressive set pieces is a graphical compiler environment used for sound synthesis called Scheme Bricks (2008). The software visualizes the nesting of a program by staggered, rightward-jutting blocks, which flash invoked actions as the program runs over time (Fig. 1).

New platforms are always attractive targets. Before summer 2010, Apple had restricted developers from allowing live scripting languages on the iPhone, potentially blocking live coding applications. Therefore the challenge of creating a live-coding-themed app for the iPhone as an art project seemed irresistible. The TOPLAPapp App, if that's not too chewy a mouthful, is a promotional tool I built for TOPLAP that owes a debt to Griffiths's exploration of small instruction sets and also to the instruction sound synthesis of the 1970s [29]. A grid allows the ordered placement of six letters, in fact corresponding to 11 instructions, since the small p is a modifier to the letter appearing after it. Each instruction is further modulated by the position of an associated slider (Fig. 2). The app was originally released as a puzzle, without instructions, but has since been open-sourced, so that in principle its secrets are available to those who might care to look. The sound synthesis engine exploits the dynamic creation of breakpoints that indirectly specify a time-amplitude waveform, and the choice of the next breakpoint follows from the grid of instructions (the program is the list of letters and their parameters) [30].

From software to wetware: Live coding itself raises consciousness of human beings within the innards of algorithms. The human live coders who flirt within the algorithmic environments, teasing and tinkling the guts of the processes, are the most powerful agents around. Their presence continually reinforces the truism that software is written by people and makes live its construction and deconstruction.

Artists have reflected on the place of the human in the digital arts, guided by a heightened awareness in multiple disciplines of the role of the human body in interaction. The image of the hunched-over laptopist, relatively inert but for dancing fingers, may suggest a central pose but can itself be subverted. Some artists, while still exploring algorithms, have actively removed the computer, or at least transformed its facilitation role.

On the borders of more bodily explicit live coding (as an algorithm rewriting activity), a number of projects can be mentioned. Originally and independently, in a 2007 Hong Kong Max/MSP workshop and at a dorkbot camp near London, some have explored the live patching of bodies analogous to a signal processing network (the active on-the-fly reconfiguration necessary for significant live coding itself is less explored, but the act of patching is itself a highly evocative pedagogical tool) [31]. Using conductive ink, the Humanthesizer <www.bareconductive.com/> recently explored the closing of circuits by human contact, a feat anticipated by 40 years in Bruce Haack's early-1960s Dermatron [32]. Demonstrating once more the links to 1950s happenings and 1960s text pieces, and subsequent rule structures in the improvisation community such as John Zorn's *Cobra* (1984), the ensemble Halal Kebab Hut explore live algorithmic game pieces, where performer actions are dependent on previous actions of their fellows following charts [33]. In Nik Hanselmann's *bodyfuck* (2009), computer vision is used so that the performer's distinct gestures can be tracked as commands in the diabolical programming language brainfuck <www.vimeo.com/7133810>.

Against such a backdrop, I have been actively exploring roles for more humanly active live coding with various collaborators, with explicit modification of algorithmic processes during a performance. With the choreographer Teresa Prima, I ran experiments with a blackboard and dance instructions. With the improvisation ensemble In Sand, I explored instruction-list passing and modification within an ensemble of acoustic musicians. Most intensively, the duo Wrongheaded (a collaboration with Matthew Yee-King) have been performing algorithmic dancing routines that combine computer live coding and human dance (Fig. 3). Human actions are coded on the spot and themselves interfere with the duo's ability to continue working at the computers by which they specify the algorithms.

Although this live coding of human movement has arisen out of an explicitly software-based movement, there are precedents within the world of contemporary dance itself. From the rule-based systems of Cage and Cunningham through Robert Dunn's choreography class [34] to more recent formalist experiments, focus on process has oscillated with that on product. When confronted with live coding, Prima mentioned João Fiadeiro's "real time composition" as her reference point—for example, his piece *Existência* (2002) and its in-the-moment creation with interventions. In the ChoreoGraph system first proposed and implemented by Michael Klien and Nick Mortimore in the 1990s, live scoring of dancer actions is achieved through a software interface, with the working score accessible by the dancers on stage monitors [35]. This is equivalent to the live scoring systems for performers explored in computer music and allows the same intervention in running processes. The extent to which live systems rewriting has been a central feature of choreographic work in contemporary dance is arguable, but there are certainly some analogous structures and devices. When maintaining a separation of musicians and conductors, performers and choreographers, the feedback loops of action to the ability to program new instructions (as found in Wrongheaded's work) are not paralleled. Yet while live coding under the banner of "live coding" per se has proceeded independently, there come moments of reflection and conscious search that suddenly make clear the parallel interests of the two fields.

10 (base 9) Conclusions

Why is live coding of consequence? It touches on many angsts of modern cultural life; lacking a singular rule set for life, it is highly therapeutic to play out many options! Live coding mediates human action within (self-defined) frameworks of rules that are themselves made to be broken, raising both sociopolitical and technological issues. Its relevance to our computer-sped world is hard to deny. There is always one step into abstraction, with direct actions guided by temporary templates. Yet ultimately humans have the advantage of full freedom of action, and live coding is rarely authoritarian in the sense that instructions are unbreakable; perhaps the whole point is to maximize the potential to change one's mind.

Searching for new performance ven-

ues (sweetened by a little Performing Right Society Foundation for new music money), TOPLAP in the U.K. have been diversifying. They have explored a variety of alternative venues, from pubs through planetariums (Color Plate B) to a January 2010 performance in a converted anatomy chamber, here co-opted for the dissection of algorithms. This local activity is accompanied by an international scene where "live coding" has become a common term to add to the categories of work solicited in festival calls. Live coding is not solely a performance bloodsport nor bereft of audience satisfaction, and the modalities of engagement extend beyond mere laptop audio and visuals. There seems to be no end of the potential permutations of the human amongst the algorithms.

Acknowledgment

{inf.do{"thankyou".post}}.fork

References and Notes

Unedited references as provided by the author.

All web links checked 1 November 2009 unless otherwise stated.

1. Adrian Ward, Julian Rohrhuber, Fredrik Olofsson, Alex McLean, Dave Griffiths, Nick Collins and Amy Alexander. "Live algorithm programming and a temporary organisation for its promotion," in *Proceedings of the README Software Art Conference*, Aarhus, Denmark, 2004.

2. Andrew Brown "Code Jamming," *M/C Journal* 9(6) (Dec. 2006) Retrieved 22 Aug. 2009 from <http://journal.media-culture.org.au/0612/03-brown.php>.

3. In a posting to the livecode mailing list, Alex McLean revealed many interesting definitions created by his survey participants in an online study. See "Live coding survey — please respond!" in the archives, 17 September 2009, <http://lists.lurk.org/mailman/listinfo/livecode>.

4. This activity was reputedly carried out at a perl users meeting in a bar, once upon a CPU cycle.

5. Alex McLean, "Hacking Perl in Nightclubs" (31 August 2004). Retrieved 22 August 2009 from <www.perl.com/pub/a/2004/08/31/livecode.html>.

6. Craig Latta, "Quoth, a dynamic interactive fiction system." Retrieved 1 Nov 2009 from <http://netjam.org/projects/quoth/>.

7. A nice video example is provided by Edo Paulus. Retrieved 1 November 2009 from <www.eude.nl/projects/max-live-coding/>.

8. A recurrent issue on the livecode mailing list has been that of graphical versus textual programming. No disparagement of Max/MSP's capabilities is meant in this paper, for without doubt Max/MSP is a sufficiently complex system to make for arresting results and further nests other programming languages within itself (there are some third party externals that house entire other computer music environments, such as rtcmix~ or chuck~. See for instance Brad Garton's work at <http://music.columbia.edu/~brad/software/index.html>.

9. Nicolas Collins, *Handmade Electronic Music,* 2nd Edition (New York: Routledge, 2009).

10. Documented in Curtis Roads, "The Second STEIM Symposium on Interactive Composition in Live Electronic Music." *Computer Music Journal*, 10(2): 44–50 (1986).

11. Douglas Hofstadter. *Metamagical Themas: Questing for the Essence of Mind and Pattern* (New York: Basic Books, 1985).

12. I have often begun talks on live coding by handing out "Live Coding Cards," which read "Pass this card on after modifying this instruction."

13. Saul Albert, "Who Wants to Be? . . . an audience" (2008). Retrieved 7 November 2009 from <http://whowantstobe.co.uk/wwtb-engage.pdf>.

14. Checked 9 November 2009, <www.spring-alpha.org>.

15. PB_UP <http://pbup.goto10.org/>

16. Nick Collins, "The Analysis of Generative Music Programs," *Organised Sound* 13(3): 237–248 (2008).

17. See for example Gareth Loy, *Musimathics*, Volume 1 (Cambridge, MA: MIT Press, 2007) and Gerhard Nierhaus *Algorithmic Composition: Paradigms of Automated Music Generation* (New York, NY: Springer-Verlag/Wien, 2009).

18. Or to extend back further, perhaps 100,000 years, the very dynamism of language is an original source of human agencies outlining actions and revoking them in mid-flow.

19. Ward et al. [1], also available from <www.toplap.org/index.php/Read_me_paper>.

20. <www.toplap.org/index.php/HistoricalPerformances> and <www.toplap.org/index.php/Toplap Papers>

21. Click Nilson. "Live coding practice," in *Proceedings of New Interfaces for Musical Expression (NIME),* New York, 2007.

22. Ward et al. [1].

23. Adrian Ward was concurrently working on a parody of Max/MSP called Map/MSG where boxes contained REALbasic code and could be adjusted in concert.

24. See Note [12] for more on practice. Increased media exposure, for example BBC coverage of live coding from a London gig in the summer of 2009 <http://news.bbc.co.uk/1/hi/technology/8221235.stm>, is a powerful force for introspection on the accessibility and contents of performance.

25. A rarer performance mode called full slate has been broached, though not performed with to any great degree. The easy option of using a preset is a little like full slate coding, in that it starts from a developed patch, but as an opposite to the blank slate, the objective is to remove parts from a complex apparatus in an interesting manner as you work back to emptiness.

26. Here I have sought to change the rules of academic paper writing, but of course the system can bend to accommodate this. In order to allow flexibility to refute my own actions at a future date, I hereby attribute this endnote to ~placeholder, the 11th-century Norman philosopher.

27. They are built with his own general-purpose audiovisual live coding software Fluxus <www.pawfal.org/fluxus/>. For more project details, see <www.pawfal.org/dave/index.cgi?Projects>.

28. Griffiths himself acknowledges the influence of a space invaders remake and Fluxus deconstruction by Gabor Papp (personal communication 10 November 2009).

29. Paul Berg. "Composing Sound Structures with Rules," *Contemporary Music Review* 28(1): 75–87 (2009).

30. This is not the only manifestation of live coding on a phone. Fredrik Olofsson has released a SuperCollider class, RedPDU, which allows code to be written on the phone as an SMS and sent to an external compiler: <www.fredrikolofsson.com/f0blog/?q=node/391>.

31. See <www.flickr.com/photos/newtables/sets/72157600029205941/> and <http://dorkbotlondon.org/wiki/index.php/Human_Patching_Live_Coding>.

32. On a related note, David Tait mentioned to me the example of the remote-controlled choir from the BBC comedy ideas program Genius.

33. Simon Katan, "Halal Kebab Hut—Techniques of Composition" (2004). Retrieved 6 November 2009 from <www.halalkebab.co.uk/writings/HKH techniques.pdf>.

34. Sally Banes, *Writing dancing in the age of postmodernism* (Newhaven, CT: Wesleyan University Press, 1994). Thanks also to Scott deLahunta for discussions and leads on this topic of choreographic precedents.

35. Scott deLahunta, Michael Klien and Nick Rothwell, "Duplex/ChoreoGraph: in conversation with Barriedale Operahouse." 2 May 2002. Retrieved 7 November 2009 from <www.sdela.dds.nl/sfd/frankfin.html>.

Manuscript received 9 November 2009.

Nick Collins is an experienced pianist and computer music performer, active in both instrumental and electronic music composition as well as explicit live coding. Recent concerts have been outdoors in Mexico City, indoors in New York, and at a planetarium in Plymouth, U.K. Further details, including publications, music, code and more, are available from <www.informatics.sussex.ac.uk/users/nc81/>.