What Does Live Coding Know?

Geoff Cox
Aarhus University
gcox@dac.au.dk

ABSTRACT

Live coding can be seen to reflect contemporary conditions in which our lives seem to be increasingly determined by various scripts and computational processes. It demonstrates the possibility for more expansive ideas that emerge in the uncertainties of improvised performance. This paper further situates the practice of live coding in the context of artistic research and the notion of 'onto-epistemology'. One of the challenges, it is argued, is to bring code back into 'material-discursive' practice so that code can be understood for what it is, how it is produced and what it might become. This is arguably the critical task of live coding: to expose the *condition of possibility*; in other words, to remain attentive to the contradictions of what constitutes knowledge in contested fields of practice, and to demonstrate modes of uncertainty in what otherwise would seem to be determinate computational processes.

INTRODUCTION

Live coding has emerged over the past decade as a dynamic creative force that has gained critical attention across cultural and technical fields (Blackwell et al, 2013), but there is little in the way of discussion about its critical potential and ability to generate new forms of knowledge. In this short paper, I want to open up this line of thinking and situate live coding in the context of artistic research and the inter-relation of epistemological and ontological domains.

By invoking the notion of 'onto-epistemology', I refer to the work of Karen Barad (2007), who combines epistemology (the theory of knowing) and ontology (the theory of being) into an ethical framework that she calls 'apparatus' (after Michel Foucault). In her conceptualization of apparatus, she develops an understanding that apparatuses and subject/objects mutually create and define each other. In this way apparatuses are not to be seen as passive articulations of power/knowledge but instead active and 'productive of (and part of) phenomena' (Barad 2007: 98). In this sense an argument can be developed that departs from the anthropomorphic tendency to situate the programmer centrally as the one who introduces uncertainty (as, for instance, through artistic intention or human error, Cox & McLean 2013: 63) to a position that takes into account the wider apparatuses in which human/nonhuman subjects/objects co-create and together establish uncertain outcomes. In other words, the making, doing and becoming of live code, coding and coders are materialized through in the inter-action of elements.

Of course to try to discuss such complex inter-actions is an impossible task within only a few pages of a short conference paper but at least I hope to offer some starting points for further work (not least, in my own contribution to the early stages of a multi-authored book project about live coding). Part of the intention is this paper is simply to demonstrate the potential of live coding to disrupt some of the various power/knowledge regimes through which it circulates as performance, experimental computer music, computational culture, aesthetic expression, and so on. The potential to disrupt expectations and introduce uncertainty is particularly important, as I hope is obvious, as we are increasingly bound to closed and opaque coded systems that do not provide access to any understanding of their inner operations, let alone encourage the manipulation of them at the level of writing or reading code, or through real-time inter-actions and critical experimentation.

One of the challenges, it is argued, is to identify code as an integral part of coding practice so that it can be understood for what it is, how it is produced and what it might become once it is materialized. This is arguably the critical task of live coding: to expose the *conditions of possibility*; in other words, to remain attentive to the contradictions of what constitutes knowledge in contested fields of practice, and to demonstrate modes of uncertainty in what otherwise would seem to be determinate computational processes.

EMERGENT KNOWLEDGE

In unsettling some of the assumptions of the institutions of art and the academy, 'artistic research' (or research through artistic practice) might also be useful to draw attention to the way that live coding refuses to be fixed or foreclosed by a particular knowledge domain (such as computer science, electronic music or computer music). Instead it offers a challenge to some of the assumptions of what constitutes knowledge and how it might be articulated in ways where the outcomes are less certain or prescribed.

The discussion around artistic research has become fairly well established in general with the proliferation of critical writing that addresses non-propositional forms through which knowledge is disseminated in order to accommodate artistic modes of thinking and argumentation (Schwab and Borgdorff 2014: 9-20), but there is little as yet that extends to artists working with computational forms. The suggestion therefore is that non-traditional methods, such as the live inter-actions of programmer, program code and the practice of coding, further expand the range of possibilities for knowledge production and question how 'various communicative and thinking "styles" guide the flow of knowledge' (Holert 2009: 8). Decisions are made, enacted and further adapted in real-time.

Artistic research practice can also be seen part of the more general critique of epistemological paradigms and the possibilities for alternative forms to reshape what we know and how we know it, and begin to define some of the limits of knowledge and what escapes its confines (in the realm of alternative or non-knowledge). It becomes clear that formal epistemologies are inherently paradoxical and alternative paradigms such as live coding understood as artistic research practice might help to reshape how and what we know about how knowledge is produced through reflexive operations and recursive feedback loops. For example, Alex McLean's *feedback.pl* (2004), a self-modifying live code editor written in Perl, has been oft-cited as an example of this by establishing how both the programmer and the program are able to edit code while it runs in real-time (Yuill 2008; Cox 2011; Cox & McLean 2013). In the example it is the inter-relation of apparatuses and subject/objects that create and define each other, and thereby demonstrates how neither the programmer nor program are able to make finite choices or decisions as such.

This is somewhat like 'action research', a reflective process of problem-solving, in which the act of writing or coding is made active to question what it generates in terms of research outcomes. But more than this, live coding presents a further challenge to the conventions of research practices, including those that attempt to incorporate practice as a mode of research that destabilize expectations or goal-oriented approaches to research design, in its embrace of uncertainty and indeterminacy. This is emphasized in the recognition of the 'decision-problem' that helps to define the limits of computation. According to Alan Turing's 1936 paper 'On Computable Numbers, with an Application to the Entscheidungsproblem (Decision Problem)', there are some things that are incapable of computation, including problems that are well-defined and understood. Computation contains its own inner paradoxes and there is always an undecidable effect in every 'Turing Complete' program however decidable it might appear, and it is not logically possible to write a computer program which can reliably distinguish between programs that halt and those that 'loop' forever. The decision-problem unsettles certainties about what computers are capable of, what they can and cannot do.

Live coding might be considered to be a form of improvised action in this way in which the uncertainty of outcomes are made evident through the introduction of further 'live' elements outside the computer. All decisions are revealed to be contingent and subject to internal and external forces that render the performance itself undecidable and the broader apparatus an active part of the performance. It becomes clear that research practices, methodologies and infrastructures of all kinds are part of larger material-discursive apparatuses through which knowledge is produced and circulated. For example, as with all coding practices, the false distinction between the writing and the tool with which the writing is produced is undermined (Cramer 2007), and neither academic writing nor program code can be detached from their performance or the way they generate knowledge as part of larger apparatuses that create and attempt to define their actions (even when they fail). With the example of *feedback.pl*, the subject/objects relations become confused and it becomes evident that these processes are indeterminate and open to further change and modification.

Furthermore, live coding, like all coding, is founded on collective knowledge, or what Paolo Virno (after Marx) would identity as ‘general intellect’ as the ‘know-how on which social productivity relies’, as an ‘attribute of living labour’ (2004: 64-5). This know-how refers to the combination of technological expertise and social or general knowledge, and recognizes the importance of technology in social organization. This is something I have written about previously, to describe how the contemporary linguistic and performative character of labour and coding practices combine to produce an uncertain relation between code objects and code subjects — between code and coders – the ‘means-end’ of code (Cox 2011). The politics of this extends to various communities of socio-technical practice such as the way that source code is distributed and shared through the social movements around software (Free Software) and various legal instruments (such as copyleft software licenses). It also extends to the use of code repositories like Git and GitHub that offer distributed revision control and source code management, including collaborative working tools such as wikis, task management and bug tracking (https://github.com/). Such examples exist as part of a political spectrum of ‘just-in-time production’ where performative and improvisational techniques have been put to work.

There is an excellent essay by Simon Yuill “All Problems of Notation will be Solved by the Masses” that develops similar connections to free software development and collaborative practices, including comparison with the improvisation techniques of the Scratch Orchestra in the late 1960s: ‘Livecoding works from within a particular relation between notation and contingency. The specificity of code is opened towards the indeterminism of improvisation.’ (Yuill 2008) Indeterminacy extends to the way that knowledge is produced too through improvisatory techniques, and the discussion of artistic practice can be partly characterised in this way as a means of process-based knowledge production.

If reference is made to Foucault’s *Archaeology of Knowledge* (1972), this contingent aspect is already established perhaps, and thereby what constitutes knowledge is open to question, and various stratgeies are developed to uncover otherwise undiscovered or emergent knowledge. Yet, additionally, it is necessary to also conceptualize knowledge that is less human-centred to include ways of understanding and acting in the world that exceed what is normally seeable, readable and knowable. This requires ‘media’ and not just humans becoming ‘active archaeologists of knowledge,’ as Wolfgang Ernst puts it (2011: 239). Thus what is already understood as *archaeology of knowledge* is extended to *media archaeology* and beyond the human sensory apparatus to the nondiscursive realm of technical infrastructures, and including computer programs. Live coding might similarly expand the range of possibilities for alternative epistemologies in allowing for discursive and *nondiscursive* knowledge forms.

The example Ernst gives is ‘Fourier analysis’ as able to isolate individual components of a compound waveform, concentrating them for easier detection or removal. He considers the ‘operative machine’ as able to perform a better cultural analysis than the human is capable of (Ernst 2011: 241). Furthermore, with reference to Ernst and the concept of ‘microtemporality’, Shintaro Miyazaki introduces the concept algo*rhythm* to emphasize how the very processes of computation possesses epistemic qualities. He explains:

‘An algo*rhythmic* study cultivates a close reading of, or to be more accurate, a close *listening to* the technical workings of computers and their networks on precisely this deep level of physical signals. Analysed on different scales from the microprogramming on the level of the CPU, or the physical layer of communication protocols between server and clients, to interface applications on all levels where human interaction is needed, to longer time spans such as the daily backing up of large amounts of data, distributed and networked algo*rhythmic* processes can be found on many levels of our current informational networks and computational cultures.’ (Miyazaki 2012)

The algo*rhythmic* approach offers an epistemic model of a machine that makes time itself logically controllable and, while operating, produces measurable effects and rhythms. The practice of live coding – and more overtly ‘Algoraves’ – resonates with these descriptions in drawing together dynamic processes, signals, rhythms, temporalities and moving bodies. Part of this involves the necessity of developing a fuller understanding of onto-epistemological operations. Without this know-how, we would simply miss the detail on how various effects are produced, how human/nonhuman relations and agencies operate together, and how codes are embedded in wider systems of control and command. In *Philosophy of Software*, David M. Berry suggests that we need to pay more attention to the "computationality of code", to understand code as "an ontological condition for our comportment towards the world" (Berry 2011: 10, 13). If code becomes the object of research in this way, its ontological dimension becomes crucially important for the way that code is understood as a condition of possibility (Ibid: 28). An onto-epistemology of live coding would therefore involve inter-actions of discursive and nondiscursive forms to challenge how we conceptualize code-being and code-becoming. Yhe point of this essay is to stress the ways that code performs and is performed through the practice coding, and together how both operates onto-epistemologically.

UNCERTAIN ENDING

The code performs with the coder, inter-acting in uncertain and indeterminate ways. Indeed it is with his condition of uncertainty that I would like to end this essay echoing Ernst's reference to stochastic mathematics, to stress that there is an indeterminism between human and nonhuman knowledge that comes close to the uncertainty principle (2011). The 'uncertainty principle' asserts that no thing has a definite position, a definite trajectory, or a definite momentum, and that the more an attempt is made to define an object's precise position, the less precisely can one say what its momentum is (and vice versa). Things are known and unknown at the same time.

Foucault's notion of apparatus is useful in this way as it allows for an analysis of power/knowledge through understanding things to be part of larger relational systems. However Barad extends this definition by insisting that apparatuses are 'themselves material-discursive phenomena, materializing in intra-action with other material-discursive apparatuses' (Barad 2007: 102). To explain in her words:
'Although Foucault insists that objects (subjects) of knowledge do not preexist but emerge only within discursive practices, he does not explicitly analyze the inseparability of apparatuses and the objects (subjects). In other words, Foucault does not propose an analogue to the notion of phenomenon or analyze its important (ontological as well as epistemological) consequences.' (Barad 2007: 201)

The onto-epistemological reading insists on broader notions of subjects and objects and their inter-actions. This is made clear with respect to how technologies allow for discursive and nondiscursive knowledge forms. Again Barad makes this apparent:

;Knowing is not about seeing from above or outside or even seeing from a prosthetically enhanced human body. Knowing is a matter of inter-acting. Knowing entails specific practices through which the world is differentially articulated and accounted for. In some instances, 'nonhumans' (even being without brains) emerge as partaking in the world's active engagement in practices of knowing. […] Knowing entails differential responsiveness and accountability as part of a network of performances. Knowing is not a bounded or closed practice but an ongoing performance of the world.' (Barad 2007: 149)

Drawing on the 'new materialism' of Barad, and the media archaeology of Ernst and Mayazaki, opens up ways to signal that it is simply not possible to generate knowledge outside of the material and ontological substrate through which it is mediated. Thus the inter-relation of epistemology and the ontological dimension of the materials, technologies, things, and code is established as active and constitutive parts of meaning-making. Live coding offers a useful paradigm in which to establish how the know-how of code is exposed in order to more fully understand the experience of the power-knowledge systems we are part of, and to offer a broader understanding of the cultural dynamics of computational culture as part of the apparatus with ontological as well as epistemological consequences.

Live coding can be seen to reflect contemporary conditions in which our work and lives seem to be increasingly determined by various scripts and computational processes. More expansive conceptions and forms that emerge in the uncertainties of performance become part of the critical task for live coding: to expose the conditions of possibility, to remain attentive to the contradictions of what constitutes knowledge and meaning. An onto-epistemology of live coding thus offers the chance to engage with the inner and outer dynamics of computational culture that resonates in the interplay of human and nonhuman forces. It helps to establish uncertainty against the apparent determinism of computation and how we think we know what we know.

REFERENCES

Barad, K. 2007. *Meeting the Universe Halfway*. Durham & London: Duke University Press.

Berry, D. M. 2011. The *Philosophy of Software: Code and Mediation in the Digital Age*. Basingstoke: Palgrave Macmillan.

Blackwell, A., McLean, A. Noble, J. and Rohrhuber, J. eds. 2013. *Dagstuhl Report: Collaboration and learning through live coding*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=13382

Cox, G. 2011. 'Mean-End of Software.' *Interface Criticism: Aesthetic Beyond Buttons*. Pold, S. and Andersen, C. U. eds. Arhus: Aarhus University Press, 145-161.

Cox, G. and McLean, A. 2013. *Speaking Code: Coding as Aesthetic and Political Expression*. Cambridge, Mass.: MIT PRess

Cramer, F. 2007. '(echo echo) echo (echo): Command Line Poetics.' http://reader.lgru.net/texts/echo-echo-echo-echo-command-line-poetics/

Ernst, W. 2011. 'Media Archaeography: Method and Machine versus History and Narrative of Media.' *Media Archaeology: Approaches, Applications and Implications*, Hutamo, E. and Parikka, J. eds. Berkeley: University of California Press.

Foucault, M. 1972. *The Archaeology of Knowledge*, London: Routledge.

Holert, T. 2009. 'Art in the Knowledge-based Polis.' *e-flux journal* 3. http://www.e-flux.com/journal/art-in-the-knowledge-based-polis/

Miyazaki, S. 2012. 'Algorhythmics: Understanding Micro-Temporality in Computational Cultures.' *Computational Culture* 2. http://computationalculture.net/article/algorhythmics-understanding-micro-temporality-in-computational-cultures

Schwab, M. and Borgdorff, H. 2014. 'Introduction.' *The Exposition of Artistic Research: Publishing Art in Academia*. Leiden University Press.

Turing, A. 1936. 'On Computable Numbers, with an Application to the Entscheidungsproblem (Decision Problem).' https://www.wolframscience.com/prizes/tm23/images/Turing.pdf

Virno, P. 2004. *A Grammar of the Multitude: For an Analysis of Contemporary Forms of Life*. New York: Semiotext(e).

Yuill, S. 2008. 'All Problems of Notation Will be Solved by the Masses.' *MUTE*.
http://www.metamute.org/editorial/articles/all-problems-notation-will-be-solved-masses