Book review

Alan F. Blackwell, Emma Cocker, Geoff Cox, Alex McLean and Thor Magnusson, *Live Coding: A User's Manual*. Cambridge, MA: MIT Press, 2022. ISBN: 9780262544818. doi: https://doi.org/10.7551/mitpress/13770.001.0001

What is live coding? This is the question at the start and at the heart of *Live Coding: A User's Manual*, by Alan Blackwell, Emma Cocker, Geoff Cox, Alex McLean and Thor Magnusson. The authors offer a fairly direct answer in the first chapter:

> Live coding involves showing the screen or making visible the coding process as part of a live performance. Broadly speaking, it describes the improvisatory real-time composition of predominantly computer-generated audiovisual material, in which the writing of code itself (or other executable instructions) is presented as a live event for an audience. Alongside witnessing the coder engaged in the live act of coding (laboring at their laptop), the code itself is also presented – typically projected – in real time as it is being worked on as a visible part of the performance. (p. 3)

This may suffice to give the unfamiliar reader some idea of what live coding looks like in practice. The cover art, by Joana Chicau, also illustrates the idea by depicting the code used to generate it. However, the general aim of the book is to consider live coding as expansively as possible, investigating it less as a particular technique or method than as a philosophy or critical orientation towards art, technology and their relationship. Thus, the question 'What is live coding?' and its implications pervade the book's 352 pages.

Chapter 1 serves as an introduction to both live coding and to the book itself. It begins by disabusing the reader of any confusion regarding the title: 'If you are expecting a conventional user's manual, then put this book down' (p. 1). The authors explain that the book is not a user's manual *for* live coding, but rather an exposition of how live coding *itself* can serve as a user's manual. They proceed to address the fundamental question 'What is live coding?', offering general phrases such as 'writing software in real time', 'changing rules while following them', '*conversational programming*', 'thinking in public' and '*on-the-fly* or *just-in-time* performance', while noting the difficulty (and danger) of attempting to define a practice that has self-redefinition at its core. It seems easier to say what live coding is *about* than what it *is*: 'Live coding is about people interacting with the world, and each other, in real time, via code. Live coding is about making software *live*' (p. 2). This latter statement refers to live coding's 'critical orientation toward the otherwise conventionalized work of programming and software engineering' and how it poses 'questions and challenges to some of the underpinning values and ideologies of a wider computational culture' (p. 3). Live coding shows the screen and 'unveils the underlying operational layer of activity beneath … computational performance' (p. 4), a 'critical gesture' in opposition to the tendency towards smart devices that are ever more imperceptibly integrated into life and thus ever harder to question. The authors 'argue that all performance practices (including music) offer a special way of understanding software and that this has radical potential for all software – not only for artists, their audiences, and art theorists but also for engineers, philosophers, and activists' (p. 5). In short, 'live coding helps us to gain new insights about what software can be' (p. 6).

Chapter 2 explains where live coding came from, tracing its *Partial Histories* through interviews with 17 live coders (or live coding groups) and situating it within the broader histories of computer music, electronic music and personal computing. The authors follow the artistic and technical interests of a disparate generation of 'geeky artists' from their formative years in the 1980s into the early 2000s, a time when 'code was slowly entering public consciousness' (p. 19). Adrian Ward and Alex McLean formed slub and wrote a 'Generative Manifesto', venues and festivals served as sites for art activism, James McCartney demonstrated SuperCollider at ICMC 2000, and Julian Rohrhuber released the Just In Time programming library (JITLib). In this heady atmosphere, Rohrhuber and Renate Wieser convened the Changing Grammars symposium in 2004, a 'live audio programming symposium' that became 'the watershed moment of live coding, as the meeting at which [the Temporary Organisation for the Promotion of Live Algorithm Programming] was formed and immediately after which the TOPLAP manifesto was first drafted' (p. 21). In addition to this personal narrative, the authors point to precursors such as Forth (especially Doug Collinge's Moxie package for timed procedures and Dave Anderson and Ron Kuivila's work on threading), SuperCollider (inheriting from both the MUSIC-N family of computer music languages and Smalltalk), the *Morpheus* generative album, and Rohrhuber's work on proxies.

The remainder of the chapter addresses the growth of the live coding community, issues of inclusion and diversity (further addressed in Chapter 8), institutions and hierarchies, the phenomena of algoraves, and the future of live coding.

Where Chapter 2 traces the paths of a relatively small set of individuals and ideas into the past, Chapter 3 changes tack and presents an expansive view of live coding in the present. The *Expositions* consist of statements and photographs from 42 different live coders (or groups), from Aji to Xambó. Though necessarily incomplete (much like the preceding *Partial Histories*), this collection nonetheless gives an impressive sense of the wide variety of people, places and perspectives involved in live coding today. The group comprises artists, composers, dancers, educators, instrumentalists, language designers, musicians, organisers, performers, poets, researchers and tool builders. Although no formal prompts or questions are provided, the practitioners' statements share recurring themes: Why and how did they get into live coding? How is it integrated into their practice? What is its value? How does the technical context affect the quality or reception of the music (or visuals, dance, or poetry)? What are the key challenges (especially social and cultural) for the live coding community? What may become of live coding in the future – what unrealized potential or unmet hazards await? The live coders offer a range of answers to these questions and more in this fascinating set of expositions.

Chapter 4 shifts from the book's practice-focused portion to its speculative side, which subdivides neatly into *concept* chapters (notation, liveness, time-criticality) and *philosophy* chapters (epistemology, politics). Chapter 4 is the first concept chapter, concerned with *Notation*, how it 'resonates with different meanings and values within different disciplinary traditions' (p. 128), and the implications for the interdisciplinary practice of live coding. Live coding is a 'supremely notational practice' in that code is 'even more explicit than staff notation', and yet it 'finds itself caught between two worlds: it is too ephemeral to be score-based culture and yet too centered on text to be oral culture' (p. 130). Like Chapter 1, Chapter 4 tries to give a sense of what live coding *is* by saying what it is *about*: '*disrupting* the deterministic logic between notation and process, bringing it into a creative feedback loop' (p. 131) of improvised editing, with the result that live coders 'do not become coded but rather *embody* code' (p. 132), working in notation without following it themselves. The authors connect live coding to live art, weaving and the history of notation to support their argument. They go on to establish the significance of the 'environment in which live coders express themselves' which 'exemplif[ies] certain views on what is important' (p. 134) and 'prescribes a particular world of possibilities' (p. 146), as in TidalCycles's emphasis on pattern, supported by *mininotation*, or Orca's spatial grid, where control can flow in any direction. Other topics include the potential for predictive, self-modifying and recorded live coding; the exploratory role of notation in live performance; the distinction between *prescription* and *perception* (as the results of executing prescriptive notation may nonetheless be surprising); and issues of authenticity, vis-à-vis Nelson Goodman's notions of *autographic* and *allographic* art. Finally, they discuss notation as a social project, drawing on Christopher Alexander's work on pattern languages, Adorno's views on the centrality of scores, and Cornelius Cardew's Scratch Orchestra, concluding that live coding offers 'radical potential' to 'keep it live' in music and other notational forms because it is simultaneously 'notation and execution' (p. 157).

Picking up on Chapter 4's final theme, Chapter 5 deals with *Live Coding's Liveness(es)* and 'how diverse live coding practice – improvisational and compositional – enrich a wider theoretical debate on the issue of liveness' (p. 159). As in Chapter 4, live coding 'requires that its very liveness be understood from more than one epistemological and ontological perspective'. Accordingly, much of the chapter is devoted to examining live coding through different lenses: ontologies of liveness, experiences of liveness, concepts of *vitality* and *flow*, and technological 'degrees of liveness'. The authors observe that 'speed and immediacy can easily become mistaken for liveness', resulting in an emphasis on efficiency and flow over risk, reflection, and 'the critical value… within moments of delay and technical resistance' (p. 170). The authors identify this confusion with the 'fetishization' of immediacy over liveness and 'the loss of reflection space and intervals' in contemporary life, asking 'How can the relation between liveness, immediacy, and efficiency be uncoupled?' Their answer is live coding's capacity for risk and serendipity, as code is 'a material (or even collaborator) that the coder works with and can be surprised by' (p. 171). Furthermore, live coding is 'a critical practice engaged in the interrogation and exposition of its own means of production' that (per Simon Yuill) 'creates a virtue by exposing something that is normally concealed', reflected in the showing of the screen (p. 179). The authors connect this to the community's '[s]trongly held copyleft and creative commons principles' that 'underpin live coding as a community in "common"' (p. 178). Other themes in this chapter include human–machine entanglement, the body, performance practices beyond music and visuals (such as live art and choreography), and the performativity of code.

https://doi.org/10.1017/S1355771823000432 Published online by Cambridge University Press

Chapter 6 proceeds from liveness to *Time Criticality in Live Coding*, focusing on 'the disjunction between algorithmic or machine time and the performer's embodied experience of lived time' (p. 181). As in Chapter 4, live coding languages play a significant role, as they 'embody different technical and indeed philosophical understandings of [their] creators' (p. 183), contrasting with the logic of time in imperative languages and addressing the 'fundamental *incommensurability* lying at the heart of live coding' (p. 185): 'changing rules as they are followed' while satisfying history and causality. Per Rohrhuber, the two basic approaches are the *state* picture ('continu[-ing] with the state left behind by the previous version of the code'), found in live coding systems such as Extempore and Sonic Pi, and the *causal* picture ('recalculat[ing] the history of the process as though the code has always been as it is now'), realized in various ways by the SuperCollider (with JITLib), TidalCycles and Hydra environments. Human perceptions of time present their own difficulties, as they are shaped by 'competing – even contradictory – rhythms, regulations, and philosophical rhetoric' (p. 187) and reshaped by contemporary technologies that give rise to the 'work-life indeterminancy' endemic to modern '*liquid times*' (p. 188). In response, the authors ask how live coding can 'contribute toward new understanding about our contemporary temporal experience' and 'draw together human and machine registers of time in ways that are not reductive to either' (p. 203). The ensuing discussion draws on the philosophies of Bergson, Husserl and Heidegger to establish the notion of a 'three-part present' (p. 192), which the authors argue finds expression in live coding as the performer attends to the past-of-the-present (code history, the buffer), present-of-the-present (current state) and future-of-the-present (scheduled or soon-to-be-executed code) continually while performing. Further discussion touches on non-Western conceptions of time, *kairos* vs. *chronos*, '*slow coding*' in opposition the 'accelerated temporalities of contemporary life' (p. 199), and the problem of ending (linking the halting problem to the 'endless loop' of live coding systems).

Chapter 7 moves from performance to philosophy by asking *What Does Live Coding Know?* (not what live coders know, as might be addressed by a conventional user's manual). It begins with the history of programming, from the era of mainframes and 'formal processes of bureaucratic translation' (p. 207) through sweeping changes brought about by falling hardware costs and a 'transformed understanding of the practice' (p. 208) associated with 1960s counterculture and the research agendas of Douglas Engelbart and later Alan Kay. The twin threads of Smalltalk (to GUIs, agile programming, wikis, and software design patterns) and Lisp (to the read-eval-print loop or REPL, Emacs, and the free software movement) are identified as crucial lineages in the development of live programming, wherein 'creative engineers . . . engage in craft practices of modifying the tools they use for their own work' (p. 212). These developments lead into a wider discussion of programming as craft, drawing on theories of craft and creative knowledge, culminating in the conclusion that 'live coding *is* a craft – and given this craft, it seems that code must be its material' in spite of its apparent immateriality (p. 217). A related question is the purpose of code: 'for live coders, code is not an instrument of control for imposing order on a chaotic world', as in software engineering, but instead 'an activity that generates chaos in a highly technical world', 'creative rather than regulative' (p. 213). This analysis shifts the conversation from what live coding *knows* to what it *does*. The authors argue that it 'pressures the *if-then* thinking of computational logic toward the *what-if* of speculative experimentation' (p. 219), that it has the critical potential 'to speculate on the emergent forms of human and non-human knowledge' that result from feedback loops between humans and 'epistemological technologies' such as the computer, and that it 'offers a useful paradigm in which to establish how the know-how of code is exposed in order to more fully understand how code subjects and objects mutually create and define each other' (p. 228).

After seven chapters addressing 'What is live coding?' in seven ways, the eighth poses the question politically, asking *What Does Live Coding Want?* in the world. The authors paint a picture of that world as one of 'corporate data repositories, increasing levels of automation, and global digital infrastructure' (p. 229), where 'the big-data revolution [extracts] value from creative work to enrich those who own the infrastructure' (p. 234), with 'ever-increasing global precarity' and overlapping crises (p. 242). In the face of all this, what can live coding do? According to the authors, a great deal: 'a radically open aesthetic practice', it can 'subversively undermin[e] some of the more insidious infrastructural and ideological values inherent to computational culture' and help 'mobilize an engagement with social justice in the face of algorithmic corporate regimes and governmentality that threaten to undermine freedom of thought and action' (p. 231). It 'reverses the homogenization of creative technology, particularly music technology where intelligent beat trackers, arpeggiators, autotuners, mastering tools, or harmonizers present music that has been algorithmically averaged to meet audience expectations' (p. 234). It accommodates alternative visions for machine learning, artificial intelligence and smart devices that empower users instead of controlling them, offering 'a way to keep humans engaged as authors rather than supervisors of automation' and 'bring back [technologies] from the world of mass production and control to the world of craft' (p. 237). The authors temper this

world-saving spirit with a sober view of the limitations of movements such as free/libre open-source software, open science, and live coding, drawing attention to how 'the ideals of openness are easily co-opted for purposes of oppression' (p. 238) and to the '*long networks* in which live coding participates and on which it depends', from open-source software to mineral extraction (p. 239). Reiterating the challenges facing the wider world, the authors acknowledge that it is 'tempting to see conditions of practice as . . . outside of one's influence or control' (p. 243) but urge against apathy: live coders can create conditions of practice by building the (virtual and physical) environments that foster and shape communities.

In live coding, 'patterns may unfold over multiple scales' (p. 154), and the same goes for *Live Coding: A User's Manual*. Several themes recur throughout the book, including human–machine entanglement, community & inclusivity, and the difficulty of pinning live coding down. Indeed, in the introduction the authors declare that 'it is this pluriversal capacity of live coding to resist or trouble any easy classification, categorization, or explanation that we take as our provocation for (the impossibility, or at least the challenge of) writing this book' (p. 1). This capacity manifests in every chapter: in the many possible answers to 'What is live coding?' in Chapter 1, in the partial histories and paths of Chapter 2, and in the diverse practices and paths in the expositions of Chapter 3. But it is especially present in the second, 'speculative and conceptual' part of the book. The authors weave a rich tapestry of references, unspooling a formidable body of work in order to consider their subject from multiple critical perspectives. This exposes the curious reader to many ways of thinking and offers numerous threads to follow. In some places, however, the sheer density of references and shifts in viewpoint risks tangling up the reader in their dizzying warp and weft.

This likely results from the coauthoring process, which the five authors discuss candidly in the first chapter

> Each and every chapter has been coauthored in various ways. Certainly, this has been a complex undertaking. At times, different contributing voices become tangible through a perceived shift in style or semantics or through the meeting of different – even seemingly incompatible – references or ideas. . . . In places, the transition from one voice to another might appear smooth and seamless; elsewhere, the reader might notice the sudden break or change in tack. (p. 7)

And the last:

> . . . we have explored ways for bringing different and diverging ideas and perspectives on live coding into dialogue without homogenizing them within a single authorial view . . . As the book evolved, the differentiation of individual authorial voice became more complicated (woven together) through the collaborative process of coproduction and reciprocal *thinking with*. (p. 242)

This complication reflects both the difficulty of composing a monograph with five authors and the difficulty of writing a book about live coding. As the authors note, 'live coding can be conceived of as a practice of *complication*', from 'the Latin *plicare*, meaning "to fold, weave". *Complicate* then means to fold or weave together' (p. 241). Thus, the authors weave together 'diverse practices and theories', reflecting live coding's operation 'at the threshold between these different practices and disciplinary perspectives . . . in the style of the book itself' (p. 7). This complication is a source of strength and also ambivalence; without a 'single authorial view', it falls to the reader to see five views at once.

Live Coding: A User's Manual is open-access, available under 'a copyleft CC-BY-SA license that in so many ways reflects the ethos of the live-coding community' (p. xv). (Read it at livecodingbook.toplap.org or the MIT Press.) The authors draw a parallel between the book and live coding itself, which 'is process driven, endlessly subject to revision and modification – like this book, which remains open to further development through the use of creative commons licenses' (p. 231), and they 'offer the various chapters as necessarily works in progress, open to further timely updates, that reflect the subject matter of the book as a dynamic form and practice' (p. xv). This is echoed in the book's closing words: 'Live coding wants to be undone, unraveled, unwoven, and rewoven anew' (p. 243).

I have reviewed the book in this spirit, unweaving and reweaving fragments into a scrap that hopefully gives some sense of the whole cloth. Although it is by no means concerned solely with sound, I expect that readers of *Organised Sound* will find *Live Coding: A User's Manual* an intriguing and inimitable monograph that is grounded in the sonic arts, with particularly rich connections to the history of computer music in Chapter 2, to algorithmic music, the history of notation, and art theory in Chapter 4, and to media theory and performance research in Chapter 5, and insightful reflections on the interweaving of art, technology and society throughout.

Ian Clester
Email: ijc@gatech.edu

https://doi.org/10.1017/S1355771823000432 Published online by Cambridge University Press