This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

8 What Does Live Coding Want?

In this final chapter of the book, we shift our attention from what live coding does or knows to the question of what it wants.[1] In characterizing the discussion this way—as if live coding were a living entity with its own desires—we extend the various threads across the chapters relating to human and nonhuman agency and how live coding might provide insights into its own future development. The question of what it wants is in keeping with our acknowledgment of the performativity of code but also with its underlying sense of purpose in the world. In this way, throughout various chapters we have tried to allow live coding to speak for itself. Initially, we included other voices from the live coding community, and subsequently, we addressed the expressive performativity of code itself—its livenesses and time criticality. To put it in Barad's terms, performativity takes place across the space and time of live coding and at different scales that link together its various intra-active elements.[2] As such, we think that live coding can help to reveal some of the details and significance of these relations (or entanglements), as well as demonstrate its potential to invent new forms of practice.

This last chapter speculates about some of these possible futures but also attempts to address some of the issues that we think merit further reflection as the practice of live coding develops. What is broadly at stake is a deeper exploration into the ways that space and time are constructed when live coding, especially in the context of corporate data repositories, increasing levels of automation, and global digital infrastructure. One thing that live coding helps to establish is that algorithms, increasingly seen to manipulate and control our social systems, are themselves recipients of live updates. Neither humans nor algorithms operate autonomously. They do so in relation to other entities and as part of larger sociotechnical assemblages and infrastructures that are constantly evolving and subject to changing conditions. In other words, both humans and algorithms do things in the world, but it is necessary to understand them as part of broader ecologies in order to comprehend their sense of agency (and environmental consequences), as well as their coconstituted agency when they perform together.

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

In the context of live coding, the phenomenon of the algorave resonates with this description, as an unruly event in which people are encouraged to dance to music generated from algorithms.[3] The implications of manipulating algorithms in the context of dance culture take on political significance. Dance culture, outside of its formal versions, is often considered to be a site of resistance to dominant forms of language, public assembly, and social behavior (dancing in the street as a form of protest, for instance). The UK rave scene, emanating from electronic dance music genres of house, acid house, and techno in the late 1980s, developed a countercultural reputation for the use of illegal drugs (such as LSD and ecstasy) and for subversively convening parties at unauthorized venues.[4] The algorave seems somewhat distant from these anarchist impulses and the sociopolitical movements behind Chicago house and Detroit techno, for example,[5] and yet implicitly holds on to some of the subcultural spirit in messing with normative behavior and rules at the level of the algorithm—in terms of the experience of ecstasy as alternative reality. In other words, dancing to different "algorhythms"[6] can be extended to broader imaginaries and instances of power.

The politics of live coding is there to be further performed, not least in the context of developments in machine learning and its predictive capacity. Here, again our starting question is pertinent: Adrian Mackenzie asks what machine learning wants.[7] The anthropomorphism of the machine in this sense—although problematic in other ways—draws attention to the underlying sense of purpose in these algorithmic procedures. If the machine could answer the question about what it wants, what would it say?[8] Mackenzie's point is that these predictive techniques demonstrate "operational power" that "generate statements and prompt actions in relation to instances of individual desire."[9] The techniques indicate something about how machine learning is generalized and how it produces subjectivities and the desire to predict particular things generated from data sets that are already ideologically determined. Machine learning is *found wanting*.

A tension emerges around the divisions of creative labor here—between programmers and machines and how they program together and apart. Rather than the work of traditional programming, what becomes even more challenging is the way in which automation has taken new forms in the field of artificial intelligence (AI) and in the subfield of machine learning, where it becomes far harder to differentiate what and who is doing what.[10] Given that machine learning is derived from the logic of calculation,[11] the perceived power of algorithms rests on an understanding of how they operate, which is made clear in Mackenzie's book *Machine Learners*, in which he examines specific algorithms and data practices to understand the particularity of human-machine relations and their transformations.

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

The assumption, as Mackenzie also points out, is that everything that exists becomes reducible to stable and distinct categorization.[12] Live coding is a practice that operates in a rather different register and troubles classification in multiple ways, refusing to be easily fixed, defined, or generalized. As we have outlined in earlier chapters, live coding opens up new ways of thinking about notation, liveness, and temporality. It also involves the misdirection of technological know-how, with human and nonhuman agencies engaging in a collaborative and coemergent process of experimental performance that embraces uncertainty and failure, risk, and surprise. Live coding thus emerges as a radically open aesthetic practice capable of destabilizing and disrupting the instrumentality and determinism of algorithms and data structures and of subversively undermining some of the more insidious infrastructural and ideological values inherent to computational culture.

In the following we outline some of these political challenges for live coding and suggest that it can help to mobilize an engagement with social justice in the face of algorithmic corporate regimes and governmentality that threaten to undermine freedom of thought and action. As a dynamically creative and generative practice, live coding offers a means of appropriating the algorithm for different ends or even to stress *means without end*.[13] Its potential politics comes from its ability to expose the differentials of power, materiality, subjectivity, agency, and so on and how meanings are produced through reconfigured human-machine relations. Although we have tried to pay attention to some of these grand challenges in terms of intersectional politics and postcolonial, minority ethnic, and disadvantaged narratives, clearly there is far more work to be done.

This final chapter does not offer solutions but further develops the problems in the spirit of criticality: first, situating live coding in the broader context of data structures and developments in machine learning; second, addressing the colonial legacies and inherent exclusions of algorithmic culture; and finally, returning to the title of the book to stress that live coding is about people interacting with the world, and each other, and about social relations and their transformation. As we have already stated, the liveness of live coding invites reflection on what it means to be alive—to have bodies and to interact with other humans and nonhumans, including machines. We stress that live coding is neither deterministic nor universalist in this way and how, by its very nature, it 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—offering a means for thinking and doing in the world that is not constrained or prescriptive in its direction or purpose. These bold statements need further unpacking, of course, so first we address the broader context of data infrastructures

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

and automation and how live coding helps to establish that algorithms are not as fixed or as deterministic as they might first appear. What live coding wants, then, can only be uncovered by engaging more fully with the sociotechnical conditions in which it arises.

Data Structures and Algorithms

Live code drills down and digs into the logic of digital systems.[14] It presents them for inspection and offers an alternate universe in which artists attempt to take control. Two fundamentals of computer science are, first, data structures—data organized in a certain way—and, second, algorithms, or rules for manipulating that data.[15] The two are often permeable or interdependent, with rules defining structure and structure establishing the rules for action. However, as argued by theorist Andrew Goffey, it is important to remember that algorithms are also culturally situated as linguistic sites for pragmatic action.[16] It may once have been the case, in classic cybernetic systems, that the programmed goal state would (in principle) be open to inspection. But a broader understanding of algorithms leads us to ask about the ways in which rules and structures work together and how they might be manipulated.

For anthropologist and anarchist activist David Graeber, the algorithmic rules from which society constructs bureaucratic systems are a form of *structural violence*, fundamentally opposed to the freedom that is both allowed and celebrated in human play.[17] Whether by left- or right-wing politicians or by Libertarian Silicon Valley corporations, human words and actions are structured into algorithmic courses and categories. Modern markets and democracies supposedly act in the interests of the public—but this so-called public is itself a structural artifact, brought into existence by the infrastructural mechanics of electoral systems, opinion polls, and media channels. Now in the early 2020s, disclosures of covert political influence wielded via social media platforms, such as Facebook,[18] generate widespread anxiety over the continued rise of extreme far-right populism while newspaper headlines state "Why 'Ditch the Algorithm' Is the Future of Political Protest,"[19] in relation to systems of educational quantification disrupted by the COVID-19 pandemic.[20]

Live coding, in this analysis, is not simply a poetic or performative allusion to the hacking of cybernetic infrastructure. As well argued by Jacques Attali,[21] music, through its abstract form, is able to explore and experiment with structures that are later adopted in the economic and social domains. Here, live coding brings into being new kinds of rules in which the structures being made are intended to be transient—an invitation to play, not to build. Live coding is a political demonstration of restructuring, not purely

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

deconstructing. Live coding projects such as Rodrigo Velasco's on-the-fly code poetry and Kate Sicchio's Terpsicode make this clear when they step outside the conventional digital realm, producing rules for the dancing body, for tangled threads, for theoretical texts, or for social processes.[22] These instantaneous bureaucracies are generative, producing new experiences rather than being subjected to existing structures. Could there be a world in which artists make the rules? Graeber contrasts the structural violence of bureaucratic systems with the creative freedom of the anarchist or artist. He claims that the true spirit of the Left is not to be found in a Socialist state but in the liberty of playful expression. The play of making new rules, new algorithms, and new structures is indeed a technological speech act, but one that brings into existence new kinds of social relations, as well as human-machine relations. Consider the weeklong, nonstop online live coding events featuring hundreds of live streamed performances coordinated by Argentinian artist and developer Damián Silvani's muxy software,[23] or the features built into the Hydra live coding platform that allow live coders to submit and receive code patches as a kind of social networking through code. Such software extends live coding systems outward, in order to organize new social practices.

Infrastructural Monopolism and Craft Resistance

Behind these operations, and largely invisible, is the infrastructure of big data, in which data and services are delivered from a massive, intentionally obscured *cloud* of server farms across global fiber-optic networks.[24] Global corporations compete to exploit this infrastructure, each working to recruit customer users into its own proprietary ecosystem: consumer devices that harvest data from the user's home or pocket, archives of every action and utterance the customer makes, and software that sells these data to advertising agencies or back to the customer themselves as new services. The proprietary ecosystem architectures are designed to lock in the user as a loyal citizen of Apple, Google, Facebook, or Amazon, through which the corporation gains maximal rights to the resulting data. Programmers are increasingly reliant on code produced by these companies, particularly for the machine-learning libraries at the core of data-driven business. Statistical pattern analysis allows companies not only to monitor every action of their users but also to predict those actions in a dynamic now recognized as *surveillance capitalism*.[25] However, users constantly experience disconcerting tensions between conveniently intelligent prediction and creative action or decisions. The everyday dynamic of predictive text means that whether writing in love or in condolence, a user's devices are intervening to assist, brushing aside any poetry with the statistical recommendation of familiar words they have used before.

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

This substitution of archived platitudes for creative expression is the real business model of big data. When a user struggles to express an original query for Google, the browser leaps to offer the same question that ten thousand other people have already asked, ideally directing them to a product linked to specific Ad Words. Even when avoiding invitations to purchase, "free" answers involve, at best, delivering an article that the PageRank algorithm has judged to be most popular or the carefully anodyne compromise of a hundred Wikipedia editors. Individual authorship is neither needed nor wanted. Whether celebrated as open source or creative commons, the reality is that nobody gets paid for the work that was done, other than Google, which acts as the "rentier of cognitive capitalism," in the analysis of media philosopher Matteo Pasquinelli,[26] or even engages in "institutional plagiarism," as suggested by Alan Blackwell.[27]

In contrast to these algorithmically mechanized and statistically standardized online experiences, live performance confronts the audience with a real, embodied person on the stage. Live coding 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. Whereas algorithmic assistance might encourage performers to abdicate from the elementary creative decisions of each moment and newer generations of machine-learning systems engage in institutionalized plagiarism through the processing of anonymized training data for the automated pastiche of *style transfer*, live coders are able to define the terms on which remixing will proceed. In live coding, the structural components of the sound are openly acknowledged in the code that produces it and, as a result, are made more visible.

Perhaps the question, then, is whether the live coder can escape the infrastructural supply chain of the music industry, whether iTunes, YouTube, or Spotify.[28] Put simply, just as the industrial revolution delivered all surplus value to the bank accounts of the capitalist landowners, so, too, has the big-data revolution been extracting value from creative work to enrich those who own the infrastructure. Live coders could look to Luddism as one way out. The Luddites formed a prototypical union movement that, not unlike live coders, stood for craft practices and against full automation.[29] Despite their modern-day image, the Luddites were not against technology in general; rather, their radical, distributed, and at times successful campaigns were to protect the livelihoods of artisans who were accomplished users of their own technologies. Indeed it is plausible that software engineers will in the future find many elements of their own industry replaced by AI automation (as hinted by GitHub's AI copilot), just as handweavers found their livelihoods undermined by machine automation. Power looms stand not only for the automation of weaving but also for limits on the possibilities

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

of weaving. The algorithmic patterns intrinsic to weaving handcraft were dehumanized and creatively constrained within the grids of Jacquard's punch cards. Just as weavers sit at handlooms, live coders sit at computers, humans and machines working together in ways that suggest alternative, potentially more social and sustainable forms of control.[30]

The Live Coding *Other*

As has been established, live coding tends to be presented as a practice in which the coder is in relative control of the machine, a scenario in which the machine is a passive receiver of instructions and follower of commands, executing according to script. However, computational systems need not be so submissive. As we saw in chapter 4, live coders have from the beginning explored models of programming in which ideas of actors, interactive objects, conversational programming, or machine learning become more overtly constitutive in the relationship between the coder and the machine.[31]

The prospect of the live coding language learning about the coder is both enticing and disturbing to the narrative of taking back control. Reflecting continued advances such as code completion, templating, and automated refactoring in software-engineering tools, and even the generation of novel code,[32] the live coding language might suggest, predict, comment, or correct code that is being created in real time. These systems might learn from more than one user, gathering data from the crowd in a manner similar to how data analytics is increasingly applied in commercial music technology contexts to track user behavior, perhaps even identifying new music genres as they emerge. The live coding alternative is interesting in this context given that most, if not all, live coding systems are committed to open-source principles of collaboration—not least that the code is projected to the audience. In an open-source and open-access world, live code data analytics could allow participants to become more aware of and connected to wider cultural movements.

In principle, the definition of neural network architectures for machine learning could also be live coded. What might not be so open to the audience, or publicly projected, would be the statistical outcome of the training or the model structure in which pretrained networks are used. This is a problem studied in the MIMIC research project, which puts machine learning, particularly its Sema live coding language, into the hands of creative coders and aims to support *small-data* machine learning, real-time initiation, and training, all as part of the live performance.[33] Relatedly, the artist-driven Patterns in between Intelligences project opens up AI through dance and live coding. This performance approaches intelligence in terms of oscillatory processes in the brain, which it extends into oscillatory processes between live coders, live dancers,

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

interactive etextiles, and pattern-recognition and pattern-generation algorithms.[34] The explorative, critical, and questioning principles of live coding apply here as well: if the commercial companies are feeding artists and audiences precooked tools for musical creativity, live coding can still represent a locale where users are able to experiment, learn, develop, communicate, and share through a critical relationship with emerging technologies. It might well be that the public understanding and experience of machine learning and AI will be through live coding and creative coding—that is, areas where programming is used for human expression and where the concepts of sharing and communicating are still prioritized over commodification and datafication.

It is likely that many technological devices will increasingly adapt to their users in the future. A smart refrigerator (a perennial example in the imaginaries of future consumption[35]) should ideally learn the behavior of each fridge owner, ordering (or recommending) what each person needs rather than forcing predefined and standardized food consumption on all. The same is happening across the Internet of Things in models of cars, sporting appliances, smart watches, game controllers, and, potentially, musical instruments. With the latter we might imagine a situation in which two physical musical interface controllers come with the same default mapping (gestures-to-sound) from the factory, but each user might then train the instrument such that it adapts to their musical needs and habits. Two users who bought the same musical instrument in the shop might, after a few months, have trained two quite distinct instruments. The training would have derived from the use and supervision of the owner. The two users might even swap training data (since the data would simply be a file on the device), creating a sense of alienation and estrangement with the instrument.[36] Such experiences are familiar in other performance contexts; for example, violinists often remark that if they lend an instrument to someone else to play, it feels slightly different after being returned.[37]

Now an interesting question is whether the same type of programming instructions might be interpreted differently by differently trained program interpreters or programming editors. Might the system learn features of its user—for example, in terms of sounds (types of synthesis, such as FM or subtractive), effects (distortion, reverb), rhythmic preference (strict, swing), mix (volume of individual instruments, mastering effects, such as compression). In the future, could live coded notation be interpreted differently by instances of the live coding language, just like notated music is interpreted differently by every performer who plays it? Just as natural human languages change through use, systems will learn and cybernetically adapt to their user's behavior, and there is no reason to think that this won't happen with live coding systems, or that this is necessarily a problem. The cultural importance of the systems described here is that they will be open, trained, explored, and presented in real time such that

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

users of technology will still be able to yield control, understand, peek under the hood, develop, participate, and change, as people have always done with the technologies they make and use. Lisanne Bainbridge noted that one irony of automation is that an automated system requires an expert to supervise it and step in when it goes awry, but it is difficult to maintain this human expertise in an automated task they are disassociated from.[38] Live coding offers a way to keep humans engaged as authors rather than supervisors of automation.

The continuing concerns over developments in machine learning are also plain to see whenever predictive algorithms, or predictive analytics techniques, are applied in the broader context of the creative economy. Even critical practices that employ machine-learning techniques run the danger of perpetuating the same logic—whether intentionally or not—because they are necessarily aligned to the capitalist production underpinning their infrastructure. In other words, there are some serious worries about the forms of creativity produced through machine learning, given the broader political context in which it arises. Yet, despite these worries, practices can also emerge that engage with the claims of machine learning and the ways in which it produces knowledge and forms of creativity through reconfigured human-machine relations. What other technologies can live coders bring back from the world of mass production and control to the world of craft?[39] What alternative practices might emerge that take account of, but are not reducible to, the capitalist creative economy? We might refer once again to Agre's "critical technical practice" to stress the importance of social and political aspects of technical fields such as AI.[40] He asserts that AI becomes a discursive practice through the way the technical terminology offers intellectual generativity, developing analogies between otherwise disparate technical and critical activities and intellectual traditions. In addition, we would suggest the need to look beyond creative, technological, and critical traditions that are rooted in colonialist and extractivist worldviews. The ways in which we have conceptualized human-machine relations may have become less stable in the case of live coding and be grounds for reinvention, but they are still rooted in Western cosmology.[41] We hope the examples in chapter 3 demonstrate how contemporary live coding is responding to some of these challenges through diverse practices.

Live Coding in a Globalized Context

Previous chapters have discussed the imaginaries opened up by live coding, but it is important to consider what practical boundaries might remain, even if adopting the liberating ideals of movements such as free/libre open-source Software (F/L/OSS).

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

The F/L/OSS movement, in which "'free software' is a matter of liberty, not price" was becoming a mainstream feature of contemporary arts and data activism by the time the TOPLAP manifesto was written. As noted by Christopher Kelty in 2004, F/L/OSS had "broken free of its connection to software and become common among artists, writers, scientists, NGOs, and activists."[42] The once liberating ideals shared by this nexus of media artists, policy advocates, and open-source hackers established a zeitgeist for the arts in the early twenty-first century. And yet this critical tradition that takes as its focus the ability to have access to code (and sharing screens) is not particularly revealing or effective unless it is understood as part of the larger set of sociotechnical relations within which it operates. When it comes to machine learning, for instance, as Mackenzie explains, the code logic is different, as the learning does not rely on symbolic logic alone. The further point is that there is more than code here, situated as it is in open-source platforms and communities of interest that help to shape the ways in which power and knowledge are expressed.[43] This is both inclusive and exclusive of others.[44] The problem is that there was little acknowledgment at the time these movements were founded and thrived in the Global North, where the advocates and interventionists were White and male by a large majority. Just as the free software community excluded women to a far greater extent than the (already male-centric) computing industry,[45] it seemed that attention to software as a moral enterprise in itself could often draw attention away from the individual activist's own privilege and participation in systems of oppression.

The ideals of openness are easily co-opted for purposes of oppression. As explained by Linda Tuhiwai Smith in relation to Indigenous peoples:

> Multinational companies have been given transnational freedoms that enable them ultimately to move labor across borders (importing and exporting people for the labor market), to foster an intellectual property regime that has few ethical limits, to shape national laws and values at the expense of national identities, and to develop themselves in competition with governments.[46]

Similarly, the supposedly open, but still Western, infrastructure of the scientific knowledge economy means that the scholars of Africa become subject to scientific extraversion and are oriented away from their own countries.[47] Movements that are associated, in the Global North, with inclusion and consultation are similarly co-opted for other purposes in the South. The agile software movement (as mentioned in chapter 7), with its origins among liberal thinkers of Portland, Oregon, has subsequently become another source of asymmetric relations in global outsourcing.[48]

If we establish some critical distance from the moral enterprises that have defined ethically oriented movements such as open science, free software, or indeed live

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

coding, might there be any obligation to distance live coding itself from global infrastructure, as a necessary element of its liberating agenda? We should first be clear that, while observing how live coding is indeed implicated in global systems of inequity constructed through the legacy of colonialism, addressing the problems of live coding is not an adequate redress. We do not wish "to focus on decolonizing the mind, or the cultivation of critical consciousness, as if it were the sole activity of decolonization; to allow conscientization to stand in for the more uncomfortable task of relinquishing stolen land."[49] Similarly, we would not wish to claim a "postcolonial" viewpoint from which to avoid complicity in these systems. As Aborigine activist Bobbi Sykes observed: "What? Post-colonialism? Have they left?"[50]

Nevertheless, commentators on the software industry, drawing on theoretical discussions of decolonization and postcolonialism, have identified useful warnings and precautions that are relevant to live coding. As proposed by Lilly Irani and others, "Postcolonial computing, then, is not a project of making better design for 'other' cultures or places. It is a project of understanding how all design research and practice is culturally located and power laden, even if considered fairly general."[51] Similarly, Geoffrey C. Bowker and Susan Leigh Star's argument in *Sorting Things Out*, on classification and how we structure our world through language and design, clearly implies that systems designed in the Global North might not fit so well to work patterns elsewhere in the world.[52] In response, what is suggested is "a bag of tools" for critical reappraisal of computing technologies.[53] Drawing on these ideas, we are reminded once more of the *long networks* in which live coding participates and on which it depends. These extend beyond open-source software components and communities of support to networks, servers, manufacturing facilities, and mineral extraction. As compellingly diagrammed by Kate Crawford and Vladen Joler in their *Anatomy of an AI System*,[54] and subsequently evidenced at length in Crawford's *Atlas of AI*,[55] the abstractions of contemporary computational culture remain founded in material realities of colonial oppression and exploitation.

These unwelcome realities of our contemporary situation seem entrenched in ways that live coding would struggle to challenge or modify. Nevertheless, as indicated above, there are tactics that can be applied in sites of innovation—and live coding is fundamentally enacted as a practice of innovation and creation. Many live coders, for example, attend to the *coalescing regime* of the community, asking what work has been placed outside its bounds. For example, why has live coding not been widely welcomed by hip-hop, grime, or drill producers as a tool for vocal performance and social critique? When listening to the impressive transcriptions of Fela Kuti compositions that artist and programmer Evans Augustt created in Sonic Pi,[56] one might ask

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

how remediation via code could develop the political activism central to the origins of Afrobeat. As with the Afrofuturism of Sun Ra and Black Panther, or indeed the sanctification of Marcus Garvey in reggae music, the harder implications of pan-Africanism in relation to systematic racism and injustice warn their audiences against the trivial appropriation of exotic samples or dance rhythms. Many of these commercial musical practices have been established within, and constrained by, the capabilities of digital audio technology. Live coding could certainly be a tool for the disruption of such boundaries, for which we look to follow examples being set by the growing live coding communities in Latin America, from the Colectivo de Live Coders in Argentina to the birth of Algorumba in Medellín, and the grounding in living heritage, such as through the Neokhipukamayoq manifesto (discussed in chapter 2).

It may be necessary to be more alert to the implications of live coding's own generous intentions, more clearly articulating a developmental "theory of change" in association with the global ambitions for live coding.[57] Maybe this is something live coding wants—to simply be useful.

Conditions of Use

As stated in chapter 1, the title of this book borrows from Georges Perec's novel *Life: A User's Manual*.[58] The original French title, *La Vie mode d'emploi*, makes clear a relation between the terms *use* or *user* and the notion of employment, with *use* meaning "to employ for a purpose," "to make use of," "take advantage of," or "to consume." Indeed, a user's manual (a *mode d'emploi*) is often something to be used—to be put into action or to be applied in practice. It could be easy to conceive of use only as the capacity of a commodified object to serve a useful purpose or satisfy some extrinsic and existing demand, need, or want. Rather than focusing on the usefulness of live coding (and how it might serve an external purpose or want), how can we look beyond the utilitarian to really consider what live coding itself might want? Indeed, the etymology of the term *employ*, "to entangle," from the Latin *implicare*, meaning "to enfold," "to involve," "to be connected with, associate or else unite," also complicates any narrowly utilitarian or even instrumentalized application. In these terms, this book is not so much a *mode d'emploi* or user's manual *on* or *about* live coding as an attempt to explore live coding *as* a user's manual or guide.

We return to the title of the book, *Live Coding: A User's Manual*, to underscore that live coding is about people interacting with *life*—with the world and with others—with social relations. Or rather, by returning to the title, we again foreground the significance of the term *user*—and of the user's interaction with both computer technology

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

and an increasingly technologized world. We refer once again to artist Olia Lialina's lamentation on the disappearance of the user and the critical implications therein. As Lialina observes, the evolution of digital technologies has in part involved an evolution of alienation, the "alienation of users from their computers."[59] She states that "the denial of the word 'user' in favor of 'people' becomes dangerous. Being a User is the last reminder that there is, whether visible or not, a computer, a programmed system you use."[60] Counter to the notion of the "invisible user,"[61] live coding can be seen as part of a wider movement of resistance, which epitomizes in Lialina's terms a certain mindset or mode of interaction with hardware and software that "makes the user visible, most importantly to themselves."[62] Live coding's practice and politics of making the screen visible are also gestures of making visible the possibility and potential of a user, or a multitude of users who might otherwise be rendered invisible. It is a practice for making visible the symbiotic relationship between human and machine, as well as the broader infrastructures that support it. The potential of a *critical-technical user* is that they are not simply implicated or complicit within an increasingly technologized "society of control,"[63] which they can neither fully understand nor begin to resist but retain or reclaim some capacity to act within—or even to complicate—that system.

Indeed, live coding can be conceived of as a practice of *complication*. Throughout this book, live coding is shown as a practice that complicates and exceeds easy categorization or containment within any singular system—whether of notation, of liveness, of temporality, of epistemology, or of coding itself. Live coding willfully undoes—disrupting and unraveling established categories and ways of thinking while weaving a more complex relation and collaboration between the sciences and the arts and between human and nonhuman forces. From *implication* and *complicity* to *complication*—etymologically, these terms each contain the Latin *plicare*, meaning "to fold, weave." *Complicate* then means to fold or weave together. In these terms, live coding's practice of complication is not one of making things difficult nor only of destabilizing certain (binary) definitions and demarcations of making strange. Affirmatively, it has the capacity to reveal or make visible something of the complexity of posthuman entanglement and intrarelating.[64]

At times, writing this book has also been a complicated undertaking. It has evolved through a process of gradual development—its work has taken time. For all of its just-in-time immediacy and liveness, our attempts to engage with the complexity of live coding as a practice have required a slow approach. As Michelle Boulous Walker reminds in *Slow Philosophy*, "Honouring the rhythms and temporality of deep and careful thought—thought that is threatened by speed, efficiency and interchangeability—means, perhaps, honouring the importance of a slow engagement with the work that

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

we do."[65] Our own slowness attests to the challenges that we have encountered during the process of our collaborative writing as we have attempted to *write with* live coding rather than necessarily write *about it*; as 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 we have tried to reflect on both the practical and conceptual implications of live coding as a critical technical practice.

In the early stages of the process, our approach had distinctly experimental ambition, willfully playing with various rules and constraints and striving for a form in which the writing itself remained close to the practice of live coding and in which the different authorial styles were more amplified and clearly evident. 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*. Likewise, though we conceived the structure of the book as broadly bipartite—the first part more practice oriented, the second part more speculative and conceptual in its register—we hope that the reader will navigate between, complicating any neat separation of practice and theory.

During the period of writing this book, we watched live coding continue to evolve as a practice and as a community in ways that could not have been predicted at its inception. It has now been over twenty years since the specific term *live coding* first appeared (around the year 2000), and considerable time has passed since the TOPLAP manifesto was first drafted in 2004. Indeed, in so many ways the state of the world feels very different to those millennial years. As outlined already, live coding emerged in relation to the coming together and culmination of very specific conditions: the influence of critical precedents and forerunners; significant technical developments enabling greater access to and availability of computers, as well as increased capacity for real-time programming; higher educational institutions receptive to experimentation and the genuine risk of transdisciplinary collaboration; the existence of cultural venues open to new possibilities of practice; the techno-optimism of people who embraced the global digital network as a liberating force, often driven by companies that promised not to be "evil";[66] international festivals and symposia that enabled the cross-pollination of ideas and practice to create a live context for conversations and collaborations, for experiencing live coding *together*. Writing this chapter in 2021 against the wider context of ever-increasing global precarity—economic, environmental, political, and social—and indeed from the perspective of numerous lockdowns and widespread losses experienced during the current coronavirus pandemic, the fragility and interconnectedness of various ecologies of life and practice become ever-clearer. The task of confronting racial inequalities and dismantling the colonial edifice of White privilege gained a heightened sense of urgency in

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

the wake of the police murder of George Floyd in May 2020 and the subsequent Black Lives Matter protests. In parallel, the implications of the current climate emergency for all forms of life are a live crisis that no field of practice can afford to ignore.

We don't yet know how live coding will respond to these new challenges. It could be tempting to see conditions of practice as something given, something outside of one's influence or control. Although life will always be unpredictable, certain conditions of practice can be created as much as awaited and actively produced rather than lamented, resisted, or simply reacted to. Live coding exists in a technical ecosystem, and when new languages, techniques, and technologies emerge—be it haptic interfaces, a new language, or libraries for AI, for example—live coders are likely to pull them into their practice, critically evaluating and contextualizing them in the performative setting. It is perhaps easy to forget that live coding did not just emerge in response to certain conditions but also created the conditions for its own emergence. Many first-generation live coders were actively inventing the various programming languages and live coding environments that are now used by live coders all over the world. Many live coders create and curate, as well as perform at, the various festivals, symposia, workshops, seminars, and algoraves through which the practice of live coding is shared and evolved and are core members of academic and research programs. How might live coding retain the liveliness of its instituting while resisting institutionalization? This question has also been very present in the writing of this book. Certainly, there are new communities of live coding coming into being all over the world that retain a sense of the urgency, even insurgency, of those initial instituting moments of live coding and that continue to invent and reinvent the conditions of and for practice. Still, the evolution of the practice and community of live coding has involved a broad shift from the conditions of necessity and urgency (when the means did not yet exist so had to be invented) to one of *plenty*, to a plethora of technological options and choices for the prospective live coder.[67] However, technology (and indeed live coding) is something that you *do*, not something that you simply consume or own.

Tracing the trajectories of live coding's evolution is not an exercise of nostalgia nor a gesture of retrospectively looking back. Rather, this book attempts to reinvigorate the critical questions that drove live coding practice in its inception while anticipating the possibilities of live coding's future still yet to be written and still remaining in potential. Live coding is not a progressive practice in conventional terms—it is not concerned with the normative teleology of development, improvement, and growth. Live coding wants to be undone, unraveled, unwoven, and rewoven anew.

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a portion of the eBook at doi:10.7551/mitpress/13770.001.0001

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

This is a section of doi:10.7551/mitpress/13770.001.0001

Live Coding

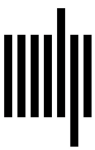
A User's Manual

By: Alan F. Blackwell, Emma Cocker, Geoff Cox, Alex McLean, Thor Magnusson

Citation:

Live Coding: A User's Manual
By: Alan F. Blackwell, Emma Cocker, Geoff Cox, Alex McLean, Thor Magnusson
DOI: 10.7551/mitpress/13770.001.0001
ISBN (electronic): 9780262372633
Publisher: The MIT Press
Published: 2022

The open access edition of this book was made possible by generous funding and support from the author



The MIT Press

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024

MIT Press Direct

© 2022 Massachusetts Institute of Technology

This work is subject to a Creative Commons CC-BY-SA license.
Subject to such license, all rights are reserved.

The MIT Press would like to thank the anonymous peer reviewers who provided comments on drafts of this book. The generous work of academic experts is essential for establishing the authority and quality of our publications. We acknowledge with gratitude the contributions of these otherwise uncredited readers.

This book was set in Stone Serif and Stone Sans by Westchester Publishing Services.

Library of Congress Cataloging-in-Publication Data

Names: Blackwell, Alan F., author. | Cocker, Emma, author. | Cox, Geoff, author. | McLean, Alex, 1975– author. | Magnusson, Thor, author.
Title: Live coding : a user's manual / Alan F. Blackwell, Emma Cocker, Geoff Cox, Alex McLean, and Thor Magnusson.
Description: Cambridge, Massachusetts : The MIT Press, [2022] | Series: Software studies | Includes bibliographical references and index.
Identifiers: LCCN 2022008717 (print) | LCCN 2022008718 (ebook) | ISBN 9780262544818 (paperback) | ISBN 9780262372626 (epub) | ISBN 9780262372633 (pdf)
Subjects: LCSH: Computer programming—Philosophy. | Agile software development. | Creation (Literary, artistic, etc.) | Algorithms—Psychological aspects.
Classification: LCC QA76.6 .B5794 2022 (print) | LCC QA76.6 (ebook) | DDC 005.1301—dc23/eng/20220527
LC record available at https://lccn.loc.gov/2022008717
LC ebook record available at https://lccn.loc.gov/2022008718

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2239297/c011100_9780262372633.pdf by guest on 19 September 2024