5 Live Coding's Liveness(es)

The liveness of live coding is exemplified by the *just-in-time* or *on-the-fly* nature of its improvisatory extemporizing—its performing and showing of live *thinking-in-action*[1] differentiating it from certain kinds of notation-dependent forms of (musical) performance and generative approaches to audiovisual performance that are both scripted or coded in advance of being *played*. While chapter 4 addresses different notation and writing systems more explicitly, this chapter explores how such abstracted understandings might become reembodied and enlivened. However, the intent is not to valorize or fetishize the liveness of live coding as evidence of its supposed authenticity or originality but to address how diverse live coding practices[2]—improvisational and compositional—enrich a wider theoretical debate on the issue of liveness.

Expanding upon Philip Auslander's examination of live performance within mediatized culture,[3] the human-machine entanglement at the heart of live coding offers a highly complex, hybridized, and yet still undertheorized model of liveness.[4] What are the implications of live coding for our understanding of the concept of liveness? What kinds of liveness are being produced, and with what effects? What does this indicate about the relations between technology, performance, and even "life" that live coding suggests? How might live coding relate to, as well as contrast with, other live art forms and even potentially oppose their appraisal of immediacy and openness?

While the liveness of live coding remains a distinguishing feature, it has been shaped by different genealogies of ideas and practices, both philosophical and technological. The interdisciplinary nature of live coding—emerging between the lines of various scientific and artistic disciplines—requires that its very liveness be understood from more than one epistemological and ontological perspective. That live coding's liveness is predicated on mediatization and mediation through technology complicates any neat distinction between live and digital entities or phenomena. Indeed, within live coding, liveness refers both to nonhuman "machine liveness" (or "degrees

of liveness," articulated by computer scientist Steven Tanimoto as the increased immediacy of semantic feedback and real-time character of computational processes enabled through technological advancement[5]) and to the contingencies and vitalities of embodied human experience.

The previous chapters establish much of the historical background of live coding and its intersection with the development of particular programming languages that enable interaction with a running system that does not stop while waiting for new program statements. This allows the code to run immediately as it is executed, with the performer able to adjust and respond accordingly. In 2004 the Changing Grammars symposium indicated a shift in the use of programming language from a tool for "generating" sound to code conceived as a "conversational practice" in which the "grammar of sounds" could be improvised from within the code as part of the live performance itself. For some, live coding operates in resistance, critique, or as an alternative to the prevalence of generative processes within computer-based performance where algorithms that have been coded in advance are activated live and then left to run their course. Rather than giving over responsibility to the unfolding of an algorithm's logic, within live coding the performer consciously adopts a medial position, actively maintaining the conditions that will keep the action dynamic. Certainly, our aim in emphasizing the liveness of live coding is not to undervalue or underestimate the pregramming and practicing that happens "behind the scenes" and in advance of a live coding performance but rather to draw attention to a specificity of liveness operative and activated in and through the real-time actuality of live coding performance itself.[6]

A live coder might write (as code) a sequence of notes or other algorithmic structures, with the *live* performance involving the modification of intensity, volume, and speed. Admittedly, here there is a danger in representing live coding as sequencing one note after the other like in a score, for a live coder need not concern themselves with individual notes at all, instead having those generated from a higher-order process. For other live coders, the promise of increased liveness brings the possibility for greater improvisation and collaboration, and the code content itself is programmed as *real-time composition* as a live event.[7] Here, live coding advocates the agency of the computer alongside that of the performer, with technical constraints and machinic resistance forming a key part of live coding's collaborative and performative texture. The human-computer interaction within live coding foregrounds a more complex, nuanced, or even entangled relation. In this case, technology is not so much put to use as a mindless production tool kit but rather is worked with, the process unfolding through attending to or even collaborating with the resistances and affordances exerted by the technology.

For some performers, live coding involves the bricolage of preexisting or prerecorded samples and prescripted sequences brought together as a live event. These samples and sequences are effectively precomposed anterior to the situation and then modified and reconfigured through live performance, with the capacity even to be recorded and reworked and perhaps even named or titled.[8] For others, especially some of live coding's early pioneers, the practice of *live writing* coding is underpinned by a heuristic principle that works without a predetermined plan, including the possibility of starting from scratch. For example, live coding ensemble Benoît and the Mandelbrots describe their approach to *zero prepared code* as "hour-long musical conversations, starting by zero" while at the same time recognizing, "We have already a lot of prepared functions given to us by the programme, and we are reorganizing them live for our purpose."[9] For live coder Rangga Aji, live coding from scratch is a way that "I can learn how to directly or slowly decide what kind of musical structure that I want to build without depending on any prepared code."[10] ALGOBABEZ (Shelly Knotts and Joanne Armitage) state that they "rejected the use of prewritten code and structures" in favor of "acting in the moment, responding to context . . . developing a structure as we work, continually creating and resolving tensions."[11] Indeed, for some live coders, nothing is saved, recorded, or archived in support of future replaying: the performance both begins and ends with the blank screen/slate.[12]

These diverging approaches and attitudes to the liveness of live coding have given rise to different expectations and aspirations for both live coding's live performance and developments within its technology. On the one hand, within the live coding community there is a call for improved media technologies that enable greater immediacy of feedback, a shift toward predictive coding (explored later in the chapter) modeled on previous patterns and habits that support a faster, more fluid—perhaps even virtuoso—species of programming *comprovisation* (composed improvisation or improvisation with a composed structure).[13] On the other hand, there remains interest in a mode of improvisational performativity that harnesses the unpredictable, the unexpected, or the as yet unknown, where live coding is conceived as a vital site for experimental *per*-forming (the prefix *per-* meaning through, so *through-forming*), with new content arising in and through the very forming of the live performance itself. Rather than regard these tendencies in antagonistic relation, this chapter explores how different notions of liveness—indeed different degrees of liveness, alongside *a*liveness and undeadness—are negotiated within live coding and, moreover, how the development of intelligent machines might better facilitate live coding's liveness without eradicating the critical intervals and in-between spaces necessary for improvisatory invention and intervention. This chapter highlights the different livenesses that live coding seeks

to harness, where the immediacy of near real-time programming languages enabled through technical advancement meets the embodied heuristic of coding live.

The Ontology of Liveness

Within media culture and performance studies, the concept of liveness has been much debated and contested, inflected at different historical junctures in relation to wider questions of presence and absence; mediation, recording, and documentation; eventness and now-ness; ephemerality, temporality, and contemporaneity. While issues of temporality are addressed more fully in chapter 6, some theoretical coordinates around liveness are offered here as a way of identifying a conceptual terrain into which live coding intervenes and adds new understanding. Though it has been some decades now since their original publication, performance theorist Peggy Phelan's *Unmarked: The Politics of Performance* (1993) and media theorist Philip Auslander's *Liveness: Performance in Mediatized Culture* (1999) remain oft-cited references in the continued debate on what constitutes the ontology—indeed the ideology—of liveness. Phelan's account of the ontology of live performance argues that

> performance's only life is in the present. Performance cannot be saved, recorded, documented, or otherwise participate in the circulation of representations of representations: once it does so, it becomes something other than performance. To the degree that performance attempts to enter the economy of reproduction it betrays and lessens the promise of its own ontology.[14]

Auslander formulates a counterargument to Phelan's construction of performance as "representation without reproduction" and the assumption therein that its unmediated liveness is somehow more *real* than mediatized events. He demonstrates how the concept of liveness itself is a product of *mediatization* and that since the early twentieth century *live performance* has been mutually entangled with and has coexisted alongside recordings, nonlive media, and various forms of technological reproduction.[15] Auslander asserts that the very concept of liveness emerges in and through the relation with its perceived opposite, mediatization, rather than having any preceding ontological condition or existence.

For Auslander, life itself is in performance but is also in technology. He explains that while live and mediatized events may be distinct in terms of their position in the cultural economy, this is not as a consequence of their intrinsic characteristics but is more a result of the sociopolitical conditions within which they operate.[16] Auslander refers to Jean Baudrillard's notion of *mediatization* to indicate how media are instruments of wider sociopolitical processes administered by a "single code."[17] An interesting aspect

of the politics of live performance has been its apparent ability to resist commodity forces and the hegemony of dominant culture by its immateriality and nonobject status (e.g., through happenings and other conceptual art traditions). But this position is unsubstantiated according to Auslander, as no clear-cut ontological distinctions can be made between live and mediatized events. The concern remains with the changing character of performance in a situation in which human performances are entangled with media performances, and we must reject the simplistic notion of live events as unmediated or *real* as if outside representation.[18]

Auslander's seminal text provides an important backdrop to an understanding of the complex relations between live and recorded media but is a work very much of its time. Although he outlines how mediatized performance has been traditionally granted authority through its reference to the live or real event, liveness has now been even more fully incorporated into its mediatized forms, and performance is less and less outside of its technological conditions. Live coding extends the ontological condition of liveness in ways that media studies and performance studies have not yet conceptualized. Indeed, as Auslander acknowledges, "Liveness is not a stable ontological condition but a historically contingent concept, a moving target that is continuously redefined in relation to the possibilities afforded by emergent technologies of reproduction."[19] So how do the different registers of liveness operating in live coding offer insights into the live(d) experience of our digitized, mediatized, streamed, and increasingly "mixed-reality" world? That is, as sociologist Nick Couldry suggests, how are they shaped by the ever-new modes of technologically mediated liveness—*online liveness*, *group liveness*—that challenge notions of spatiotemporal (co)presence and proximity in relation to what constitutes the live?[20]

The Experience of Liveness

Performance scholars Matthew Reason and Anja Mølle Lindelof argue for a shift in focus from liveness to multiple and shifting "liveness-es" and to the pluralities of "experiencing live," acknowledging the different meaning and value that liveness holds across different disciplines, including music, performance, and media studies.[21] For example, they note how "bodily co-presence between audience and performances is central to performance studies. . . . Music studies have typically engaged with liveness in relation to recording techniques and sound ideals. . . . Within media studies liveness has been discussed in relation to transmission technology."[22] Live coding is a practice that operates at the interstices of these disciplinary domains, in turn activating their different attitudes to liveness. Reason and Lindelof call for a reexamination of the concept of

liveness in contemporary performance by moving beyond the contested questions of ontological difference and the relationship between the live and the mediated to focus on liveness in terms of processes of experiencing, processes of making, and processes of audiencing.[23] Accordingly, this chapter shifts from the sociotechnical, ontological perspective of Auslander to explore how liveness operates and is experienced within live coding, attending to both the embodied nature of coding from the mutually interwoven perspective of human performer and audience as well as to the "degrees of liveness" functioning within the nonhuman processes of the machine.

More than a purely conceptual procedure, live coding involves a sense of embodied awareness in which principles of knowing *how* and knowing *when* are as privileged as knowing *what*. Rather than reducing the role of the human operator—an accusation levied at some forms of computer-generated performance—live coding requires heightened levels of dexterity, attention, cognitive agility, and tactical intelligence. It is a practice of timing and timeliness, of biding one's time and knowing when to act—that is, a practice of *tact*. Indeed, live coding does not just involve the logical manipulation of code language but rather unfolds as much through the complex embodied relation of rhythm, repetition, and response. While code is abstract, mathematical, and seemingly disembodied, live coding—as a performance—draws us back to the body of the performer. The computer itself tends to deny the body, reducing touch to a single pointing finger and sense to a symbolic eye.[24] Computer scientists, anxious in affairs of the flesh, strive to avoid it—witnessed in the advocacy of NUI, a natural user interface that aims to eradicate the last vestige of touch.[25] Yet, for all that computing neglects—those bodies hunched over keyboards and mice, at their desk-and-chair sets in the offices of the world—the live coding performer is unavoidably embodied ("made flesh").

Certainly, there is an inherent kinetic, even *kinaesthetic*,[26] dimension to the live writing of code involving direct physical engagement with the machine: a sensorimotor movement vocabulary of microadjustments, changes, and shifts performed in the frantic keystrokes, in the shuttling of the cursor around the screen, in the flash points of activation and execution. Analogous to the pulsing live body within other forms of performance, the cursor marks the point of decision-making within the live programming of code, the movement of the coder's thinking as it oscillates between sensemaking through the discontinuous, abstract notational form of code and the continuous—even sensuous—experience of coding as a lived experience. Live coding deviates a linguistic-numerical computational logic—algorithmic thinking—toward artistic application, involving not only a musical-rhythmic intelligence or sensibility but also the more intuitive knowledge(s) commonly associated with creative, embodied thinking.[27] This *know-how* is arguably of a more tacit kind, involving the tactility of active exploration

or haptic perception that psychologist James Gibson describes as "the sensibility of the individual to the world adjacent to his body by use of his body."[28] While chapter 7 further elaborates the epistemological implications of live coding's *embodied knowledge*, this chapter considers its embodiment through the prism of liveness.

Over the past decade, a number of research projects and symposia have attended to the live interactions between body and machine, to the navigation of various human and nonhuman forces and flows, and to the specificity of the liveness of embodied thinking-in-action within live coding. In 2014 the University of Sussex's Emute Lab in the UK hosted a symposium titled Live Coding and the Body to address this burgeoning field of practice and to expand ideas about what live coding is. The symposium raised questions concerning the interaction between live coding and the body, such as: How can live programming contribute to a better understanding of the body, language, and notation in live performances that use digital technologies? To what extent is the coding integrated into the practice of performing arts?[29] Consequently, as the field expands so, too, does the frame of reference and conceptual ground, with the issue of liveness rethought through the prism of somatic practices and affect and embodiment theory drawn from choreographic studies. This perhaps reflects a wider *choreographic turn* within contemporary culture where concepts of embodiment, performativity, corporeality, and choreopolitics, alongside philosophies of movement originating in dance, have increasingly entered the expanded interdisciplinary field.[30]

Indeed, for over a decade, live coding has extended beyond the performance of computer-generated music and projected visual effects, expanding rapidly to embrace practices—especially from live art, performance, and choreography—that foreground the potential of code in explicit association with the live performing of the body. For example, live coder Kate Sicchio uses the umbrella term *hacking choreography* for describing live coding performance scores for dancers performing choreography "where a choreographic score would be changed live in a performance setting."[31] Since 2011 Sicchio has been involved in an ongoing performance research collaboration with Camille Baker called Hacking the Body and HTB 2.0, involving *hacking* the data from the body to create new forms of choreography.[32] Since 2016, transdisciplinary coder Joana Chicau has been

> investigating diverse notation systems from both dance and web programming, which I then merge into a hybrid form of algorithmic composition: a "choreo-graphic-code." . . . The liveness of code writing became a way to activate the choreo-graphic-code, exposing various processes and dynamics of web computing while enhancing the physicality of the body—the body that is in constant friction between the constative (reality describing) and performative (reality producing). Engaging with different forms of choreographic thinking has been a way to bridge and enhance the somatic and semantic within coding.[33]

The research project Live Notation: Transforming Matters of Performance (introduced in chapter 4) focused on liveness as a connecting principle for exploring the relation between live coding (performing with programming languages) and live art (performing with actions). This project privileged the durational, embodied, nonrepeatable moment of performance, drawing attention to an improvisatory or even *kairotic* species of liveness rather than the live reactivation of a preexisting script or score that is rehearsed or planned in advance (the concept of *kairotic coding* is explored further in chapter 6). Focusing on the vitality and embodied materiality—the liveliness or even aliveness—within live coding opens up the exploration of liveness to these wider theoretical orientations.

The Vitality of Liveness

Psychologist Daniel Stern uses the term *vitality* to describe qualitatively the "manifestation of life, of being alive."[34] He examines how vitality, and vitality affect, manifest within everyday life, psychology, and the performing arts (as distinct from the domains of emotion, sensation, and cognition). For Stern, "Movement, and its proprioception, is the primary manifestation of being animate and provides the primary sense of aliveness."[35] Additionally, Stern states, "Vitality dynamics refer mainly to the shifts in forces felt to be acting during an event in motion, and thus focus more on the dynamic qualities of the experience, in particular the profile of the fluctuations in excitement, interest, and aliveness."[36] He argues that dynamic forms of vitality "concern the 'How,' the manner, and the style, not the 'What' or the 'Why.'"[37] Here, the knowing *how* to which Stern refers is of a qualitative kind rather than a type concerned with the imperative technics and techniques of production as such. His focus on *how* addresses the modulating changes in affect and attention as they occur within the liveness of action. The vitality dynamics of *how-ness* that Stern describes have a specific "temporal contour," or profile, a sense of processual directionality, intentionality, and movement vector.[38]

While Stern focuses on performing and time-based arts such as dance and music—practices that take place in real time—it is worth noting that the forms of vitality he describes are not considered to be exclusively the domain of improvisatory practices. Indeed, live performance offers no guarantee of vibrancy or vitality—its experience might be on occasion more akin to "deadliness,"[39] and composed or scored composition has the capacity to be *more* vital (alive) than improvisation (whose "live" is not always experientially enlivened). The question of how to reinvest composition with vitality or a sense of liveness therefore remains a perennial challenge across diverse performing arts. Here, as theorist Erin Manning argues:

> The "how" of the work as it is replayed across settings and environments is its commanding form. This "how" is emergent each time anew and is always a complex mixture of technique and technicity. Technique to keep the piece rigorous, to give it the subtlety and nuance and precision it requires. Technicity to make the work outdo itself, to make the work.[40]

Significantly, for Manning, this foregrounding of the *how* involves a radical "letting go" of the original in order to refind it again, where "to begin is to begin again, differently, impossibly, impractically."[41] This is not simply the repetition of what is already rehearsed and ready in waiting. As Manning elaborates, "No movement can be cued, aligned to, or performed in exactly the same way twice. . . . What emerges as a dance of attention cannot be replicated."[42]

Increasingly interdisciplinary in its scope, live coding provides a frame for sharing and debating the overlapping and divergent conceptualizations of issues such as real time, liveness, and embodiment within diverse performance practices. However, this turn toward embodiment is not a turn away from the machine but rather a way to address the anthropocentric privileging of the human body and its capacities alone. Indeed, for literary critic N. Katherine Hayles, the posthuman is an embodied mode of being where information and materiality are not conceived as separate entities.[43] Expanding the interdisciplinary frame beyond the performing arts to include textile and weaving practices, the Weaving Codes/Coding Weaves project (discussed in chapter 4) provided a context for exploring the specificity of thinking-in-action while improvising within a live running code and how it might relate to the embodied *thought-in-motion* while working on the loom.[44] Central to the project was an attempt to dislodge the dominant model of working out, wherein an idea is conceived in advance and applied to material, in favor of a process of feeling one's way, in which various levels of operation and cognition are activated live. The concerns of the project resonate with Tim Ingold's writing on "being alive," alongside his conceptualization of the "textility of making," which integrates aspects of embodiment, material properties, and agency to address the movement and processes of negotiation between material and human action and between materials and forces.[45]

The immanent and embodied experience of liveness within live coding might be conceived in terms of *flow* or *flow state*. Flow describes a hyperfocused state of optimal experience—or mental state—conceptualized by psychologist Mihály Csíkszentmihályi as "full absorption," immersion, or "total involvement" in the process of an activity. Csíkszentmihályi argues that flow states involve an experiential transformation of time, alongside the merging of action and awareness: actions become spontaneous, even automatic—intrinsically rather than extrinsically meaningful. Live coding—like many other improvisatory practices—involves states of heightened concentration and

temporal disorientation in which an intrinsic value is placed on the process in and of itself. For Csíkszentmihályi, flow is also "based on a concrete experience of close interaction with some Other, an interaction that produces a rare sense of unity with these usually foreign entities."[46] It is perhaps in this sense that the live coder collaborates with their technology through a process of interaction in a system of action that is greater than the intentions of the individual self, that involves letting go of some control while *also* maintaining a heightened sense of mental activity and the activation of one's skill commensurate to the challenge.[47]

The optimal experience of flow depends on achieving the sweet spot of desirable difficulty between the nature of the challenge and one's available skill, between one's intention and capacity. For Csíkszentmihályi, to achieve a state of flow there should be a focused sense of intention that still allows one to gauge "Yes, this works; no, this doesn't."[48] However, in order to activate these microdecisions from "in the zone" of flow it is necessary to receive immediate feedback. Herein lies the challenge for the improvisatory live coder, for while live coding is described as a mode of real-time composition, in reality there is a lag (or delay) between the writing of code and its execution, which further complicates the issue of the *liveness* within live coding.[49] The live coder might be working on code that has not yet been executed (a preparatory step) but that is nonetheless visible as part of the "live performance" through the projection of the screen. In the case of the live coding of music, the live actions of the musician-coder might not necessarily correspond with the sound heard by either coder or audience in the moment. What the coder is witnessed doing live is not always coterminous with the sound that is experienced live. There is a disjunction (for both audience and coder) between what is seen onscreen as being worked on live and what is heard in the present moment of the live performance itself.

Unlike other forms of live composition, the relationship between a performer's actions and the resulting effects can be *asynchronous*. Or, rather, there are parallel threads of *arrhythmic* liveness: the live (yet arguably no longer live) unfolding of the effects generated through the event of coding and the live event of writing code itself, both of which are simultaneously experienced by the coder-performer and audience. The live coding performance evolves through continual feedback—the coder expresses the code that expresses sound back to the coder, which allows them to understand the code that they have expressed and to modify it. The live coder writes code in preparation for the future, for the next change, which is then executed when the time is right through the "gesture-sound event" of activating the code. Indeed, it is the potential of *minimized latency* in the execution of code that has enabled live coding, with improved media technologies allowing for greater immediacy or timeliness of execution feedback,

effectively making the performance more real time, increasing the perception of its liveness.[50] Accordingly, we now look briefly at the evolution of minimized latency and its implications for the liveness(es) of live coding.

Degrees of Liveness

Steven Tanimoto articulates the technical evolution of minimized latency in terms of "degrees of liveness." Tanimoto developed a hierarchical system initially for describing the four different degrees of liveness within programming: Level 1 (informative), in which no semantic feedback about a program is provided. He argues that this first level involves the four separate phases of edit, compile, link, and run. In liveness level 2 (informative and significant), semantic feedback is available on demand for a selected component: "The programmer would do something, would ask for a response, and some time later, the computer would respond."[51] In level 3 (informative, significant, and responsive), incremental semantic feedback is automatically provided with an incremental program edit: "The computer would wait and sometime after the programmer did something, would respond."[52] In level 4 (informative, significant, responsive, and live), incremental semantic feedback is automatically provided for other data events such as mouse clicks or exceptions: "The computer wouldn't wait but would keep running the program, modifying the behavior as specified by the programmer as soon as changes were made."[53]

Tanimoto has since elaborated two further levels of liveness, which in addition to swifter feedback response involve tactical and strategic prediction. Made feasible through the use of machine-learning technology, statistical analysis of programmer behavior, and "logical reasoning about meaningful choices," Tanimoto argues that in the new liveness level 5 (tactically predictive), "The computer not only runs the program and responds, but also predicts the next programmer action. . . . Instead of the environment lagging behind, or just keeping up with the programmer, it stays a step ahead of the programmer."[54] Level 6 would involve further "intelligent inference of the programmer's intentions or desires."[55] For Tanimoto, the "intelligence required to make such predictions into the system is an incorporation of one kind of agency—the ability to act autonomously. Agency is commonly associated with life and liveness. One might argue that here, liveness has spread from the coding process to the tool itself."[56] These technical developments promise improved human-machine interfaces and improvisation with predictive coding (modeled on previous habits) supported by faster response and processing times, seemingly maximizing the potential for flow, for increased spontaneity, and for more immediate coding on the fly. Rather than a passive tool, the computer is afforded a degree of decision-making responsibility based on its

capacity to second-guess, preempt, or predict the next step within a creative flow of action as it learns more about a performer's preferences and tendencies.

Certainly, improvements in semantic feedback and the development of tactical prediction and autocompletion could enable greater possibilities for live improvising "in the moment" or "in the zone."[57] However, such technological advances might not necessarily give rise to new forms of improvisatory liveness conceived as a mode of live thinking-in-action, of real invention and intervention. Indeed, the *pressure to perform*—even entertain—within our contemporary reputation economy places high demands on the performer. Under these conditions, technical developments might serve the betterment or even efficiency of a given performance—that is, become instrumentalized at the service of technical virtuosity through reducing the potential for error alongside increasing the performer's capacity for repetition of complex sequences of coding.[58] Here, the intrinsic motivation associated with flow states (heightened value on the process and its challenges) gives way to a form of extrinsic motivation dependent on external factors, including the success of the performance product, which becomes measured according to the normative criteria of acceleration and immediacy: the eradication of error and delay in favor of more easily attainable complexity and precision, speed and efficiency, and productivity and repeatability.[59] Certainly, speed and immediacy can easily become mistaken for liveness, which in turn can result in less risk. Indeed, rather than measuring the success of human-computer improvisation based on how effectively technologies facilitate the process of *creative flow*, the collaborative potential of improvising with computers might also recognize the critical value—even a different sense of computer agency—within moments of delay and technical resistance.

Although liveness refers to the higher levels of immediacy or even efficiency within increasingly real-time computational processes—in response to the demand for ever-faster transactions and instantaneous feedback—it also speaks of the potential for the unpredictable within performance, the capacity for risk, error, and serendipity.[60] Indeed, some live coders might even introduce errors and crashes on purpose, whether by coperformers interfering with each other's code, by ixi lang's "suicide" function (which causes the process to shut down at a random moment), or just by closing their laptop to abruptly stop the sound. So how might technological innovation increase the potential for the liveness of live coding, where a heightened level of collaboration or coimprovisation with machines allows for the emergence of the unexpected by combining increased performance capacity with a continued embrace of risk and uncertainty, even the affirmative potential of error and failure? Indeed, has the fetishization of liveness been replaced with that of immediacy—the conditions of contemporary life marked by the loss of reflection space and intervals?[61] How can the relation between liveness,

immediacy, and efficiency be uncoupled? How might predictive technologies—perhaps counterintuitively—enable that which cannot be predicted in advance, a truly experimental or surprising mode of liveness made possible through a more intuitive collaboration between human and nonhuman? For example, live coder Elizabeth Wilson describes being

> drawn to the idea of being able to share creative responsibilities with the computer. I've wanted to avoid the constraints of gestural control that comes with most musical interfaces. I found that by automating processes that previously required manual skills leaves more mental capacity for traversing unexplored areas of creative space and uncovering new territories of ideas.[62]

Within live coding, code is conceived neither as a *passive tool* nor an *autonomous predictive process* but rather as a material (or even collaborator) that the coder works with and can be surprised by. Indeed, the liveness of creative flow refers less to the quantitative speed of action or immediacy of feedback and more to the quality of its creative interactions. For musicologist Marc Leman, interactive flow is facilitated through embodied cognition involving dynamic, prereflective processes—whether playing an instrument, dancing, listening, or using new interactive technologies—in which one grasps the situation even before one is fully aware of it and responds on the fly. His "expressive moment" resonates with the notion of kairotic timeliness previously discussed (in chapter 4 and then elaborated in chapter 6), in contrast to the timeliness of machinic immediacy promised by real-time feedback.[63]

The Performativity of Live Coding

These different modes of liveness—liveness as the immediacy of execution between the writing of code and its effects, liveness as a dynamic process involving the interaction of unpredictable forces, liveness as a coemergent process—can be further elaborated with reference to the concept of performativity. Or, rather, different species of liveness become foregrounded through reference to different conceptualizations of performativity since this concept—like liveness itself—has varying meanings and values within different disciplinary and cultural contexts. The intent, however, is not to rehearse a comprehensive survey of performativity but rather to point to different lineages of thinking that offer alternative perspectives on the liveness of live coding. On the one hand, the performativity of live coding can be conceived through the prism of *speech act theory*, derived originally from J. L. Austin's much-referenced *How to Do Things with Words*, from 1955.[64] In contrast to *constative language* (descriptive language that can only be verified as true or false), Austin argued that a performative utterance (provided it is uttered in a "felicitous context") does what it says.[65] Additionally, beyond simply

enacting what it says, performative language actively creates: it is operative in the sense that it brings something into existence. It not only makes a statement but also performs an action. In these terms, live coding's programming languages can be said to be operative—they do what they say; moreover, they do what they say at the moment of saying it.[66] Austin's conceptualization of performativity emphasizes the executive function of language, and live coding arguably exemplifies this species of performativity. In these terms, the performativity of code is maximized by minimizing the gap or latency between utterance and its effect, between code's *speech act* and its resulting action.

Working in the tradition of science and technology studies, sociologist Adrian Mackenzie argues that the "performativity of code" offers a challenge to the commercial imperative of software development and also the social relations associated with it.[67] His example is the Linux operating system, the most pervasive example of free/open-source software development, not least through the enforcement of its GNU General Public License. Like the work it does, the coding performance disrupts the normative relations associated with work, what code does when it runs, and who benefits from this. Any sense of agency assigned to code—that Tanimoto identifies, for instance—relies on the relation of "code's existence as both expression and process."[68] It is inherently performative in this sense, and in the case of live coding, the performative act of coding becomes a prototype for its further action—rehearsal and performance at once, a score performing itself.[69]

Although this operative aspect of software is clearly one of its key attributes, many thinkers working in the software studies tradition have attempted to problematize the understanding of coding according to Austin's model of performativity. In "Language Wants to Be Overlooked: On Software and Ideology," writer and computer programmer Alexander R. Galloway makes the assertion that code differentiates itself from writing by doing what it says it will do but calls for an understanding of code in its own terms rather than anthropomorphizing it as speech or performance.[70] In contrast, taking issue with what she considers to be the inevitable anthropomorphism in Galloway's essay and drawing upon her background in both literature and systems engineering, Wendy Hui Kyong Chun asks: "How can code/language want—or more revealingly say—anything? How exactly does code 'cause' changes in machine behavior? What mediations are necessary for this insightful yet limiting notion of code as inherently executable, as conflating meaning and action?"[71] Her point, in drawing attention to the mistake of separating instruction from execution, is that code does not always do what it says but does things in a "crafty, speculative manner in which meaning and action are both created."[72] She is interested in how a whole series of operations produce something we refer to as source code and in how this in itself is something imprecise.

Indeed, to claim something as *source* appears essentialist since it is not a thing but rather a set of relations. There is a temporal complexity to this: it becomes source only after the action has taken place—"Source code only becomes a source after the fact."[73]

More to the point, Chun argues that source code becomes a substitute for a range of other operations and signals. Drawing on the work of Jacques Derrida, she asserts that "source code becomes a source only through its destruction, through its simultaneous nonpresence and presence.[74] . . . It is neither dead repetition nor living speech; nor is it a machine that erases the difference between the two."[75] It is undead writing. Live coding appears like an instantiation of this state of undeadness, as coders and codes call up preexisting code and layers of code to make them self-evidently *live*, oscillating between what is understood to be readable, writable, and executable. Chun conjures up the undeadness of information as well as its relation to the commodity fetishism and the undead labor of technology as a "ghostly abstraction" that comes to haunt all technologies.[76] Thus, coding practices can be seen to be acts of "sourcery," full of imaginative potential based on the (undead) logic of programmability, what Chun refers to as "'programmed visions' which seek to shape and to predict—indeed to embody—a future based on past data."[77] Live coding might also be understood in this way as an attempt to reanimate dead materials—both in terms of coders and codes—highlighting the potential to draw together meaning and action across multiple layers of operation. Live coding is both live and *a*live (where the prefix "a" here indicates neutrality), its code immediately executable and also capable of being put on hold, suspended in animation until an auspicious point is reached for bringing it back to life.

Codes operate in oscillation and with the impossibility of dealing with a fixed state. The performativity of live coding effectively exposes its own *mediality*, as a means to another state, without closure—with reference to philosopher Giorgio Agamben's writing, as *gesture*. If gesture is generally something that is considered to be lost in contemporary culture—as Agamben thinks—then live coding perhaps recovers some of its dynamics. For Agamben, "The gesture is the exhibition of a mediality: it is the process of making a means visible as such. It allows the emergence of the being-in-a-medium of human beings and thus it opens the ethical dimension for them."[78] He argues that the event of language is political inasmuch as it relates to the free use of "pure means," as "politics is the sphere neither of an end in itself nor of means subordinated to an end; rather, it is the sphere of a pure mediality without end intended as the field of human action and of human thought."[79]

Perhaps, as Virno argues, "what really counts is the act of enunciating and not the text of the enunciation."[80] This can be understood to relate to theorist Dieter Mersch's assertion that "aesthetics focus on the singular, on this-here or something that can

be shown. Yet it is never clear what (*quid*) it is, only that (*quod*) it is."[81] He argues that "we are dealing with 'showings' that in equal measure reveal something and show themselves while in showing, hold themselves back. . . . Their métier is not representation, but presence."[82] Here, the communicability of live coding exceeds language (code conceived only as performative utterance in Austin's terms): its performativity operates *with*, *in*, and *through* the materiality and mediality of the performance itself, through "surrender(ing) to the event and its experience."[83] Following such reflections, the performative act of showing the screen within live coding might be considered less a device for communicating what or how code is unfolding at the level of linguistic or methodological reasoning and more a device to show only that *something* is unfolding live.

Certainly, the idea of performativity is an expanded—and indeed expanding—concept in which the original Austinian conceptualization (with its various Anglo-American derivations, extensions, and critiques) has been deviated within a performance studies context to refer more broadly to the mattering of a performance's performance.[84] Alternatively, within a Germanic context the notion of performativity is untethered from the Austinian emphasis on speech, communication, and linguistic sensemaking and from the overtly pragmatic—even utilitarian—idea of enunciative execution that seeks to achieve a subject's (pre)intended effect and is approached instead through ideas of embodiment, eventhood, and cocreation. Theorist Erika Fischer-Lichte identifies a parallel "performative turn," activated in Germany in the 1960s through the work of literary historian and theorist Max Herrmann, which resists the privileging of text and semiotics by foregrounding the social dynamics of the performance event. Fischer-Lichte elaborates the idea of an "aesthetics of the performative" and the transformative potential therein to address the performativity of performance as one of *autopoiesis*: the self-producing operations of a living system.[85]

For Fischer-Lichte, the "continually operating feedback loop provided in any performance event by the ongoing interactions of performers and audiences" offers an exemplary system of autopoiesis.[86] The "self-organizing system" that Fischer-Lichte identifies within performance-as-event is marked by a sense of contingency. She states, "Contingency became a central aspect of performance with the performative turn of the 1960s. . . . The feedback loop as a self-referential, autopoietic system enabling a fundamentally open, unpredictable process emerged as the defining principle."[87] Significantly, while Fischer-Lichte emphasizes the importance of bodily copresence within this model (between performer and audience and also between spectators themselves), it is for enabling a feedback loop of coproduction or coemergence rather than for acting as a marker of liveness in and of itself. Or rather, this feedback loop *is* the marker of autopoietic liveness within performance.

While Fischer-Lichte is referring more broadly to theatrical performance, her conceptualization of liveness predicated on the feedback loop of autopoiesis resonates with live coding practice. Or rather, live coding complicates Fischer-Lichte's conceptualization since she argues that mediatized performances "sever the co-existence of production and reception. Mediatized performance invalidates the feedback loop."[88] Live coding, in fact, activates a feedback loop between the live performer and the machine, between live and mediatized, establishing the parameters for live modification where both human and computer edit the same script.[89] Feminist theorist Donna J. Haraway argues that what is required is the conceptualization of *sympoiesis*, or "making with," rather than autopoiesis, or self-making.[90] Indeed, live coding necessitates a radical rethinking of the concept of performativity, providing concrete examples of practice that reflect the wider conceptual "turn" from the anthropocentric as emerging within various theoretical orientations, including new materialism, vital materialism, agential realism, and *actor-network theory*.[91] Writing from the perspective of feminist new materialism, Karen Barad proposes "a specifically posthuman notion of performativity—one that incorporates important material and discursive, social and scientific, human and nonhuman, and natural and cultural factors. A posthumanist account calls into question the givenness of the differential categories of 'human' and 'nonhuman,' examining the practices through which these differential boundaries are stabilized and destabilized."[92] Barad argues for relational coconstitutive *intra-actions* between humans and nonhumans, stating that "on an agential realist account, agency is cut loose from its traditional humanist orbit. Agency is not aligned with human intentionality or subjectivity. . . . Agency is the enactment of iterative changes to particular practices through the dynamics of intra-activity."[93] Live coding provides material examples of this intra-activity, the entanglement of human and nonhuman agencies active in a process of collaborative coconstitution.

Live coding sets up an important tension between the human-centeredness of creative action and nonhuman agents that presents new ways of understanding and acting in the world—that exceed what is knowable by the programmer. This also means that a program cannot be reduced to its functional aspects or understood as a tool either. Instead it is a dynamic construction process of description and performance at the same time. Additionally, while live coding performers and audiences might share the same time-space, live coding is not reliant on spatial-geographical proximity for its liveness. Indeed, some live coding performances take place with the performers themselves operating remotely from different geographical locations yet still collaborating within a feedback loop of coproduction.[94]

Relational Liveness

Although agreement on the ontological constitution of liveness arguably remains at a stubborn impasse, Fischer-Lichte's model of performativity does enable a rethinking of what Reason and Lindelof call the "processes of audiencing," the experiencing of liveness and the sociopolitical implications therein. For Fischer-Lichte, a model of performativity based on the notion of "self-generation involves the participation of everyone, yet without any single individual being able to plan, control, or produce it alone. It thus becomes difficult to speak of producers and recipients."[95] In a conventional (entertainment) model of performance-as-product, a performance may well have been crafted and rehearsed in advance or else is dependent on the singular skill and agility of the performer: the audience members are not actively engaged or implicated in the performance; they are merely its recipients, its consumers. Significantly, performative self-generation involving the feedback loop of coproduction or coemergence (as outlined by Fischer-Lichte and described in the previous paragraphs) subverts a model of virtuoso performance—here meaning technical prowess, finesse, or mastery—that all too easily renders the audience awestruck by the aura of the performer, as passive spectators of staged spectacle. This transmissive mode of performance could be conceived in terms of Marxist educator and philosopher Paulo Freire's critique of transmissive education, the unidirectional flow of knowledge from the expert to the ignorant that ultimately serves to oppress, stifle curiosity, and disempower.[96] Drawing on the writing of philosopher Martin Buber, Reason and Lindelof argue that liveness involves the potential for "mutual surprises," echoing the dialogic interplay of a conversation that for Buber is "not one whose individual parts have been preconcerted, but one which is completely spontaneous, in which each speaks directly to his partner and calls forth his unpredictable reply."[97] The sense of dialogic liveness within live coding—coding conceived as a *conversational practice*—refers to both the relation between coder and computer and between collaborating coders and their respective machine, as well as between performer and audience.

Live coding performances have also become open to wider contingencies, such as variances in flows of electricity and other environmental factors that affect performance, where entities coexist in a strange and entangled ecology. Here, *Life Coding* is artist Martin Howse's term to describe some of the practices of live coding to open hardware and broader ecological concerns, thus extending the action of coding to the idea of the world itself as a potential operating system, opening up the wider connections of coding to materialist and environmental concerns. For example, Howse's *Earthboot* (2012–2014) boots computers using electricity drawn from the earth, with

variance in underground currents interpreted directly as code by the microprocessor-based device.[98] In this way both energy and operating system are drawn from the local terrain. This follows a media archaeological perspective and broader ecologies relating to matter—from the extractive use of minerals to broader infrastructures—that have real effects on life systems.[99]

Live coding's performativity involves a complex entanglement of exchanges and feedback loops between different human and nonhuman forces. Yet, beyond the making of mutual surprises, its mode of liveness creates the aesthetic and sociopolitical conditions for the formation of an emergent community inaugurated in and through the shared experience of the unfolding performance.[100] Curator-writer Miwon Kwon uses the term *temporary invented community* to describe those specific social configurations that are "newly constituted and rendered operational through the coordination of the art work itself,"[101] produced through a form of "collective artistic praxis."[102] Here, beyond the making of the performance itself, the shared experience of live performance can involve the constitution of a *public*. In these terms, performance *precedes* public, or rather, it is that which brings its community into being. Anthropologist Christopher Kelty nuances this idea as a "recursive public . . . a public that is constituted by a shared concern for maintaining the means of association through which they come together as a public."[103] Taking his example from free and open-source software development, the idea of a recursive public operates in a dual sense "first, in order to signal that this kind of public includes the activities of making, maintaining, and modifying software and networks, as well as the more conventional discourse that is thereby enabled."[104] Second, he argues, the use of the term is to "suggest the recursive 'depth' of the public, the series of technical and legal layers—from applications to protocols to the physical infrastructures of waves and wires—that are the subject of this making, maintaining, and modifying."[105] For Kelty, what makes recursive publics distinctive is the capacity to create infrastructure as an "activity of being public or contesting control" without this becoming sovereign, alongside the "ability to 'recurse' through the layers of that infrastructure, maintaining its publicness at each level without making it into an unchanging, static, unmodifiable thing."[106] The practice of live coding creates and maintains the recursive conditions of its own coming into being through the inauguration of an entire living ecology.

Like free and open-source software development in general, live coding is performed in public in multiple ways—at live coding public performances, of course, but also through the ongoing collective and collaborative processes of its production and distribution—challenging some of the normative social relations of production as discussed elsewhere in this book. For live coder Iris Saladino, "Communities operate by decentralizing and horizontalizing information, processes, events, and decisions with respect for others as

the core of all interactions. We aim for collaboration and we mistrust competition."[107] Live coding performances and festivals are often staged over long durations, interwoven with ad hoc workshops with, as artist-programmer Simon Yuill notes, "participants learning and adapting the tools of the performance as they take place."[108] As live coder Abhinay Khoparzi states, "Performances themselves become mini-workshops where one-on-one and one-to-many interactions with audience members can develop into a learning space."[109] For Yuill, live coding's emphasis on practice over end product can be conceived as the "virtues of practice," where it "is practice that is consciously linked to, and helps define, particular practitioner communities: groups defined not by a common aesthetic, style, nor common collection of cultural references, therefore, but by commitments to shared practices."[110] Live (living) coding environments are produced, structured, and circulated through lived communities, code repositories, and software distributions (such as in the case of the communities around SuperCollider, Sonic Pi, TidalCycles, ixi lang, or Hydra). Clearly, there is a further relation to other community formations (such as hacklabs and maker spaces), but again the real-time aspects and its potential for fast prototyping and distribution distinguishes the practice of live coding as a recursive public in Kelty's sense, integrating what Yuill describes as "the distribution of the knowledge of how to produce into that which it produces."[111]

Strongly held copyleft and creative commons principles underpin live coding as a community in "common." Indeed, within live coding the principle of making visible its process extends beyond what might usually be considered the performance itself. The audience often witnesses the processes of preparation as much as what might conventionally be considered the performance. Referring to Andrew Sorensen's live coding practice, Stephen Ramsay states that what we witness at the start of the live coding performance "is the equivalent of watching a symphony warm up."[112] Arguably, there is something more radical at play, where the revelation of preparation is less akin to the symphonic process of tuning in or up (which might be witnessed *ceremoniously* in theaters but is still not considered part of the performance). Instead, for many live coders the back(stage) and front(stage) of the process are indivisible.[113] Within some live coding performances, there is no concealment of preparation or setting up, no cuts to be made after the fact—all is visible, all part of the work, an apparatus laid bare. Preparation becomes folded into the practice itself; it is part of (and not prior to) the performance.

This *showing* of the backstage of the performance process (the activities of warming up, of setting up the parameters of one's environment, of the defining of functions), alongside the revelation of the script (as it is being modified synchronous to its execution or actualization), differentiates live coding as a critical practice engaged in the interrogation and exposition of its own means of production and working processes.

For Yuill, "Live code is unashamed to expose the bare materiality of its production . . . akin to revealing the stage machinery in a Brecht play. It creates a virtue by exposing something that is normally concealed."[114] In so doing, the live ecology of live coding practices appears to resist or refuse the conventional idea of the virtuoso performer, the *auratic* authority of the individual coder demonstrating *mastery* of their art.[115] In these terms one might also situate live coding within a wider lineage of post-1960s performance practices that reject virtuosity (and its willful concealment of the practice or effort), instead exposing the mechanisms of production, including the incorporation of rehearsal and preparation into the space-time of the performance itself.[116] Here, as curator Catherine Wood asserts, "The 'making' of the work is simultaneous with and dependent upon its 'doing' as performance; it is process *and* action. Tautologically, the work (of art) is simultaneous with the work (effort) of its execution."[117] Or, as Yuill asserts, live coding involves the "presentation of the 'work' itself as an open-ended mutable piece of code rather than as a static discrete artefact."[118]

This chapter has drawn on different theoretical ideas of liveness and performativity to demonstrate that live coding does not sit easily within any singular theoretical framework: its liveness is to be apprehended from multiple interdisciplinary perspectives, in which the *hierarchy of liveness* conceived in computational terms collides with a wider discourse on embodiment, vitality, and performativity. Certainly, the liveness of live coding is enabled by technical advancements that make the real-time nature of its live performance possible. However, this significant shift in programming capability is more than technological: the liveness(es) operative within live coding open(s) up new reflections on the ontological and ideological questions of what constitutes liveness, where live and mediatized modes of performance interweave, and where human and nonhuman livenesses become entangled. Indeed, the mixed livenesses of live coding—oscillating between states of liveness, *a*liveness, and undeadness, and between machine and human liveness—invite further investigation of its multi- and microtemporal states, extending beyond the issue of liveness to that of live coding's *time criticality*, which becomes the focus of chapter 6.