Musical Live Coding in Relation to Interactivity Variations

Musical Live Coding in Relation to Interactivity Variations

GEORGIOS DIAPOULIS

Chalmers University of Technology, Gothenburg, Sweden
Email: georgios.diapoulis@chalmers.se

This article explores the similarities and differences between live coding and traditional music performances. The focus is on how bodily movements are expressed and whether pre-reflective processes may be activated during a live coding performance. While reports from practitioners vary on percepts of embodiment, the community is missing a theoretical background that reflects on practice. Understanding pre-reflective processes in live coding can benefit performance practices and tool development. As a live coder, I reflect on personal experiences and explore what I call 'interactivity variations', a term to denote different gestural manners of interactions during a performance. I observe patterns of embodiment among various live coders who use diverse performance systems from online videos. Out of 11 examples of performance systems, two cases demonstrate interactivity variations that can activate pre-reflective processes while another exploits direct manipulation. I present some implications for the patterns of bodily movement during a live coding performance and discuss how descriptive and prescriptive notation can be important and potentially influence our sensorimotor network. The article contributes a first account of a sensorimotor theory on live coding performances, reflecting on practice and embodied music cognition by presenting an aesthetic analysis of 11 online video examples.

1. INTRODUCTION

Live coding and traditional musical performance can be viewed as the two extremes on an imaginary continuum of music performance studies. While motoric and cognitive skills are demanding for both performance styles (Palmer 1997; McLean 2014), I will primarily concentrate on the qualitative differences that arise from the motoric skills performed through bodily gestures. This discussion will establish the foundation for my main research question: Are pre-reflective processes evident in live coding performances? The term 'pre-reflective process' is used interchangeably with 'fast processes' or 'subconscious processes' (Leman 2016).

My primary theoretical framework is drawn from music psychology and music perception research on music performance, employing Leman's theoretical framework on pre-reflective processes for expressive interactions. I subscribe to the phenomenological approach of autopoietic enactivism (Varela, Thompson and Rosch 2017), a sensorimotor theory of embodied cognition that sees cognition as an emergent phenomenon of sensorimotor activities. I also investigate how the user gesturally interacts with the input interface, which may involve addressing human–computer interaction (HCI) as needed. Essentially, the main emphasis is on the live coder, with a secondary focus on the physical interface and programming language. The field of research on music performance is largely indifferent to the specific musical instruments used. Instead, it focuses on musical structure, bodily movement and emotional responses (Palmer 1997). I concentrate on bodily movement and hope that future research will also look into musical structure and emotional responses, as they are currently unexplored areas in live coding.

Here, I describe how live coding is conducted by combining approaches from both HCI and music psychology, with an emphasis on gestural interactions. The term 'gestural interactions' refers to both musical gestures (Leman and Godøy 2010) and bodily gestures used in HCI, with gestures, viewed in the context of human–material interactions (Ishii, Lakatos, Bonanni and Labrune 2012), and the material, in this case, being sound. Musical gestures unify bodily movement and meaning and can therefore serve as a means of studying subjective percepts resulting from bodily movement (Jensenius, Wanderley, Godøy and Leman 2010). My motivation is to explore what can be learnt about a performer's cognitive and sensorimotor processes through observation of various systems and practices.

2. METHOD

In this article, I discuss 11 performance systems and practices, primarily by conducting complete and unstructured observations of live performances I have attended or watched online, reviewing a diverse corpus of literature and reflecting on my experiences as a live coder. Here, the literature spans music psychology and perception studies and research on live coding and HCI. Unstructured observations refer to the idea that an observation is carried out without systematic

Organised Sound **00**(00): 1–13
doi:10.1017/S1355771823000444

criteria guiding the observed showcases. It is an approach commonly adopted when there is little theoretical background on the phenomenon being observed or when the researcher does not know the outcome of the study (McKechnie 2008). Complete observations involve the observer refraining from interacting with the participants during the observation. In the case of live coding, complete observational studies have recently started to appear (Diapoulis and Dahlstedt 2021a) due to the increasing availability of online audiovisual material.[1] Ten years ago, such observations were not feasible due to the scarcity of online resources.

Thus, unlike observational studies in ethnomusicology, the methodology used here does not involve a coding scheme or other systematic criteria. Instead, the observation is an ongoing process documented through reflective diaries or other forms of documentation such as sketching diaries. Personal practices have certainly influenced the selection of the 11 video examples. Over the last year, I selected examples that I encountered, and together they formed a structured set of live coding systems that illustrate diverse practices of interactivity variations[2] on single-person examples (see section 5). Live coding is a community that documents itself (Haworth 2018) and reflects on itself, and my intention is to contribute to the research goals and ethos of the community, of which I consider myself a part.

3. TRADITIONAL AND LIVE CODING MUSIC PERFORMANCE

In traditional music performance, whether instrumental or vocal, there exists a direct relationship between energy and sound (Leman 2007). Put simply, the more effort we exert in striking a drum, the louder the resultant sound will be. This allows performers and audiences to engage with the process of sound generation, resulting in a more rewarding and enjoyable experience. Such a reward mechanism is related to our ability to make predictions between bodily movements and resultant sounds. For example, when observing a drummer lift her hand to the sky and strike the snare drum, we expect a loud and powerful sound as a result. Conversely, live coding does not involve this relationship (Dahlstedt 2018). In live coding, performers experience an indirect involvement with the music due to the use of notation (Nash 2012). There exists a dissociation between action and perception, as the bodily gestures performed by the live coder do not directly correspond to the production of sound. This can result in minimal bodily movement, which raises questions about the significance of typing on a keyboard (Haga 2008).

This indirect level of description is why live coding has been referred to as a *propositional improvisation* practice (Goldman 2019) and a radically different form of performance compared with traditional music performance (Sayer 2015). Goldman explicitly states that he does not practise live coding, while Sayer's study does not specifically reflect on practice. On the other hand, practitioners view live coding as necessitating a distinct range of sensorimotor skills. For example, Fredrik Olofsson does not feel any lack of immediate feedback, regardless of the non-physical interactions involved during coding (Nilson 2007).

Music performance is an embodied practice (Palmer 1997; Godøy 2021). Embodiment, here, 'assumes that subjective experiences are expressed in bodily changes' (Leman 2007: 236). During live coding, our subjective experiences include how we reason about the running program, how we appreciate the musical outcome and how we plan the progression of the performance (Diapoulis and Dahlstedt 2021b). All these are expressed in bodily changes, which can range from minimal keyboard movements to overt dancing to the beat. These movements can be synchronised and non-synchronised with the musical outcome.

Live coding practitioners have reported percepts of embodiment during performances, though such reports can be contradictory. According to Baalman (2015), embodiment is so profound in live coding that even grammars and programming languages can shape motoric execution patterns. By contrast, Hutchins (2015) sees a lack of embodiment in coding and aims to achieve it through live patching with synthesisers, because of tactility percepts. A recent workshop at NIME on gestures and embodiment in live coding also suggests diverse understandings of embodiment within the community (Salazar and Armitage 2018).

4. FROM NOTATION TO SOUND AND FROM SOUND TO MUSICAL MINDS

Live coding is sometimes described as a state of mind (Tanimoto 2015). It blurs musical concepts such as composition, performance and improvisation, while also challenging standard views on programming, such as the typical software engineering development cycle. The difference between live coding and most music improvisation practices is that live coding is an improvisation practice based on notation (Baily 1999; Magnusson 2019). Moreover, the notation is written on-the-fly, and can also be maintained with its accurate temporal evolution (Rohrhuber, de Campo,

[1]TOPLAP archive.org (https://archive.org/details/toplap), and YouTube Eulerroom (www.youtube.com/@Eulerroom).

[2]A term inspired by the established term in music psychology *performance variations*, which refers to different manners of expressive performance.

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press

Wieser, Van Kampen, Ho and Hölzl 2007), thus can be reproduced by a machine. The standard paradigm of musical live coding involves a composer-programmer typing on a keyboard and sharing his/her screen with the audience. Roberts and Wakefield (2018) have referred to this standard paradigm as canonical live coding. This term refers to the practice that first appeared in the early 2000s, where a command line prompt was used as an interface equipped with an interpreter or a compiler. Interpreter-based languages are more popular due to their runtime immediacy on the musical outcome (Collins, McLean, Rohrhuber and Ward 2003).

When executable code chunks are rendered to audible sounds, the coder faces the indirect involvement between code and sound. Long sequences of typed individual characters are enfolded in the many steps required to evaluate a code expression. This is known as the complexity of the interface in HCI (Myers 1994), which refers to the number of steps required to perform a programmable action, such as opening a text document using drop-down menus on a text processor. In the cognitive dimensions of notation, a similar notion is that of the *closeness of mapping* (Blackwell and Collins 2005), a term from the psychology of programming that indicates how notation relates to output (McLean and Wiggins 2011). It is reasonable to assume that not all live coders are trained in blind typing techniques or use the same type of keyboard. Thus, the multiplicity of typing a single line of code can be enormous. Typing is also an open-loop motor program (Palmer 1997), which makes it an activity prone to errors, as there is no informative feedback mechanism to adjust or correct ongoing movements.

When listening to the generated sounds, the live coder somehow makes sense of the relation between the live writing of the code and the musical outcome. This may be seen as a mental link between the code and the sound, which enables the performer to listen to how structured code executions sound and imagine how novel code evaluations may sound. Thus, in live coding, both musical imagery and music listening have central roles during music-making (Diapoulis and Dahlstedt 2021a). Musical imagery is a mental process that can induce musical experiences or elicit musical realisations. It is a multimodal phenomenon, either auditory, visual or motoric, in which we anticipate desired effects or experience music both voluntarily and involuntarily (Jakubowski 2020).

Voluntary musical imagery, also known as *online musical imagery* to denote the imagery present during a performance, contributes to planning future actions (Bishop, Bailes and Dean 2012). Involuntary musical imagery can be induced by a phenomenon known as *notational audiation* (Brodsky, Kessler, Rubinstein, Ginsborg and Henik 2008; Keller 2012), a type of visual imagery. This is when we can internally 'hear' a melody depicted on a score. Motoric and auditory imageries are interrelated, and one can influence the other, also known as crossmodal interactions. A delay between performed actions and auditory feedback, which can be significant in live coding due to the complexity of the interface and the very poor closeness of mapping (Blackwell and Collins 2005), disrupts action-perception couplings, in turn disconnecting the temporal precision of motoric actions with listening percepts. This phenomenon is known as *delayed auditory feedback*, where action and sound are several milliseconds apart or *altered auditory feedback* when the performed action does not match the heard sound (Palmer 2012). These discrepancies can increase the gap between action and perception and hinder voluntary imagery and, consequently, embodiment. In this context, the visual aspects of the code or coding environment that can induce involuntary imagery (i.e., notational audiation) may play an important role.

When there are the disruptions in auditory feedback, how can we still embody the sound in our performance? The overt embodiment in live coding can be induced through listening to the sound, as we can exclude any tactility percepts that carry vibrations. Additionally, simulating motor actions through visual perception is possible, but it may be challenging for live coders to direct their attention away from the coding interface. In such situations, peripheral vision could be useful; for example, to be aware of people dancing nearby. Sometimes, we embody the generated sound by nodding to the beat or dancing. These are secondary aspects of musical gestures, which can have sound-accompanying or communicative functions (Leman and Godøy 2010; Jensenius et al. 2010) and do not have sound-producing functionality. Of course, this is more likely to happen when the musical outcome affords such embodied percepts, typically observed during Algorave parties, which are live coding events in dance clubs (Collins and McLean 2014).

Ultimately, the generated musical outcome facilitates an understanding and aesthetic enjoyment of the running program. One can say that the live coder uses the generated sound patterns as a proxy to form a mental model of the running program (Diapoulis and Dahlstedt 2021b). A mental model is 'any internal representation of the relations between a set of elements' (American Psychological Association n.d.) and denotes internal representations of relations and reasoning (Kosslyn 1996). Coding is an act of reasoning, although I will later argue that in live coding, reasoning on-the-fly can have more intuitive qualities when coupled with listening to the musical outcome, thus bringing forth percepts and imageries.

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press

I build upon a sensorimotor theory of embodied cognition and adhere to the autopoietic enactive view of embodied cognition, which avoids distinguishing between mental and non-mental processes (Shapiro and Spaulding 2021). Here, I explain how we live code and how sound, mediated by code, affects our bodily experiences. As such, while the term *mental model* may not be the most historically accurate, I use it to describe the phenomenon I experience as a live coder or how we make our understanding from code to sound and vice versa.

We have seen, so far, that canonical live coding involves a complex interface consisting of typing on a keyboard. This amplifies our indirect involvement between the code and generated sounds and the perceived alteration of the auditory feedback, one that is not bound to physical actions. Besides these obvious challenges, musical notation is known to give rise to the phenomenon of notational audiation, which can induce involuntary imagery in performers. Musical imagery is known to give rise to percepts of embodiment and can also initiate motor activity. As we cannot engage with musical sound using sound-producing musical gestures or primary aspects of musical gestures, there is a dissociation between action and perception. Instead, we can only exploit secondary aspects of musical gestures, such as full-body movements, and examine how micro-movements such as typing can be useful. So far, I have only presented typing in live coding mostly as an activity that hinders embodied percepts based on action-perception theory.

4.1. How the interaction is performed

During a canonical live coding performance, the user interacts in a similar way to how end-users type in a word processor software. One persona of Nick Collins, known as Click Nilson, used a cloud-based[3] word processor to collaboratively re-write his presentation script with the audience during the live code festival symposium in Karlsruhe in 2013 (Nilson 2016). The main difference when a text editor is used alongside a programming environment for music-making is that the code evaluations are performed when a code chunk can be executed, which usually requires the conscious allocation of resources. Short edit-execution cycles, such as numerical parameter adjustments, are unlikely to enter conscious awareness, as this may take up to a few milliseconds. An average typing speed is 60–80 words per minute (wpm), or 3–4 characters per second (Marklin and Simoneau 2004). Short edits may be performed even faster than that, potentially approaching the upper limits of delayed auditory feedback, as the fastest typists can exceed several hundred wpm.

This strategy, which is predominantly found in full slate[4] live coding – an approach where pre-written code is used during a performance – is widely used by coders as a risk management technique or as a means of maintaining pace and flow (Roberts and Wakefield 2018). This full slate approach, a weak approach to live coding (Magnusson 2014b), exploits pre-written code. Thus, it can be common that the performer is only editing the code instead of writing the performance script on-the-fly, which Magnusson refers to as the strong criterion. Identifying the limit between a weak approach to live coding and simply a generative reproduction of the material can be difficult, similar to how hard it is to draw the line between interpretation and improvisation (McPherson and Limb 2019).

It is generally accepted that typing on a keyboard is an activity based on serial actions. This has similarities with traditional music performance, as *serial skilled actions* are carried out during a musical performance (Palmer 1997). (In fact, sequence production spans a variety of daily activities such as speech and walking.) The main difference between live coding and traditional music performances is that the live coder does not make sound-producing bodily motions. This makes live coding a performance practice that can hinder engagement with audiences and co-performers related to embodied experience and the aesthetic enjoyment of the music (Brattico, Brattico and Jacobsen 2009). The main reason is that when humans observe sound-producing movements, they can mentally simulate these actions in their sensorimotor network (Keller and Appel, 2010; Keller 2012). Sometimes this can also produce overt bodily movement, such as dancing (Keller 2008). The same does not apply to live coding, as the performed typing actions are mostly rendered in an unordered manner, or non-linearly, in terms of both temporal order and motoric sequential patterns. Commenting on the temporal aspects of live coding, Emmerson (2017: 115) states that 'the code writing of deferred time computer programming may be assembled out of time order'. In terms of how motoric sequences unfold, this means that the same typing sequence can be produced in different ways. Thus, audience engagement via mental simulations of observed bodily movements cannot easily happen. Instead, the audiences and co-performers tend to bodily entrain through music listening and bodily movement. Maybe the only aspect we can mentally simulate is an expectation about a new code chunk evaluation by simply seeing the visual highlighting of the code. Hernani Villaseñor (2019) has been performing live while filming his typing. This can be seen as a 'virtual action-sound mapping' (Jensenius

[3]I personally attended Click's Nilson presentation.

[4]Full Slate https://youtu.be/zJFJEqblEIA (accessed 14 April 2023).

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press

2022: 155), denoting the multiple layers between action and sound, coupled with the discrete action of running the code. Live coders exploit this relation between code executions and resultant sounds. I usually do not look at the interpreter's output following every code evaluation; instead, I expect to hear the sonic changes.

During a live coding performance, novel code evaluations can have entirely predictable or not-at-all-predictable musical outcomes. Typically, a novel musical outcome may induce surprise to both performer and audience, expressed as a violation of expectations and may arise from creative explorations of the generative space (Dahlstedt 2012). Maybe the most common example of when our expectations are violated in a live coding performance is when a programming error occurs, and the generated sound patterns stop. The same can happen when we explore a generative music space, and the musical parameters suddenly tune into silence. This can make a live coding performance alien to an unfamiliar audience. The unfamiliar spectators may be ignorant of how the sound is produced, as reported in the documentary *Show Us Your Screens* (McCallum 2011). Maybe the audiences' perception will change as live coding becomes more popular. Magnusson (2019) reports that audiences unfamiliar with programming can follow the sound-generation process by attending to the live writing process. Visual aspects such as code highlighting are particularly important to this end, and the progress of special-purpose text editors, such as Gibber (Roberts and Kuchera-Morin 2012) and Strudel (Ross and McLean 2023) interfaces, has been tremendous in the last decade.

To summarise this subsection, canonical live coding involves typing on a keyboard. I have presented the activity of typing as serial skilled actions, which also applies to traditional music performance. However, whereas in traditional performance, the serial actions are tightly coupled in time and can give emergence to sensorimotor synchronisation, in live coding, the serial actions are out of time order. This suggests that using typing to interface with the programming language is not likely to result in the development of entrained bodily movements such as coordinated and rhythmic actions. Furthermore, it is unlikely that typing can induce mental simulations due to the multiplicities of individualistic typing styles, keyboard layouts and the error rate of open-loop motor programs. I can only speculate that exploiting typing gestures may induce mental simulations to co-performers and audiences on larger temporal chunks, where each micro-movement loses its significance and the gestalt of the typed chunks matters. But would that apply only to those exposed to live coding practice? Is familiarity important here? These are some open questions.

4.2. Pre-reflective processes in machine musicianship

The effect of action and perception on music cognition has been studied extensively. There is growing evidence that these two are inseparable from one another. We make our understanding by enacting our world (Varela et al. 2017), and the role of the human body is of fundamental importance. Here, musical gestures are seen as non-verbal communication, which has pro-social functions and can enable 'pre-reflective experiences' (Leman 2016: 92). For example, in electronic dance parties, a sweep from low to high frequencies may result in excitement, manifested in people raising their hands. How pre-reflective (or subconscious) processes are manifested in live coding has been barely discussed (Diapoulis and Zannos 2014; Dahlstedt 2018). This can be important as it will enable a discussion on how to better conduct live coding sessions and, consequently, how to better coordinate live coding ensembles.

I have already discussed that traditional music performance is tightly connected to bodily movement, but the movement is also related to expressive performance. Several mechanisms facilitate musicians' engagement with expressive performance (e.g., planning and anticipating future actions), entrained processes (e.g., loosely coupled bodily movement) and reward mechanisms linked to emotional resonance and aesthetic enjoyment (Leman 2016). As the embodied paradigm of music performance has gained more attention, entrainment is considered one of the fundamental mechanisms during music performance (Repp and Su 2013; Clayton et al. 2020). Entrainment is a phenomenon that appears in several mechanical systems and may bring about stability in oscillatory systems, regardless of any small perturbations within a system. Recent studies suggest that sensorimotor synchronisation (SMS) and coordination are responsible for the emergence of entrained processes during a duet performance (Clayton et al. 2020). SMS is a pre-reflective process roughly within 200–2000ms (ibid.), whereas coordination emerges at larger timescales of more than two seconds but may indicate coordination over several seconds to minutes. Godøy (2021: 2) also identifies roughly the same time intervals (0.3–5s) as being of utmost importance during a performance, describing it as the 'enigmatic relationships between notions of continuity and discontinuity in both philosophy and psychology'. Indeed, timescales below 200–300ms are mostly concerned with sound quality, including its early reflections within the acoustic surrounding. For longer durations, we mediate sound through our bodies and make music realisations. Over larger timescales, we coordinate, and social phenomena emerge as the outcome of complex interactions, such as a music performance in public.

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press

These indicative timeframes would suggest that SMS is essentially impossible during canonical live coding. Sayer (2015) supports this argument while Goldman (2019) suggests that live coding can be seen as a *propositional improvisation* practice, which operates on long-term memory mechanisms. Goldman (2019) and Sayer (2015) suggest that live coding incorporates slow, conscious processes, making the emergence of fast, pre-reflective experiences impossible. Several reasons are offered for this, such as the content of feedback, the temporal nature of the feedback and the discrete nature of decision-making (Goldman 2019).

It is indeed true that the so-called canonical live coding incorporates mostly slow processes. Several systems, such as ixi-lang and TidalCycles (McLean and Wiggins 2010b), have been developed, taking such perspectives into account. Indicatively, ixi-lang was developed with the constraint of writing and executing a code chunk in less than five seconds (Magnusson 2011). Even this may not help in enabling fast, pre-reflective processes in a live coding session. So, the question is immanent: Can pre-reflective processes be enabled during live coding performance?

4.3. Summary on interaction, perception and cognition

I would argue that from a theoretical perspective, fast and entrained processes are unlikely to occur when using typing-based live coding systems. Entrained processes can afford small perturbations and eventually reach stability, but the multiplicities of typing a single line of code can be devastating to SMS. We can see bodily entrainment in live coders primarily as dancing to the beat, such as in the case of Algorave parties. But these are merely secondary aspects of musical gestures. It is unclear how these can influence the production of micro-movements. Maybe systems that require short typed sequences to make a sound (e.g., ixi-lang, TidalCycles), albeit being out of order and non-synchronised, can provide the best chance for the case of canonical live coding to employ pre-reflective processes, along with practices such as short edits. Orca,[5] a two-dimensional control interface inspired by trackers that affords single letter commands (for a description, see Blackwell, Cocker, Cox, McLean and Magnusson 2022: 147), operates on stream-based evaluations, which helps to reduce the delayed auditory feedback.

5. LIVE CODING HERE AND THERE

In this section, I present a selection of live coding examples. The reasons are twofold: first, to elaborate on how non-verbal interactions, rendered as musical gestures, are experienced during a live coding music performance; second, to discuss what can be learnt by observing such gestural interactions about the live coders' sensorimotor control and, possibly, cognitive processes. I say possibly because we typically conduct sophisticated experimental designs to merely infer what a person is imagining (Shepard and Metzler 1971; Zatorre and Halpern 2005), which is likely impossible using complete observations.

All the examples discussed here are solo performances, constraining the observed possibilities of interactivity variations. Furthermore, most video recordings from Eulerroom on YouTube or other individual resources, are solo performances. Hence, I limited my study to solos because the embodied interactions I study (e.g., serial skilled actions) are evident when a soloist is bodily interacting with the computer interface, and I do not study inter-musician interactions.

The online video examples were intended to be available to the general public (Snee 2013; Theodosopoulou Bourlogianni, 2021) and no consent was required (Loveday, Woy and Conway 2020) to conduct the observational study and the aesthetic analysis. The links to the videos are shown in Table 1. I compiled this specific list because each video is a good example of each particular system's characteristics. No computational analysis, download or reuse were conducted for this study.

The examples presented in the table are divided into three categories: 1) canonical systems, 2) bottom-up systems, and 3) a mixed category of systems. Canonical systems exemplify the standard paradigm to live coding, as discussed earlier. As canonical live coding is very well known in the community, I will present only one example of a canonical system which really stands out (Baalman 2009). Bottom-up systems are live coding systems that depend on very little abstraction (Diapoulis, Zannos, Tatar and Dahlstedt 2022). Typically, the levels of abstraction are built up on-the-fly, as we go. They are relatively uncommon and highly constrained, which exhibits some interesting characteristics. Also, the bottom-up systems in Table 1 do not use the keyboard, which makes them radically different from canonical systems. The third category presents systems that afford gestural input as direct manipulation (e.g., a mouse), an interface that affords continuous control and facilitates recognition instead of retrieval. I restricted the scope of this category and excluded non-conventional interfaces as the bottom-up category already uses a number of examples with unconventional interfaces. The third category of mixed systems demonstrates some examples that lie between canonical and bottom-up systems, as input control in bottom-up systems mostly relies on recognition instead of retrieval.

[5]https://100r.co/site/orca.html.

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press

Table 1. Chronological list of performance systems in the observational study. *Betablocker*, *Code LiveCode Live*, *CodeKlavier CKalcuλator* and *iMac music* have earlier release dates than the release date of their corresponding videos.

Performer	Performance/System	Video URL	Category	Year
Dave Griffiths	Betablocker	https://vimeo.com/24390484	B	2006
Dave Griffiths	Al-jazzari	https://youtu.be/Uve4qStSJq4	B	2007
Marije Baalman	Code LiveCode Live	https://vimeo.com/434679284	C	2009
Georgios Diapoulis	stateLogic machine	https://vimeo.com/43121821	B	2012
Jonathan Reus	iMac music	https://vimeo.com/205714278	B	2012
Thor Magnusson	Threnoscope	https://vimeo.com/63335988	M	2013
Chris Kiefer	Approximate Programming	https://youtu.be/WwhpRtxq1Kg?t=3417	M	2015
The Duchess of Turing	Using various	https://youtu.be/hfJTF3KTnFM	M	2019
Noriega, F. I. and A. Veinberg	CodeKlavier CKalcuλator	https://youtu.be/hD-PWNDebD4	B	2019
uiae	Using PD	https://youtu.be/A-HohsA9R1o	M	2020
Fredrik Olofsson (redFrik)	SuperCollider	https://youtu.be/lqi-Vqr0qk4	M	2022

Note: B = bottom-up system; C = canonical system; M = mixed category of systems.

Before going to the examples, I want to clarify any confusion in the literature between low-level and bottom-up processes. As Roberts and Wakefield (2018) explain, low-level processes are mostly related to algorithms that operate on digital signal processing (DSP). These also afford low-level computational abstractions but do not necessarily generate or operate on formal languages. By contrast, bottom-up systems necessarily rely on formal languages. DSP algorithms are mostly engineering abstractions for time series analysis and computations. A time series is a sequence of data points, and some applications include forecasting future data points and smoothing noisy data, among others. As such, their primary function is to act as filters of information. A formal language or a computer architecture is not a filter; rather, it affords the universal process of computing any problem given enough time under Turing completeness (a Turing complete machine is a universal machine that can approximate any computable mathematical problem).

I did not include in Table 1 a variation of Betablocker (Bovermann and Griffiths 2014) written for SuperCollider,[6] as it would fit within the canonical live coding systems. In this case, Betablocker (Griffiths 2006) is used as a sound engine, which would spark a new category of low-level systems that can afford Turing completeness. Essentially, the system traverses from canonical live coding to bottom-up systems that generate low-level DSP processes. This example demonstrates how hard it can be to apply 'any easy classifications' to live coding systems (Blackwell et al. 2022: 231), as 'attempts to define this wide field … are likely to become an exercise in herding cats' (Magnusson 2014a: 14).

[6]Tai-studio, An Introduction to Detablocker: https://vimeo.com/32938807 (accessed 14 April 2023).

5.1. The case of canonical live coding

Canonical live coding is the most common approach. This includes the 'standard' interface of a computer screen and a keyboard. Whether gestural actions can be performed on a keyboard is a matter of debate; however, the current consensus is that typing on a keyboard is 'neither observed nor significant' (Jensenius et al. 2010: 16). While this is undoubtedly the case in everyday interactions with computers, the same argument cannot hold in a live coding performance context. Typing is typically observed in this scenario as the performers share their screens. Furthermore, this becomes significant at certain times, specifically when code evaluations are performed and rendered as perceivable sound patterns. The sound vibrates molecular structures in the affine medium, that is, gas, liquid or solid. This transforms the meaning of typing into significant actions, as Baalman (2009; Baalman n.d.) demonstrate in the Code Live Code Live video (Table 1). Baalman switches on the onboard microphone of the laptop, and the typing sounds are used as raw material for the musical outcome. This performance setup demonstrates how typing on a keyboard can be both observed and significant.

Baalman (2015) has reported that our programming language of preference can influence our motoric execution patterns. She elaborates more on embodiment during a performance and provides anecdotal evidence of typing automaticity. The argument is that certain typing tasks have been automatised to such an extent that they require minimum effort. These are known as body schemas, cognitive organisations of one's body that can also monitor sequences of bodily

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press

motions and appear as learned motor patterns (Leman and Godøy 2010). Whether gestural unfoldings, a term to indicate a weak temporality[7] of bodily gestures, may influence our mental model of the running program is an open question (Diapoulis and Dahlstedt 2021a). To elaborate, can this weak temporality of typing influence how we internally represent code-sound relations and reason about the running program? Godøy (2004: 58) has argued that we can mentally 'compress' bodily gestures in time using imagery, what he refers to as gestural imagery. The same cannot apply to sounds: we cannot mentally compress a sound and experience the same characteristics. Maybe gestural imagery can compensate on-the-fly due to our frustration with attending to every single moment of sound and actually influence our mental model (by encodings on our sensorimotor network). Baalman reports that our programming language of preference can shape our motoric actions so that certain sequences are encoded in our sensorimotor network as typing gestures that unfold in time when necessary. If the language of our preference can shape our motoric patterns, then can motoric patterns shape the language we use? Simply put, can the performer's gestures influence the development of the programming language? I will elaborate on these two questions in the next section.

5.2. The case of bottom-up live coding

I argue that bottom-up systems exhibit *interactivity variations* where the gestures can influence the programming language. I will mainly focus on two cases: CodeKlavier CKalcuλator by Noriega and Veinberg (2019) and the stateLogic machine (Diapoulis and Zannos 2012). These were chosen because they share distinct characteristics of gestural interactions. These two cases, along with Al-jazzari and Betablocker by Griffiths (2007) and iMac Music by Reus (2012), belong to the category of bottom-up live coding systems.

CodeKlavier CKalcuλator (Table 1) is a performance system that recognises the musical patterns of the pianist-coder. The extracted MIDI patterns correspond to gestural sequences that the pianist performs to write lambda functions, a common construct in functional programming, on-the-fly. The recognition algorithm identifies repetitive melodic patterns and then instructs the lambda functions to perform the corresponding code evaluations. The system affords simple operations, such as addition and multiplication, and anecdotal evidence from the pianist-coder suggests that it is a highly challenging approach to live code, as the mind is split into two tasks simultaneously. The first task corresponds to serial skilled actions resulting in the musical outcome, and the second task is to perform conscious operations to write and modify a running program. Playing the piano indeed involves pre-reflective processes. Then, on the notational level of the generated code, the piano-coder has to allocate cognitive resources that may hinder expressivity during the performance. This approach showcases how musical gestures can be employed to write computer programs. In practice, the pianist plays repetitive melodic patterns and incorporates them into musical improvisation.

When developing the approach to live hardware coding (Diapoulis and Zannos 2012, 2014), the initial motivation was to make live coding somewhat more *humane*. At the time, as a student of material science, I naively imagined material crystal structures that could be
live-coded based on a notation analogous to Miller indices, a notation system used in crystallography. This notation system offers a three-dimensional spatial representation of crystal structures in a binary-like notation. In this manner, elementary building blocks of material structures can be abstracted so that the engineer can imagine larger-scale crystal structures. As a result, what we literally see as a grain of salt can be reduced to a minimal expression, and from the minimal expression, we can deduce some of the material properties such as stiffness and conductance. The stateLogic machine (Table 1) is an experimental interface still in development (Diapoulis et al. 2022) and demonstrates the generation of a formal language from the bottom up. Initially, three input buttons were used, which depended on a clock-based paradigm. In this way, the user could apply stream-based updates within less than 0.5 seconds. These short-duration updates can enable pre-reflective processes. It is possible to automate certain pattern predictions without expending mental effort, such as looping into even and odd numbers. However, the clock-based interaction distorts the perceived immediacy of the interface, especially when the updates are larger than 100 milliseconds (Nash 2012). In principle, Al-jazzari, Betablocker and iMac Music are also clock-based systems. Nevertheless, there are different embodied percepts when the clock rate is fast enough, transforming the interaction into phenomenally continuous percepts.

Technically speaking, these systems may not influence the language development process but they do affect the algorithmic structure of the written programs. We gesturally intervene to modify the algorithm's very workings and there is clearly a minimal distance between the two. For instance, in Al-jazzari and Betablocker, the coder may experiment with multiple instruction sets or update them on-the-fly. Such dynamic updates on the level of an instruction set can account for developing a

[7]By weak temporality I refer to the typing gestures during live coding, which lack any sense of strong temporal couplings.

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press

language on-the-fly. This is because all studied systems construct the running program by on-the-fly assemblages of low-level languages. In iMac Music, Reus (2012) is taking a radical stance and rewires the internal hardware components of a personal computer (Table 1). This approach goes one step further and addresses live coding by intervening in the wirings of the hardware. How would live coding look when we can hot-swap modular hardware components? What would previewing algorithms look like? Would the system re-arrange its constituent structure to make a new component? I find a large range of possibilities for human–material interactions (Ishii et al. 2012) in live coding, and it is fascinating that we have not yet scratched the surface of this topic.

5.3. Mixed systems

All studied performances in the mixed category utilise direct manipulation with the mouse to some extent. The most diverse performance is that by The Duchess of Turing, where various systems are used, some of which are non-conventional, and one that can be regarded as a bottom-up system. Specifically, the TOPLAP app (Collins 2015) uses machine listening algorithms to program an instruction set. I present this performance, called Using various, because of the broad variety of systems used. All other performances use a single programming system and direct manipulation. The performance by uiae and redFrik share many similarities, but the difference between textual and visual language is a major difference. Threnoscope by Magnusson presents an important contribution to notation systems while Approximate Programming by Kiefer presents a sophisticated algorithm for on-the-fly adjustments. I selected The Duchess of Turing performance to question whether we can actually form a mental model of the running program using listening and all other systems because of the diverse direct manipulation interactions, ranging from simple parameter adjustments to complicated tree algorithms.

What happens when the coder is using a mixture of different systems? How can the performer form a mental model of the running program(s) by simply listening to the musical outcome? These questions are well illustrated in a live coding performance by The Duchess of Turing (Table 1) for the fifteenth anniversary of TOPLAP. During the performance, the composer-programmer uses a variety of systems, such as PureData (PD), MAX/MSP, CSound and Scheme, and various systems developed by Nick Collins, such as BBCut, Autom8 and TOPLAP app. The live coder switches between systems to generate sound patterns, creating a mashup of live coding systems that resembles DJ-ing or bricolage of live coding systems (McLean and Wiggins 2010a). The software systems are not interconnected; it is rather the musical outcome that connects everything together.

Another interpretation of this performance is that the language design and programming structures no longer matter when rendered to sound, as auditory perception can compensate and perceptually 'bind' everything together. Would that be somehow opposed to the view about the idiomaticity of low-level tools, as presented in McPherson and Tahıroğlu (2020)? Idiomaticity denotes that the design of a tool, regardless of how open-ended it is, carries design constraints that can distinctly shape the musical outcome. This discussion opens questions about agency. The notion of idiomaticity of programming languages looks aligned with a theoretical view of how influential agency progresses from all parties involved, to the spectator of the artwork by creating feedback networks (Dahlstedt 2021). The performance in question appears to suggest that when information is materialised, it loses its affiliation.

One engaging live coding performance by uiae, Using PD (Table 1), demonstrates how gestural interaction with the mouse modifies the system's algorithmic structure. Interestingly, as the different PD objects move around the programming panel, they modify their connectivity, likely from the relative distance between nodes. Fredrik Olofsson (Table 1, superCollider) also demonstrated a similar approach, where direct manipulation using the mouse modified the routing of the audio busses. Another system that uses direct manipulation is the Threnoscope by Thor Magnusson (2014b), which uses a generative system combining graphic scores and text-based programming. Using Threnoscope (Table 1), the coder can continuously control the parameters of the generated drone sounds, using interactive visuals that display a descriptive notation shown as coloured rings.

Chris Kiefer (2015) presented an approach based on a gestural controller called Approximate Programming (Table 1). This is based on genetic algorithms and the exploration of parameter spaces for machine musicianship. Approximate programming is an interactive programming paradigm where the user controls the generation of different unit generators, using a multiparameter gestural controller. The major contribution of Kiefer's work is the dynamic modification of algorithmic tree structures of interconnected unit generators. Such an approach to live coding demonstrates how gestural interactions can be applied to modify the algorithmic structure of a binary tree.

While it can be difficult to distinguish between algorithmic and parameter modifications, I would like to discuss this from a perspective that might clarify how I use the two terms. For a programmer, a variable is a powerful abstraction. In programming

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press

environments such as SuperCollider (McCartney 2002), a variable can be a generic box that holds different things. When a system is using variables as numerical parameters, such as a filter's cutoff frequency, then most likely, the generated sound patterns are predictable. If we use a variable to apply simple mappings between a control parameter and a range of numerical values, then we do not exploit the affordances of a variable. Of course, in some cases, simple 1-to-1 mappings are necessary and desirable; for instance, we may want a single knob to control the tempo and nothing more than that. Still, I draw attention to this here to clarify that gestural interactions using the mouse or other input devices may correspond to either simple modifications of a numerical parameter or complicated algorithmic modifications such as structural rearrangements of binary trees (e.g., Kiefer 2015).

This discussion on intervening with gestures on the algorithm itself is not related to *gesture-sound mappings* in the sense of running an algorithm to identify some gestural characteristics and render plausible sounds. I am interested in how we gesturally modify or re-program the algorithms on-the-fly. So, to avoid confusion with gesture-sound mappings, I here focus on *gesture-algorithm mappings*, which share some similarities with dynamic mapping strategies (Dahlstedt 2009; Kiefer 2015).

5.4. Summary of the observations

In this section, I have discussed how meaningful musical gestures can be produced using typing during canonical live coding. Inspired by Baalman's typing automaticity, I question whether gestures can influence the programming language. The answer becomes apparent when I examine the bottom-up systems, where bodily gestures can operate on pre-reflective processes and influence the development of the programming language. An online performance that creates mashups from various programming languages may indicate the importance of sound, not the tool. It is similar to when we have a phase transition in a material (e.g., water crystallises to ice), but the process is not reversible anymore, meaning that when the code is rendered to sound, the process is no longer reversible. Maybe this performance suggests that when the tool is ignorant of its consequences to the environment, it loses its agency.

Several performance systems incorporating direct manipulation have been examined, and I underline the broad spectrum of interactions, from simple manipulations of numbers to sophisticated algorithms operating on binary trees. These systems differ from canonical systems as they offer the user an intuitive manner for interactions. In cases such as Approximate Programming, the live coder can modify the algorithmic structure on-the-fly, using continuous gestural interactions on multiparameter controllers. This also exhibits pre-reflective activations during a performance. On the other hand, such systems may not allow code changes to be monitored. When this happens, we lose the possibility of involuntary imageries due to notational audiation. Thus, there is a fine line between intuitive control and how to monitor informative notations. For instance, in Threnoscope, the live coder can monitor the descriptive notation as interactive visualisations of coloured rings and can continuously control the parametric changes. Thus, if the system's temporal updates on the prescriptive notation are too fast, causing them to become uninformative, we can facilitate the use of descriptive notations instead (Magnusson 2019).

6. CONCLUSIONS

This article has explored whether pre-reflective processes can be activated in live coding. I presented an embodied cognition view to live coding music performance by discussing how notation, while hindering our bodily relationship with the generated sound patterns, also offers some opportunities. A first indication comes from the literature review, where the potential of notational audiation is posited as valuable to live coders due to involuntary musical imagery. Typing on a keyboard is an activity that incorporates serial skilled actions. While this shares similarities with traditional music performance because they are both based on serial skilled actions, typing is performed out of time order, making it difficult to observe entrained processes in canonical live coding. On the other hand, two cases of bottom-up live coding systems (CodeKlavier CKalcuλator and stateLogic machine) exhibit gestural interactions on timeframes that are capable of engaging pre-reflective processes during a performance. Several systems that use direct manipulation are examined, and at least one (Approximate Programming) also exhibits fast processes, although it is questionable whether the user can exploit the prescriptive notation during a performance. Owing to design constraints, using a descriptive notation can be an alternative so that involuntary imageries may be activated due to notational audiation. The study provides a first account of exploring fast processes in live coding by conducting observations from online videos and combining diverse literature, drawing on music psychology and music perception studies. The study is limited to solos, which could potentially constrain the results. But I intentionally limited the study to musician–system interactions and not musician–musician interactions. Thus, some essential parts of the performer's interactivity variations are

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press

identified, such as the temporal unfolding of bodily gestures during a performance and how similar or different this might be from traditional instrumental performance. Future studies could focus on intersubjective experiences in live coding and combine online video observations with interview studies and questionnaires to provide a more holistic view of how practitioners and the community experience fast processes in musical live coding.

This study contributes to the understanding of live coding, focusing on the gestural and psychological aspects of the practice, which have not been extensively studied before. This benefits both the practitioners, to increase understanding of the practice from within the community, and the general electronic music community, which may not be familiar with the particular conditions and practices of live coding. The methodology and theoretical background can be helpful to other under-represented groups practising generative music and autonomous music systems. As such, it may be used as a methodological template in how to conduct observational research from online videos, especially when there is little theoretical background in the field.

Acknowledgements

I thank Palle Dahlstedt for proofreading and supervision. I am also thankful to the anonymous reviewers and guest editors for making this article better.

REFERENCES

American Psychological Association. N.d. Mental Model. *APA Dictionary of Psychology*. https://dictionary.apa.org/mental-model (accessed 23 January 2023).

Baalman, M. 2009. Code LiveCode Live, or Livecode Embodied. [Performance]. *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME, 329.

Baalman, M. 2015. Embodiment of Code. *Proceedings of the First International Conference on Live Coding*. Leeds: ICLC, 35–40.

Baalman, M. n.d. Code LiveCode Live. https://marijebaalman.eu/projects/code-livecode-live.html (accessed 14 September 2022).

Baily, J. 1999. Ethnomusicological Perspective: Response to Sawyer's 'Improvised Conversations'. *Psychology of Music* **27**(2): 208–11.

Bishop, L., Bailes, F. and Dean, R. T. 2012. Musical Imagery and the Planning of Dynamics and Articulation during Performance. *Music Perception: An Interdisciplinary Journal* **31**(2): 97–117.

Blackwell, A. F. and Collins, N. 2005. The Programming Language as a Musical Instrument. *Proceedings of 17th Psychology of Programming Interest Group*, University of Sussex, 120–30.

Blackwell, A. F., Cocker, E., Cox, G., McLean, A. and Magnusson, T. 2022. *Live Coding: A User's Manual*. Cambridge, MA: MIT Press.

Bovermann, T. and Griffiths, D. 2014. Computation as Material in Live Coding. *Computer Music Journal* **38**(1): 40–53.

Brattico, E., Brattico, P. and Jacobsen, T. 2009. The Origins of the Aesthetic Enjoyment of Music—A Review of the Literature. *Musicae Scientiae* **13**(2_suppl): 15–39.

Brodsky, W., Kessler, Y., Rubinstein, B. S., Ginsborg, J. and Henik, A. 2008. The Mental Representation of Music Notation: Notational Audiation. *Journal of Experimental Psychology: Human Perception and Performance* **34**(2): 427.

Clayton, M., Jakubowski, K., Eerola, T., Keller, P. E., Camurri, A., Volpe, G. and Alborno, P. 2020. Interpersonal Entrainment in Music Performance: Theory, Method, and Model. *Music Perception: An Interdisciplinary Journal* **38**(2): 136–94.

Collins, N. 2015. Live Coding and Machine Listening. *Proceedings of the First International Conference on Live Coding*. Leeds: ICLC, 4–11.

Collins, N. and McLean, A. 2014. Algorave: Live Performance of Algorithmic Electronic Dance Music. *Proceedings of the International Conference on New Interfaces for Musical Expression*. London: NIME: 355–8.

Collins, N., McLean, A., Rohrhuber, J. and Ward, A. 2003. Live Coding in Laptop Performance. *Organised Sound* **8**(3): 321–30.

Dahlstedt, P. 2009. Dynamic Mapping Strategies for Expressive Synthesis Performance and Improvisation. *International Symposium on Computer Music Modeling and Retrieval*. Berlin: Springer, 227–42.

Dahlstedt, P. 2012. Between Material and Ideas: A Process-Based Spatial Model of Artistic Creativity. In J. McCormack and M. d'Inverno (eds.), *Computers and Creativity*. Berlin: Springer, 205–33.

Dahlstedt, P. 2018. Action and Perception: Embodying Algorithms and the Extended Mind. In A. McLean and R. Dean (eds.) *The Oxford Handbook of Algorithmic Music*. Oxford: Oxford University Press, 41–65.

Dahlstedt, P. 2021. Musicking with Algorithms: Thoughts on Artificial Intelligence, Creativity, and Agency. In E. R. Miranda (ed.) *Handbook of Artificial Intelligence for Music*. Cham: Springer, 873–914.

Diapoulis, G. and Dahlstedt, P. 2021a. An Analytical Framework for Musical Live Coding Systems Based on Gestural Interactions in Performance Practices. *Proceedings of the International Conference on Live Coding*. Valdivia, Chile: ICLC.

Diapoulis, G. and Dahlstedt, P. 2021b. The Creative Act of Live Coding Practice in Music Performance. *Proceedings of the 32nd Psychology of Programming Interest Group*, York, UK.

Diapoulis, G. and Zannos, I. 2012. A Minimal Interface for Live Hardware Coding. *Live Interfaces*, ICSRiM, University of Leeds.

Diapoulis, G. and Zannos, I. 2014. Tangibility and low-Level Live Coding. *International Computer Music Conference*. Athens: ICMC&SMC.

Diapoulis, G., Zannos, I., Tatar, K. and Dahlstedt, P. 2022. Bottom-Up Live Coding: Analysis of Continuous Interactions towards Predicting Programming Behaviours. *Proceedings of the International Conference on New Interfaces for Musical Expression*. Auckland: NIME.

Emmerson, S. 2017. *Living Electronic Music*. Abingdon: Routledge.

Godøy, R. I. 2004. Gestural Imagery in the Service of Musical Imagery. *International Gesture Workshop*. Berlin, Heidelberg: Springer.

Godøy, R. I. 2021. Constraint-Based Sound-Motion Objects in Music Performance. *Frontiers in Psychology*, **12**. https://doi.org/10.3389/fpsyg.2021.732729.

Goldman, A. 2019. Live Coding Helps to Distinguish between Embodied and Propositional Improvisation. *Journal of New Music Research* **48**(3): 281–93.

Griffiths, D. 2006. Betablocker. *General Public Licence (GPL)*. www.pawfal.org/flotsam/betablocker/betablocker.scm (accessed 9 May 2023).

Griffiths, D. 2007. Game Pad Live Coding Performance. *Die Welt als virtuelles Environment*. Dresden: TMA Helleraue.

Haga, E. 2008. Correspondences between Music and Body Movement. Doctoral dissertation, University of Oslo.

Haworth, C. 2018. Technology, Creativity, and the Social in Algorithmic Music. In A. McLean and R. Dean (eds.) *The Oxford Handbook of Algorithmic Music*. Oxford: Oxford University Press, 557–81.

Hutchins, C. C. 2015. Live Patch/Live Code. *Proceedings of the First International Conference on Live Coding*. Leeds: ICLC, 147–51.

Ishii, H., Lakatos, D., Bonanni, L. and Labrune, J. B. 2012. Radical Atoms: Beyond Tangible Bits, toward Transformable Materials. *interactions* **19**(1): 38–51.

Jakubowski, K. 2020. Musical Imagery. In A. Abraham (ed.) *The Cambridge Handbook of the Imagination*. Cambridge: Cambridge University Press, 187–206.

Jensenius, A. R. 2022. *Sound Actions: Conceptualizing Musical Instruments*. Cambridge, MA: MIT Press.

Jensenius, A. R., Wanderley, M. M., Godøy, R. I. and Leman, M. 2010. Musical Gestures: Concepts and Methods in Research. In R. I. Godøy and M. Leman (eds.) *Musical Gestures: Sound, Movement, and Meaning*. New York: Routledge, 12–35.

Keller, P. E. 2008. Joint Action in Music Performance. In F. Morganti, A. Carassa and G. Riva (eds.) *Enacting Intersubjectivity: A Cognitive and Social Perspective on the Study of Interactions*. Amsterdam: IOS Press, 205–21.

Keller, P. E. 2012. Mental Imagery in Music Performance: Underlying Mechanisms and Potential Benefits. *Annals of the New York Academy of Sciences* **1252**(1): 206–13.

Keller, P. E. and Appel, M. (2010). Individual Differences, Auditory Imagery, and the Coordination of Body Movements and Sounds in Musical Ensembles. *Music Perception* **28**(1): 27–46.

Kiefer, C. 2015. Approximate Programming: Coding through Gesture and Numerical Processes. *Proceedings of the First International Conference on Live Coding*, ICSRiM, University of Leeds.

Kosslyn, S. M. 1996. *Image and Brain: The Resolution of the Imagery Debate*. Cambridge, MA: MIT Press.

Leman, M. 2007. *Embodied Music Cognition and Mediation Technology*. Cambridge, MA: MIT Press.

Leman, M. 2016. *The Expressive Moment: How Interaction (with Music) Shapes Human Empowerment*. Cambridge, MA: MIT Press.

Leman, M. and Godøy, R. I. 2010. Why Study Musical Gestures?. In R. I. Godøy and M. Leman (eds.) *Musical Gestures: Sound, Movement, and Meaning*. New York: Routledge, 3–11.

Loveday, C., Woy, A. and Conway, M. A. 2020. The Self-Defining Period in Autobiographical Memory: Evidence from a Long-Running Radio Show. *Quarterly Journal of Experimental Psychology* **73**(11): 1969–76.

Magnusson, T. 2011. The ixi lang: A supercollider Parasite for Live Coding. *Proceedings of the International Computer Music Conference*. Huddersfield: ICMC.

Magnusson, T. 2014a. Herding Cats: Observing Live Coding in the Wild. *Computer Music Journal* **38**(1): 8–16.

Magnusson, T. 2014b. Improvising with the Threnoscope: Integrating Code, Hardware, GUI, Network, and Graphic Scores. *Proceedings of the International Conference on New Interfaces for Musical Expression*. London: NIME, 19–22.

Magnusson, T. 2019. *Sonic Writing: Technologies of Material, Symbolic, and Signal Inscriptions*. London: Bloomsbury Academic.

Marklin, R. W. and Simoneau, G. G. 2004. Design Features of Alternative Computer Keyboards: A Review of Experimental Data. *Journal of Orthopaedic & Sports Physical Therapy* **34**(10): 638–49.

McCartney, J. 2002. Rethinking the Computer Music Language: SuperCollider. *Computer Music Journal* **26**(4): 61–8.

McKechnie, L. E. F. 2008. *Unstructured Observation*. In Lisa M. Given (ed.), *The Sage encyclopedia of Qualitative Research Methods*. Thousand Oaks, CA: Sage, 907–8.

McLean, A. 2014. *Stress and Cognitive Load. Collaboration and Learning through Live Coding. Report from Dagstuhl Seminar*, **13382**: 145–6.

McLean, A. and Wiggins, G. A. 2010a. Bricolage Programming in the Creative Arts. *Proceedings of the 22nd Psychology of Programming Interest Group*, Madrid.

McLean, A. and Wiggins, G. 2010b. Tidal–Pattern Language for the Live Coding of Music. *Proceedings of the 7th Sound and Music Computing Conference*, SMC, 331–4.

McLean, A. and Wiggins, G. A. 2011. Texture: Visual Notation for Live Coding of Pattern. *Proceedings of* the *2011 International Computer Music Conference*, Huddersfield.

McPherson, M. J. and Limb, C. J. 2019. Improvisation: Experimental Considerations, Results, and Future Directions. In Rentfrow, P. J. and D. J. Levitin (eds.) *Foundations in Music Psychology*. Cambridge, MA: MIT Press.

McPherson, A. and Tahıroğlu, K. 2020. Idiomatic Patterns and Aesthetic Influence in Computer Music Languages. *Organised Sound* **25**(1): 53–63.

Myers, B. 1994. Challenges of HCI Design and Implementation. *Interactions* **1**(1): 73–83.

Nash, C. 2012. Supporting Virtuosity and Flow in Computer Music. Doctoral dissertation, University of Cambridge.
Nilson, C. 2007. Live Coding Practice. *Proceedings of the 7th International Conference on New Interfaces for Musical Expression*. New York: NIME: 112–17.
Nilson, C. 2016. *Collected Rewritings: Live Coding Thoughts, 1968–2015*. Burntwood, UK: Verbose. https://composerprogrammer.com/research/collectedrewritings.pdf
Noriega, F. I. and Veinberg, A. 2019. The Sound of Lambda. *Proceedings of the 7th ACM Sigplan International Workshop on Functional Art, Music, Modeling, and Design*. Berlin: FARM, 56–60.
Palmer, C. 1997. Music Performance. *Annual Review of Psychology* **48**(1): 115–38.
Palmer, C. 2012. 10 Music Performance: Movement and Coordination. In D. Deutsch (ed.) *The Psychology of Music*, 3rd edn. London: Elsevier, 405–22.
Repp, B. H. and Su, Y. H. 2013. Sensorimotor Synchronization: A Review of recent Research (2006–2012). *Psychonomic Bulletin & Review* **20**(3): 403–52.
Reus, J. 2012. iMac Music. https://web.archive.org/web/20161027164504/https://jonathanreus.com/portfolio/imac-music/ (accessed 8 April 2023).
Roberts, C. and Kuchera-Morin, J. 2012. Gibber: Live Coding Audio in the Browser. *Proceedings of the International Computer Music Conference*. Ljubljana: ICMC.
Roberts, C. and Wakefield, G. 2018. Tensions and Techniques in Live Coding Performance. In A. McLean and R. Dean (eds.) *The Oxford Handbook of Algorithmic Music*. Oxford: Oxford University Press, 293–317.
Rohrhuber, J., de Campo, A., Wieser, R., Van Kampen, J. K., Ho, E. and Hölzl, H. 2007. Purloined Letters and Distributed Persons. *Music in the Global Village Conference*, Budapest.
Roos, F. and McLean A. 2023. Strudel: Live Coding Patterns on the Web. *Proceedings of the International Conference on Live Coding*. Utrecht: ICLC.
Salazar, S. and Armitage, J. 2018. Re-engaging the Body and Gesture in Musical Live Coding. *Proceedings of the International Conference on New Interfaces for Musical Expression*. Blacksburg, VA: NIME.
Sayer, T. 2015. Cognition and Improvisation: Some Implications for Live Coding. *Proceedings of the International Conference on Live Coding*. Leeds: ICLC.
Shapiro, L. and S. Spaulding 2021. Embodied Cognition. In E. N. Zalta (ed.) *The Stanford Encyclopedia of Philosophy*, Winter edn. https://plato.stanford.edu/archives/win2021/entries/embodied-cognition/ (accessed 18 January 2023).
Shepard, R. N. and Metzler, J. 1971. Mental Rotation of Three-Dimensional Objects. *Science* **171**(3972): 701–3.
Snee, H. 2013. Making Ethical Decisions in an Online Context: Reflections on Using Blogs to Explore Narratives of Experience. *Methodological Innovations Online* **8**(2): 52–67.
Tanimoto, S. L. 2015. Livesolving: Enabling Collaborative Problem Solvers to Perform at Full Capacity. *Proceedings of the International Conference on Live Coding*. Leeds: ICLC.
Theodosopoulou Bourlogianni, D. 2021. Music and Autobiographical Memory: How an Analysis of Desert Island Discs May Help Conceptualise Personalised Music Interventions for People Living with Dementia. Doctoral dissertation, University of East Anglia.
Varela, F. J., Thompson, E. and Rosch, E. 2017. *The Embodied Mind, Revised Edition: Cognitive Science and Human Experience*. Cambridge, MA: MIT Press.
Zatorre, R. J. and Halpern, A. R. 2005. Mental Concerts: Musical Imagery and Auditory Cortex. *Neuron* **47**(1): 9–12.

VIDEOGRAPHY

McCallum, L. 2011. *Show Us Your Screens. Vimeo*, February 22. https://vimeo.com/20241649 (accessed 14 April 2023).
Villaseñor, H. 2019. Dar forma al espacio: Primer Festival Expresiones Contemporáneas. YouTube, 10 November. https://youtu.be/vARqRuMoPx8 (accessed 14 April 2023).

https://doi.org/10.1017/S1355771823000444 Published online by Cambridge University Press