The creative act of live coding practice in music performance

Georgios Diapoulis
Interaction Design
University of Gothenburg,
Chalmers University of Technology
geodia@chalmers.se

Palle Dahlstedt
Interaction Design
University of Gothenburg,
Chalmers University of Technology
palle@chalmers.se

Abstract

Live coding is the creative act of interactive code evaluations and online multimodal assessments. In the context of music performance, novel code evaluations are becoming part of the running program and are interrelated to acoustic sounds. Performers' and audience ability to experience these novel auditory percepts may involuntary engage our attention. In this study, we discuss how live coding is related to auditory and motor perception and how gestural interactions may influence musical algorithmic structures. Furthermore, we examine how musical live coding practices may bring forth emergent qualities of musical gestures on potentially equivalent systems. The main contribution of this study is a preliminary conceptual framework for evaluation of live coding systems. We discuss several live coding systems which exhibit broad variations on the proposed dimensional framework and two cases which go beyond the expressive capacity of the framework.

1. Introduction

1.1. Live coding for the composer-programmer

Live coding practice is a well spread performance activity among computer musicians. Since the funding act of the Temporary Organisation for the Promotion of Live Algorithm Programming, which begun with its draft manifesto (TOPLAP, 2005), a community of few live coders has now been seeing a tremendous expansion. In fact, the term live coding seems to be unclear within academic circles. Many believe that the term corresponds to streaming tutorials where professional programmers show best practices on how to "code live". While this may reflect some aspects of live coding, it does not manifest the potential of a novel computing platform.

Live coding in music performance is a multimodal endeavour. Audition, vision, touch and balance are all forming a closely knit whole during performance. All aforementioned sensory cues may engage both the composer-programmer and audience in a multimodal experience. During performance practice the live coder is typically sharing her screen with the audience. This makes the process of live coding a transparent performance act, in which failure is always a possible outcome. In this manner, both the coder and the audience incorporate the generated music as a proxy to form a mental model of the running program. Consequently, the live coder aims to modify the running program on-the-fly, so that novel auditory percepts may involuntary engage our attentional resources (Escera, Alho, Winkler, & Näätänen, 1998). Such novel performance acts that are realized within the context of "show us your screens", are widely used in algorave parties, where the audience is sometimes dancing during the concert (Collins & McLean, 2014). If the live coder fails to evaluate successfully the current code chunk and commit a system crash, she might start all over once again, or if she feels exhausted she might seek for some encouragement from the audience. This gestural communication between the audience and the live coders is well established in live performances as the algoraves are about to become 10 years old in 2022.

1.2. Humans in the loop

Live coding is a novel performance practice and maybe extends the notion of human-centric computing. This is because the human participant has an active role which is determined by the social nature of musical activities (Collins, McLean, Rohrhuber, & Ward, 2003; Thompson, 2015). On the other hand, there are still difficulties how to humanly embody our interaction with algorithms. Musicians are encountered to a first-hand experience of the semantic gap, as this is portrayed between the generated music and the

typed code expressions. The most promiment way, to date, to experience embodied interactions in live coding is taking flesh during a dance performance (McLean & Sicchio, 2014). Even in this scenario, the dancer is carrying on the performance within the predominant absence of causal or direct auditory percepts linked to musicians' gestures. Besides these drawbacks, the world of live coders is a lively and widely divergent community of hackers, expert and novice programmers, academics, professional and hobby musicians and most likely a bunch of retired enthusiasts within the next decades (Nilson, 2007). Furthermore, there is a broad range of computer music conferences that have incorporated live coding as a research topic, but most importantly there is a quasi-annual conference on live coding (ICLC), first appeared in 2015.

1.3. Outlining the purpose of the study

In this study, our purpose is to present a conceptual framework for evaluation of live coding music systems. There is a broad variation of systems and practices among live coders and to the best of our knowledge there is no study to date which examines how music systems may differ to each other. The swedish alter ego of Nick Collins has reflected on different practices and actually proposed a battery of live coding exercises (Nilson, 2007). That was a month long live coding exercitiae carried out and documented with Fredrik Olofsson. In the next section we review literature from music psychology and perception along with studies in live coding and human-computer interaction. The focal point is how gestural interactions are employed in musical interaction design. Following that, we present a preliminary conceptual framework which aims to evaluate live coding systems. Finally, we discuss how such frameworks may flourish the development of novel music systems and we reflect on the possibility of a parallel evolution of performance practices.

2. Live coding: musical activities, music performance, systems and practices

2.1. Musical activities

Musical activities may be divided into three categories: music-making, music listening and musical imagery (see Figure 1). Music-making involves both music composition and music performance. Music listening is the most widespread activity as we are exposed to music many hours per day, even involuntary when drinking a coffee in a coffee shop. Musical imagery is the activity of imagining a melody of a song, a musical gesture and so on. A typical case of involuntary musical imagery are the so-called earworms, which is when a melody is in a person's mind.

Here, musical activities are presented as progressively overlapping categories. We may claim that there is also a progressive engagement in musical activities, starting from the least engagement in musical imagery, following to more engagement during music listening and even more engagement during music-making (Luck, 2015). In that manner, during performance the live coder is employing music percepts towards building progressive levels of engagement. When an audience is attending a concert then the dynamics between audience and musicians is also an engaging experience, where dancing and gestural communication are typically of major importance.

2.2. Traditional and live coding music performance

During a concert, the generated music is heard by both performer and audience, and intersubjective music preferences may vary considerably. Whereas a live coding performance incorporates both visual and auditory percepts, to the best of our knowledge, there are no studies which assess how the visual channel contributes to audiences' appreciation. For instance, in traditional music performance is well established that exaggerated bodily movement biases our visual perception of expressivity, which in return contributes to audience appreciation (Davidson, 1993).

Contrary, in live coding the bodily movement is usually minimal. This is an issue which the live coders have to consider if they would like to tighten the engagement with audiences and co-performers. Although, the visual projections have an important role in the live experience as a whole, it is difficult to make educated assessments due to the broad variety of live coding systems. Thus, in our study we consider only the auditory percepts during performance. Here, music listening is seen as an activity which is linked to music preferences and appreciation of the generated music. The live coder is appreciating

the generated music on-the-fly and this may result to structured code evaluations, which are tested and work properly (see Figure 1). In addition, musical imagery is linked to anticipated percepts which occur when people are exposed to music-like stimuli. During music performance this contributes to planning and involves a sequence of bodily movements which are required when playing a musical instrument (Keller & Koch, 2008). Similarly, in live coding performance, the coder is planning her future actions by trial-and-error of novel code evaluations (Tanimoto, 2017). In that manner, the coder is anticipating the auditory percepts of her actions.

The difference between live coding and traditional music performance is that in live coding the learned associations are not necessarily linked to automaticity in gestural control. Instead, the coder is making progressive levels of abstraction, which may be automated to a certain extent (Nilson, 2007). Automaticity in sensorimotor control, especially in the case of typing, is being unfolded in a later stage when the planning has brought forth some sort of mental model of the novel code structures. Such practices are linked to novelty and creativity, which are interrelated concepts to some extent. Typically, creativity depends on some novelty-related component. When there is mismatch of expectations during a live coding performance, this may result to either failure of execution and possibly a system crash or to novel music percepts.

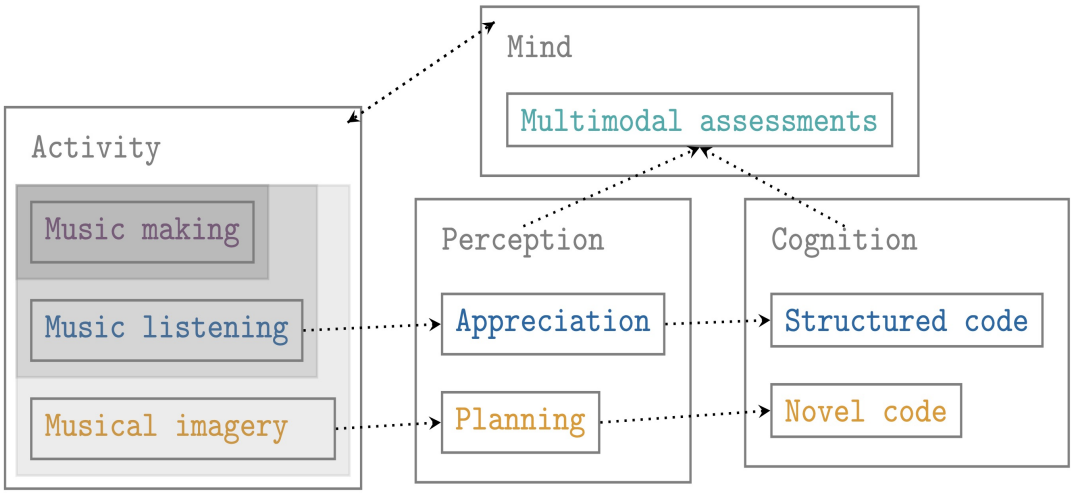

Figure 1 – Musical activities in live coding performance. Description of perception and cognition during performance.

2.3. Perceptual and cognitive aspects of live coding systems and practices

2.3.1. Motor skills in musical interface design

From a motor perspective, a system's response time cannot be smaller to the slowest part of the system (Gibet, 2010). If we transfer this from the motor domain to the user interface design, then we may conclude that the only case of intimate gestural feedback with a user interface (UI) can be achieved based on direct manipulation. Moreover, it is reasonable to assume that in many cases the user's understanding is facilitated when no algorithm is involved during gestural interactions. Contrary, the systems that are used in live coding performances and new interfaces for musical expression (NIMEs), may require a long sequence of gestural interactions which involve algorithmic complexity. The question arises, how algorithmic complexity may be linked to musical gestures as these are portrayed in traditional music performance (Jensenius, Wanderley, Godøy, & Leman, 2010). Do we have to expand the notion of musical gestures (Salazar, 2017)? This can be a plausible argument because musical interfaces share properties from both human-computer interaction (HCI) and traditional music performance. In HCI the gestures in users' interfaces are considerably different than musical gestures. In music performance there is a gestural virtuosicity which is considered a no-go in user interface design.

2.3.2. Multifaceted functionality of musical imagery

During a live coding performance the composer-programmer is making music on-the-fly by incorporating interactive code evaluations. While doing so, she is listening to the music but also imagining anticipated music percepts. The latter is known as anticipatory auditory imagery (Keller, 2012). For instance, in dance music we anticipate the bass drum to be heard on regular intervals within the musical structure. If the composer alternate these regular repetitions of the bass drum our expectations would be mismatched and novel music percepts may arise from such structural modifications. Another significant aspect of musical imagery is that during music listening, pianists have demonstrated activations of involuntary motor imagery (Haueisen & Knösche, 2001). Thus, anticipatory imagery is involved both in action planning and action execution (Keller, 2012).

Here, we stretch the importance of a sequence of gestural interactions and we question how online auditory percepts may influence such mutlilayered gestural unfoldings in live coding. We argue that there should be some sort of mental models that allow the musician to conduct on-the-fly programmatic structures that meet her musical percepts. Interestingly, expert programmers have reported imageries of gestures and other bodily movements during problem solving tasks (Petre & Blackwell, 1999). Whereas direct manipulation can indeed provide a more traditional-like sense of intimate gestural control, more complex systems may also employ gestural imagery. Godøy (2003) sees that contrary to auditory imagery, gestural imagery may be suppressed in time. That is, we can "fast-forward" musical gestures using our mind, whereas the same do not apply to sounds. One cannot compress the duration of a sound and experience similar percepts. How such "gestural compressions" may be related to goal-directed actions? Is the teleological inquiry a mechanism which can facilitate the formation mental models?

If we examine the so-called "earworms" which seem to arise out of circumstances of involuntary musical imagery (notably have been reported widely in non-musicians as well) (Williamson, 2011), then such involuntary actions may trigger the formation of mental models. Such imagery, also known as spontaneous, can be triggered by musical notation. This is known as notational audiation in literature. Live coders employ similar imagery artifacts from chunks of code, as the code is becoming a musical notation (Magnusson, 2011). For instance, the first author is employing visual imagery when performing standard live coding sessions in SuperCollider using the keyboard. This is in the form of geometrical abstractions which are typically realized using low frequency oscillators (LFOs) to manipulate melodic and rhythmic structures that are usually driven by unit generators (UGens).

Several kinds of mental imagery have been reported in both expert and novice programmers (Petre & Blackwell, 1999). For example, both expert and novice programmers reported that they employed visual imagery when structuring a program. Gestures and algorithms can be difficult to put into strict boundaries, that is to put them into segmented structures. On the other hand, auditory and visual percepts exhibit segmentation properties. For instance, a sound event may attribute segmented boundaries to gestures via the onsets and the offsets of the sound. This is how computer music is linked to bodily movement throught its temporal structure (Palmer, 1997).

2.3.3. Knit together systems and practices

So far, we have argued that a blend of effortfull and involuntary imagery takes place during performance. One instance of imagery is immidiately linked to gestural interactions (Godøy, 2003), which has temporal advantages in comparison to auditory percepts (ie. "fast-forward" gestural representations). Moreover, musical notation can function as a source of imagery-induced processes. One can think that there is a solid ground already so to engage with the study of musical gestures in live coding, but the broad variations of systems and practices bring about a plethora of gestural interactions. On top of that, we have to take into consideration some principles of human-computer interaction. For instance, standard live coding systems which are based on typing on a keyboard are known to offer terrible closeness of mapping (Blackwell & Collins, 2005). In fact, live coding systems that incorporate the keyboard for

gestural control may be seen as obscured, as typing on a keyboard is "neither observed nor significant"[1] (Jensenius et al., 2010).

Here, the contribution by Marije Baalman is more than significant (Baalman, 2009). Baalman demonstrated in her "Code LiveCode Live" session that typing on a keyboard can bring about meaning and made the transition from an unspecified and "non significant" domain to the musical gestures domain. This point is likely the very essence of this article, which is linked to the very title of this paper. That is, potentially equivalent live coding systems may bring about different performance practices. This in return, can bring forth novel contributions such as assigning meaning to "non significant" actions, like typing on a keyoard.

3. A conceptual framework for evaluation of live coding music systems based on gestural interactions

Our investigation began by examining different systems and practices in musical live coding. We reviewed half a dozen live coding systems from the viewpoint of how gestural interactions vary across different practitioners. A turning point which made us realize the importance of variations in performance practices was Marije's Baalman "Code LiveCode Live". The interesting characteristic of Marije's system is that it is potentially equivalent to a standard live coding system. Particularly, Marije used SuperCollider language which is commonly used by many live coders. The only difference between a standard live coding system based on SuperCollider and "Code LiveCode Live" is that Marije activated the built-in microphone and other sensors of the laptop while typing. In that manner she used the typing sounds on the keyboard as the raw material of the composition. This action transformed the meaning of typing in Marije's system. Typing on a keyboard cannot be seen as a non significant action neither as not observed. In contrast, typing has now become a musical gesture, which is actually a sound-producing gesture. That is, it is absolutely significant for the production of the sound. This realization, demonstrated how different practices may bring about creative acts in potentially equivalent live coding systems.

In the following section we are discussing the four main systems under investigation. Next, we present a preliminary visualization of our framework. At the end of this section, we discuss two more systems which cannot be represented using our framework. A discussion follows including potential future work and adjustments can be done to fine tune the framework.

3.1. Four systems under investigation

Here, we focus on four idiosyncratic live coding systems by emphasizing on the gestural control. These are *Al-jazzari* by Dave Griffiths, *stateLogic machine* by Diapoulis, *Code LiveCode Live* by Baalman and *CodeKlavier* by Noriega & Veinberg. The motivation was to examine cases where the musical gestures play an important role in gestural control. As such we included typical cases where the keyboard is used as the input interface, but also exotic (Diapoulis & Zannos, 2012, 2014) and metaphorical design cases, like piano performances (Tanimoto, 2017). The aforementioned variations clearly showed that music systems incorporate design metaphors (Wessel & Wright, 2002). A typical case of a design metaphor denotes a computer music system that was developed based on some existing musical instrument. For example, if we map the letters of the keyboard to a MIDI piano this would account as a metaphorical design. In contrast, literal design setups, like the standard live coding systems, are based on a keyboard which may inhibit gestural expression in comparison to playing the piano.

3.1.1. Code LiveCode Live

Marije Baalman approached the tackling issue of embodiment within laptop performance by incorpotating the clicking sounds on the keyboard into her musical composition (Baalman, 2009, 2015). In that manner, Marije accomplished direct sound to be heard during her live coding performance, as she used physical data as audio input (Nilson, 2007). That was indeed a novel contribution, although the practi-

[1] Original quote by Hulteen (1990) state that (p.310) "A gesture is a motion of the body that contains information. Waving goodbye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on it's way to hitting a key is neither observed nor significant. All that matters is which key was pressed".

calities of such dual-functionality of the keyboard as both a percussive instrument and a typing UI could not establish a "normal paradigm" for live coding music performance. In fact, Marije's apparatus was not aiming to reach this goal. Most likely her novel contribution was indicating self-referential aspects, as they unfold, during performance. If this apparatus was meant to be taken literally as a "standard" for performance, then it would have triggered a parallel co-evolution of novel keyboard setups, UIs and programming languages.

3.1.2. CodeKlavier

In the same direction the CodeKlavier system (Noriega & Veinberg, 2019), demonstrated a novel live coding performance setup by employing the clavier of the piano as input interface. The novel contribution of the "Hello world" performance [2] was that the authors literally executed a hello world program using the piano as input interface. Whereas this may sound as a "dummy" demonstration, the aim was to be a proof of concept. Below we will focus on the fourth revision of the system, also known as CKalculator[3]. If we imagine a one-dimensional space of *design metaphors* (Dahl & Wang, 2010) and *literal design* then the CodeKlavier would be on the one end of design metaphor and Baalman's approach on the other end of literal design. Moreover, the two systems differ on another dimension. Marije's design is agnostic to the significance of keypresses, whereas in CodeKlavier CKalculator design the importance of keypresses is highly significant to structure the code. By *algorithm agnostic* we mean that Marije's gestures do not have any impact on the algorithm itself. The coder is doing as many gestures as she likes, she might also do gestures without any temporal contraints and this has no effect on the algorithmic structure of the program.

3.1.3. Al-jazzari

Contrary to the previous two music systems, Dave Griffiths presented one of the very first systems which approached live coding from a low-level perspective (McLean, Griffiths, Collins, & Wiggins, 2010). Al-jazzari is building on a metaphorical design in which a computer game is used as a notation for live coding. The computational approach relies on evaluating commands from a minimalistic instruction set and the input interface is a gamepad controller. Here, every user's action has a significant impact on the algorithm. This is because a positive edge clock is registering the user's input in real-time.

3.1.4. stateLogic machine

Diapoulis and Zannos (2012, 2014) presented a low-level computational approach to deal with live coding. The users' input is provided on the lowest level of the machine, that is, the bit level. The machine is a combination of two finite state machines (FSM), a counter and a decoder, and the user interface is an automaton itself. In the revised version (Diapoulis & Zannos, 2014), the machine was able to recognize regular expressions and generated a minimal type-3 language, which enumerated seven words. Here, the design is literal as the performer is providing the input using switches. The actions of the performer are absolutely significant to the algorithm and the code precedes to the generated music.

3.2. Dimensional framework

One way to evaluate live coding systems is to rely on a multi-dimensional space. Here, we decided to constrain the proposed framework up to three dimensions. Whereas an orthogonal three-dimensional system can be misleading, we decided to employ such representation to facilitate the understanding of the reader. Our intention was to provide a comprehensible visual representation of the framework. Thus, we engaged in a process of identifying the most important semantic differentials which can reflect the variations between the systems under investigation.

Our first observation was that the *interface design* can be either *metaphorical* or *literal*. We introduce here the term "literal design" to denote that the system fulfils the requirements of a standard live coding system. That is, the interface is based on some sort of electronic components such as switches, keyboards, circuits and the like. This is the first dimension (X-axis) of the framework as shown in Figure 2.

[2] CodeKlavier - hello world (Anne Veinberg playing piano and coding at the same time!) https://youtu.be/ytpB8FB6VTU

[3] Anne Veinberg, Felipe Ignacio Noriega - The CodeKlavier CKalculator (...) | Lambda Days 2019: https://www.youtube.com/watch?v=0fL40oLU8C4

Here, we assert that the interface design should be reflected to the qualities of gestural interactions.

The second dimension of the framework examines the importance of gestural interactions on the algorithm of the system. For instance, in the case of stateLogic machine every input provided from the user modifies the algorithm of the system. To provide a more concrete example, here, we have to introduce a third dimension which has *direct manipulation* on the lower end and *algorithmic complexity* on the upper end. If we imagine a continuous gesture on a tangible interface then this is a direct manipulation gesture. The question arises, "what if there is an algorithm behind this direct manipulation gesture?" (Björk, 2021). For this reason, the second dimension of the framework clarifies whether the gesture is actually *significant* to the running algorithm or it is *agnostic* to it. Thus, the second dimension, as shown on the Y-axis, corresponds to *gestural mapping* and the third dimension, Z-axis, to *user interaction*.

The directionality of the axes was designed to represent concrete concepts on the lower end and abstract concepts on the upper end. This is a cognitive paradigm that shows a directionality from low-level concepts to high-level concepts. The three basic dimensions on the framework are shown in Table 1. The dimensions represent some of the basic processes that the live coder is engaged with during the development of a musical performance system.

Table 1 – Each axis denotes a process which is represented by semantic differentials. The directionality of the axes is composed by low-level concepts on the lower end and high-level concepts on the upper end.

	Process	Low-level (concrete)	High-level (abstract)
X-axis	*Interface design*	literal design	metaphorical design
Y-axis	*Gestural mapping*	algorithmic significance	algorithm agnostic
Z-axis	*User interaction*	direct manipulation	algorithmic complexity

Finally, as shown in Figure 2 we included a binary dimension as was proposed by Tanimoto (2017). In a live coding system, either the code preceeds the music (code-first) or the the music preceeds the code (music-first). Regarding the case of Baalman's system, we categorize it as a music-first because the typing sounds are feeded forward to the generated music. Here, we have to highlight that if the performer does not execute any commands to switch on the built-in microphone of the laptop, then no typing sounds will be heard. In that manner, the system may also be categorized as code-first. In principle, a more accurate description would be to go beyond the binary division of code-first and music-first to include more categories. In this case, Baalman's system would be a conditional music-first system.

3.3. Beyond the expressive capacities of the framework

Below we present two cases which cannot be represented with the propose framework.

3.3.1. Type-A personality

The piano composition "Type-A personality" was performed during the first international conference in live coding (Collins & Veinberg, 2015). The pianist is playing the piano but also typing on a keyboard at the same time. Indicatively, keyboard characters are shown on the score in the video of the performance. This system cannot be categorized neither as a metaphorical design nor as a literal design. Furthermore, the gestural mapping seems to be significant to the algorithm but an interview with the either the composer or the performer will shed light to it. For example, if a machine listening component performs online music analysis then the gestural interactions are significant to the algorithmic. Finally, the system seems to incorporate only direct manipulation.

3.3.2. Threnoscope

The live coding system "Threnoscope" presented a blend of visual notation coupled to a standard live coding system (Magnusson, 2014). The implementation was done in SuperCollider and the performer can use both the keyboard and the mouse for user interaction. In that manner, the system incorporates both direct manipulation and algorithmic complexity. The gestural interaction is agnostic the algorithm, although when the performer interacts with the visual notation it can adjust numerical values on different

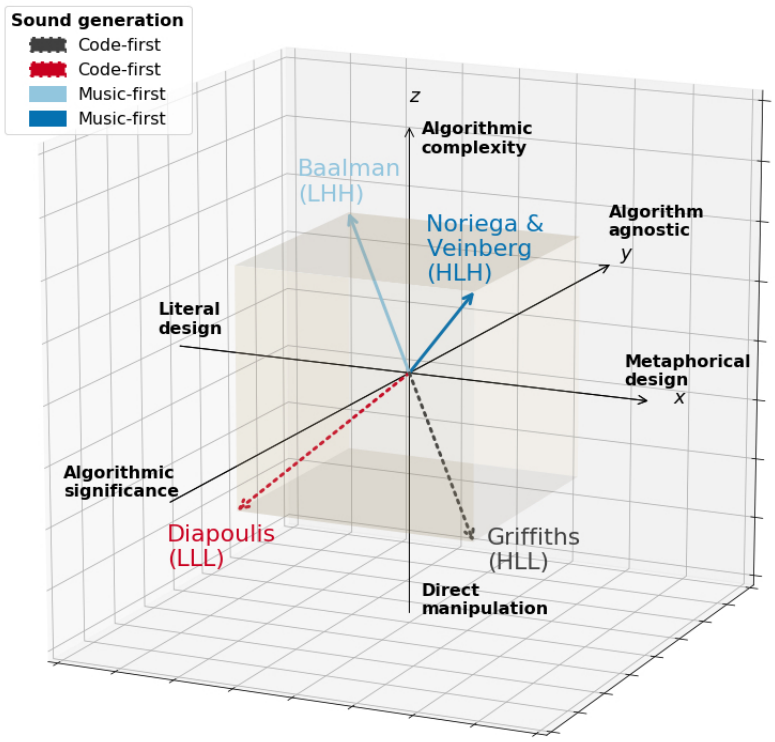

Figure 2 – Dimensional framework from the viewpoint of gestural interactions. Uppercase characters "H" and "L" correspond to high-level and low-level concepts for the triad XYZ axes, respectively. Dashed arrows show systems that the code precedes the generation of sound.

parameters.

4. Discussion

We presented a preliminary version of an evaluation framework for musical live coding systems from the viewpoint of gestural interactions. Musical gestures in traditional music performance have a long history and the musicians are well-known to be experts of sensorimotor control. A central theme in our study was to built upon a theoretical background in which the musical activities are seen as nested categories. Indicatively, music-making incorporates both music listening and musical imagery. An attempt was made to explain how gestural unfoldings may influence our mental model of the running program. This may be explained through segmented structures that are realized by auditory percepts, which in return may influence the fast-forward of gestural interactions. On the level of musical imagery, spontaneous imagery has shown to influence motor activity (Haueisen & Knösche, 2001), thus, it can be involved in action planning and execution (Keller, 2012).

A clear distinction between musical live coding systems and practices is made, to facilitate the understanding of the reader. Interestingly, we presented a case (Baalman, 2009) in which potentially equivalent systems can bring about different performance practices. Our motivation was to conceptualize how variations in performance practices may contribute to the development of novel systems. The preliminary nature of the proposed framework is expemplified by two special cases which cannot be represented in a consistent manner. Furthermore, the three-dimensional representation that was choosen for visual communication, may be misleadning for the reader as the orthogonality of the axes typically corresponds to independent concepts.

Furthermore, we introduced the dimension of gestural mapping from the viewpoint of how gestural interactions may have an effect on the running algorithm. This clarifies the reason that the direct manipulation and the algorithmic complexity are presented as semantic differential concepts.

Future studies should evaluate the validity of the framework, either using quantitative, qualitative or mixed methods. Indicatively, interview studies can be very beneficial for verifying shared conceptions among the community of live coders. How such frameworks may benefit the live coding community? We believe that by offering a conceptual framework based on the viewpoint of gestural interactions we will facilitate the development of novel performance systems. Engaging into iterative processes by practicing and making efforts to go beyond the expressive capacities of such frameworks can be only beneficial for our imagination during performance.

5. References

Baalman, M. (2009). *Code LiveCode Live.* Retrieved 2021-09-29, from `https://marijebaalman.eu/projects/code-livecode-live.html`

Baalman, M. (2015). Embodiment of code. In *Proceedings of the first international conference on live coding* (pp. 35–40).

Björk, S. (2021, September). Personal communication.

Collins, N., & McLean, A. (2014). Algorave: Live performance of algorithmic electronic dance music. In *Proceedings of the international conference on new interfaces for musical expression* (pp. 355–358).

Collins, N., McLean, A., Rohrhuber, J., & Ward, A. (2003). Live coding in laptop performance. *Organised sound*, *8*(3), 321–330.

Collins, N., & Veinberg, A. (2015). *"type a personality." a performance at iclc 2015.* Retrieved from `https://www.youtube.com/watch?v=0fX0AymCtgA`

Dahl, L., & Wang, G. (2010). Sound bounce: Physical metaphors in designing mobile music performance. In *Nime* (pp. 178–181).

Davidson, J. W. (1993). Visual perception of performance manner in the movements of solo musicians. *Psychology of music*, *21*(2), 103–113.

Diapoulis, G., & Zannos, I. (2014). Tangibility and low-level live coding. In *Icmc.*

Escera, C., Alho, K., Winkler, I., & Näätänen, R. (1998). Neural mechanisms of involuntary attention to acoustic novelty and change. *Journal of cognitive neuroscience*, *10*(5), 590–604.

Gibet, S. (2010). Sensorimotor control of sound-producing gestures. In *Musical gestures* (pp. 224–249). Routledge.

Godøy, R. I. (2003). Gestural imagery in the service of musical imagery. In *International gesture workshop* (pp. 55–62).

Haueisen, J., & Knösche, T. R. (2001). Involuntary motor activity in pianists evoked by music perception. *Journal of cognitive neuroscience*, *13*(6), 786–792.

Jensenius, A. R., Wanderley, M. M., Godøy, R. I., & Leman, M. (2010). Musical gestures: Concepts and methods in research. In *Musical gestures* (pp. 24–47). Routledge.

Keller, P. E. (2012). Mental imagery in music performance: underlying mechanisms and potential benefits. *Annals of the New York Academy of Sciences*, *1252*(1), 206–213.

Keller, P. E., & Koch, I. (2008). Action planning in sequential skills: Relations to music performance. *Quarterly Journal of Experimental Psychology*, *61*(2), 275–291.

Luck, G. (2015, September). *Lecture notes in the x-factor in music.* University of Jyväskylä, Jyväskylä, Finland.

Magnusson, T. (2011). Algorithms as scores: Coding live music. *Leonardo Music Journal*, *21*, 19–23.

Magnusson, T. (2014). Improvising with the threnoscope: Integrating code, hardware, gui, network, and graphic scores. In *Nime* (pp. 19–22).

McLean, A., Griffiths, D., Collins, N., & Wiggins, G. (2010). Visualisation of live code. *Electronic Visualisation and the Arts (EVA 2010)*, 26–30.

McLean, A., & Sicchio, K. (2014). Sound choreography<> body code. In *Proceedings of the 2nd conference on computation, communcation, aesthetics and x (xcoax)* (pp. 355–362).

Nilson, C. (2007). Live coding practice. In *Proceedings of the 7th international conference on new interfaces for musical expression* (pp. 112–117).

Noriega, F. I., & Veinberg, A. (2019). The sound of lambda. In *Proceedings of the 7th acm sigplan international workshop on functional art, music, modeling, and design* (pp. 56–60).

Palmer, C. (1997). Music performance. *Annual review of psychology*, *48*(1), 115–138.

Petre, M., & Blackwell, A. F. (1999). Mental imagery in program design and visual programming. *International Journal of Human-Computer Studies*, *51*(1), 7–30.

Salazar, S. (2017). Searching for gesture and embodiment in live coding. In *Proceedings of the international conference on live coding.*

Tanimoto, S. (2017). Challenges for livecoding via acoustic pianos. In *3rd international conference on live coding. morelia, mexico.*

Thompson, W. F. (2015). *Music, thought, and feeling: Understanding the psychology of music*. Oxford university press.

TOPLAP. (2005). *ManifestoDraft.* Retrieved 2021-09-29, from `https://toplap.org/wiki/ManifestoDraft`