CHALMERS
UNIVERSITY OF TECHNOLOGY

Livecode me: Live coding practice and multimodal experience

Livecode me: Live coding practice and multimodal experience

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

Abstract

I present a practice-led research design to explore relations between listening and non-listening conditions during a month-long live coding practice. A documentation of the live coding sessions and textual data from my daily diaries are presented in a git repository. The study offers a set of observations related to musical and programming practices, an ongoing work on a visual helper and outlines issues related to solo live coding practice.

1. Introduction

A central component of musicianship is diligent practice. Musical practice is a daily activity that musicians do, and the same applies to live coding. Musical live coding uses on-the-fly computer programs to generate sound patterns (Collins, McLean, Rohrhuber, & Ward, 2003). It can also be seen as an approach to experience composition while composing musical outcomes (Sorensen & Brown, 2007), and it is an improvisational practice. A significant difference between learning how to live code and learning how to master composition or instrumental music performance is that there is no school to learn how to live code, except a wiki page on TOPLAP website [1]. Learning by example and trial-and-error problem-solving technique is at the heart of every live coding practice. Click Nilson, a persona of Nick Collins, contributed the first study on live coding practice and proposed a set of exercises for improving coding and musical expertise (Nilson, 2007). He addressed these topics by consulting developmental and educational psychology literature studies and carried out a month-long daily practice with Fredrik Olofsson.

My perspective was to carry out daily practice sessions, having in my mind how a novice user would explore live coding. The primary motivation was to improve my musical live coding skills and explore the role of music listening during live coding practice. To explore the role of listening in the context of multimodal perception, I conducted daily sessions with listening and without listening to the generated sounds. The latter was done by muting the soundcard's audio output to the speakers. Listening to the sound while coding is an indispensable part of live coding practice (A. F. Blackwell & Collins, 2005). I posed the question, what if the live coder does not listen to the sound? Thus, I experienced whether listening to the generated sound patterns may help me to understand the written code and benefit my live coding practice.

For the present study, I carried out daily solo live coding sessions. At times it felt like a conversation with myself (Gamboa, 2022), but also a familiar thing to do as I have been practising musical instruments for many years. For the live coding sessions I used SuperCollider [2], a programming environment for sound synthesis and algorithmic composition, and documented temporally accurate representations of the coding sessions. Every day I conducted both listening and non-listening sessions, and I wrote short diary entries at the end of the day, sometimes listening the sessions afterwards by replaying the code recording.

Multimodal experience in live coding studies has been approached from an audience perspective (Burland & McLean, 2016). In this article, I focus on the subjective experience of the live coder. Contrary to Burland and McLean (2016), who discuss how audio-visual information facilitates audience

[1] https://toplap.org/wiki/Live_Coding_Practice
[2] supercollider.github.io/

aesthetic experience, my focus was how audio-visual information could be useful for the performer. Thus, instead of focusing on audience appreciation and enjoyment, I focus on the relation between the auditory and visual percepts of the live coder. How can I get informed when I do not listen to the musical outcome? For that purpose, I explored how visual information from the audio spectrum analyzer, a visual representation of the magnitude of a signal as a function of frequency, can be helpful for the performer.

This study is focused on reproducing the methodology of the seminal practice sessions by Collins and Olofsson and is also influenced by the methodological approach presented by Blackwell and Aaron (2015). It is a practice-led research approach, which can be seen as an extended concept of research through design. By focusing on the four elements by Blackwell and Aaron (i.e., identifying design exemplars, critical orientation, exploratory implementation, and reflective assessment), I contribute a code repository of the coding sessions and reflections related to musical and programming practices, multimodal perception and general issues related to the activity of practising live coding alone.

2. The practice of musical live coding

The computer can be seen as the 'natural tool' of electronic music (Nilson, 2007). During a live coding session, the code is translated to musical outcomes, and the musicians constantly listen to the generated sounds (A. F. Blackwell & Collins, 2005). Collins presented a month-long live coding practice session along with Fredrik Olofsson. The practice sessions were conducted in SuperCollider, and the documentation of the self-administered daily sessions of unaccompanied solo practice is available online (`https://swiki.hfbk-hamburg.de/MusicTechnology/819`). During this month-long practice, the two live coders had different approaches. Collins had a daily plan and exercised specific algorithmic problems, like the 3x + 1 problem, whereas Olofsson did not have a specific plan per day. The sessions were 'blank slate', meaning there was no pre-written code to be executed. The goal of both coders was to practice one hour per day. That is, one hour-long practice sessions simulating a performance setting. Collins sums up his contributions on three main aspects: (i) isolation exercises, (ii) connectivity exercises, and (iii) repertoire implications. Isolation exercises are activities that a live coder can carry out alone and do not necessarily relate to musical practice. Examples include practising fast typing or solving mathematical problems. Connectivity exercises are music-related and address issues the live coder has to confront during a live session. These connectivity exercises may include controlling musical tension, mixing audio signals, and so on. The third aspect of, what I call, repertoire implications addresses issues such as code sharing in laptop ensembles, among others.

Sorensen and Brown (2007) report five computational techniques used during live coding practice. These techniques are generic and can be extended to different aspects of electronic music, like using probability functions, periodic functions and modulo arithmetics, among others. They elaborate on the programming practices during live coding, such as code expansion, function abstractions and keyboard shortcuts. Collaboration and communication are two live coding practices that are also essential. Collaborative live coding has also been a risk management technique (Roberts & Wakefield, 2018). In my month-long practice, I did not do any collaborative sessions. On the communication aspect, I communicated my daily practices with various people, from academics to practitioners and non-musicians. However, I feel I should have been more systematic in this.

Magnusson (2015) explored notational aspects of live coding practice. Magnusson identifies a difference between prescriptive and descriptive aspects of visual notations and offers a solution to the problem of making a connection between code representations and temporal representations of musical events. Another approach to visualisation of musical forms have been recently presented (Dal Rì & Masu, 2022), either as a linear temporal evolution or as clusters showing event density per family of sounds. The authors suggest that these two visualization techniques can be complementary to each other, and reflect on issues related to attention span between coding and visual representations of musical form.

2.1. Psychological aspects of live coding practice

The present study offers a perspective in which the user's task is unlike the 'normal model' of live coding practices (A. F. Blackwell & Collins, 2005). The normal model of musical live coding involves listening to the sound while simulating a performance setting. Still, I conducted at least one session daily without listening to any sounds. Consequently, it contributes to the literature with a use case where the user's needs are unlike the needs of a live coder. In that manner, I present an unusual case of live coding practice that is unlike previous research in the field. The outcome of this study is compiled by examining the reflective diaries, and I discuss how the non-listening condition can benefit or hinder a live coding session.

3. Design of the study

The design of the study is simulating a performance setting, similar to Collins and Olofsson. By performance setting, I refer to a continuous live set with a predetermined minimum duration and a continuous evolution of sound patterns. There was no systematic preparation before each session, and in most cases, no preparation. All practice sessions were 'blank slate' live coding, that is, with no use of pre-written code. That was my main design exemplar, and I conducted a pilot and an experiment proper (A. F. Blackwell & Aaron, 2015). The study was conducted in two parts, a pilot study took place in August 2022 and a proper one in October 2022. A series of 12 daily live coding sessions were carried out during August, and a month-long daily practice was carried out in October. The experimental designs were different as an outcome of the post-proceedings policy of the PPIG conference.

Part of the critical orientation (A. F. Blackwell & Aaron, 2015) is reflected in the experimental design which was influenced by the seminal study of Davidson (1993) on the perception of expressive performance. This critical orientation led me to question how music listening is useful to the performer. Davidson (1993) designed a study in which she assigned expressive manners to violinists and recorded both video and audio recordings. Later, for validating the expressive manners Davidson conducted a perceptual experimental showing stimuli of full-body movement from the violin performances. Part of the methodological novelty of the study was that the stimuli were based either on visual information only, audio information only, or showing both audio-visual information from the violin performances. In that manner, Davidson controlled the perceptual channels demonstrating that the visual channel can bias our perception of expressivity in music performance. Here, I applied a similar orientation and designed a listening and a non-listening condition, but without assigning any expressive manners or specific tasks during my practice.

During the pilot practice sessions in August, I did three daily sessions of 500 seconds each (36 independent sessions in total). This is approximately 8 minutes, considered a short duration for a live coding performance. The three conditions of the August daily sessions are coded in the file names of the practice sessions as 'No-Audio Level meter' (NAL), 'No-Audio Spectrum' (NAS), and 'Audio-Visual' (AV) conditions. Both NAL and NAS sessions did not include any audio, as I muted the audio from the sound card to the loudspeakers. The NAL sessions were conducted without audio from the loudspeakers, and the only informative cues about the generated sound patterns came from a stereo sound level meter (see Figure 1). The NAS sessions were also conducted without any auditory cues, and the visual cues included both a sound level meter and a spectrum (see Figure 2). During the AV sessions, I could listen to the generated sound patterns and see the visual cues coming out of the sound level meter and the spectrum (see Figure 2).

During the pilot study in August, I focused on a controlled experimental design, and I also aimed to control the listening and acoustic conditions. By controlled conditions, I refer to designing a daily plan about the order of the listening conditions (NAL, NAS, AV), but also to control the sound levels from the computer to the listener. In that manner, I used a MacBook Pro laptop, reproducing the generated sounds from the built-in loudspeakers on maximum levels. I realised that this experimental setup and a controlled experimental setup could be challenging to provide fruitful results for my study. After the PPIG conference and my presentation, I received valuable feedback, which made me redesign the ex-

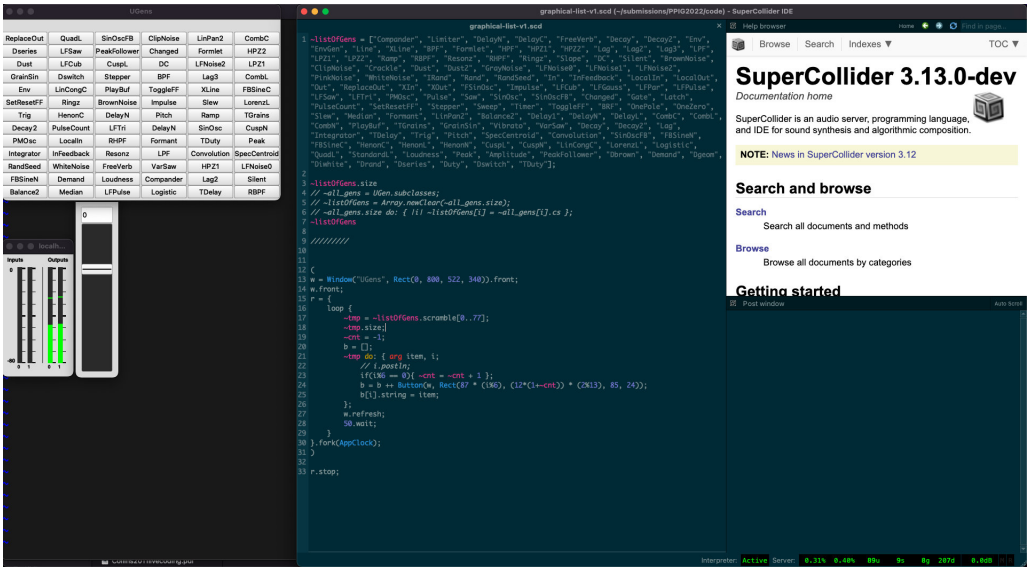

Figure 1 – Pilot experimental setup in SuperCollider for the NAL condition.

perimental setup. The main change was to discard the NAL sessions, as I found them uninformative and maybe annoying. That phase can be seen as the exploratory implementation (A. F. Blackwell & Aaron, 2015). During the experiment proper in October, I did not attempt to control the acoustic conditions. Thus, when I refer to listening to the sound output, I may interchangeably refer to listening from loudspeakers or headphones.

During the month-long experiment in October, I conducted two sessions (AV and NAS) per day of at least 1500 seconds each, by swapping the order of the first and second session every day. This is equivalent to two 25 minutes sessions per day, which is considered an adequate duration for a live coding performance. I manually monitored the duration of every session, and I did not set a hard limit on finishing the sessions. Contrary, during the August sessions, I had programmatically set a hard limit to finalise the sessions after the 500 seconds time limit. For the pilot study in August, every session was catalogued with an audio recording and a timestamped text file showing every code execution. For the experiment proper in October, the audio recordings were not conducted, mainly due to large allocations of memory storage.

The code was captured using the `History` class of SuperCollider. This class can provide reproducible live coding sessions in terms of code executions and audio output. The collected data are timestamped scripts, in the format of plain text files (`*.scd` files), for SuperCollider.

Short-length reflections were written about the daily sessions. The length varies but is no more than 200-300 words per day. Some daily diaries are as short as 1-2 sentences, especially during the October sessions. It is difficult to distinguish whether they can be considered reflective diary entries or note-taking prompts. Complete documentation of the code and the diaries is provided online (`https://gitlab.com/diapoulis/livecodeme/`).

3.1. Visual helper to aid creativity

Part of the experimental design includes a graphical interface (GUI) as a visual helper, as shown in the top-left of Figure 1 and Figure 2. It is a simple helper device that emulates using Post-it notes on the screen (A. Blackwell & Green, 2003). My motivation was to have some visual aid that shows various unit generators (UGens), and get inspiration when building sound synthesis engines. A UGen is the basic building block for sound synthesis engines in many programming languages. SuperCollider has a plethora of UGens, either in the main library or as third-party developments, which makes difficult to recall each one of them. My idea was to begin from a visual helper which could potentially be developed into a software agent. As shown in Figure 1, the initial implementation was done by simply

randomising a list of UGens and printing them on a GUI. Initially, I used a dense matrix format, with dimensions 13x6, refreshing the GUI every 50 seconds. During the pilot study, I selected a constrained set of UGens to be used across the study. The initial list had 128 entries, and I manually selected basic signal generators (like sine and noise oscillators), routing UGens, filters, triggers and envelopes. I found this family of UGens to be limiting my coding practice.

During the study, the GUI did not change significantly. I kept its main passive functionality, and I reduced drastically the amount of UGens shown at a time. During the experiment proper the GUI was showing 8 UGens at a time, and it was updated every 30 seconds (see Figure 2). Some preliminary research on how this can be developed into a software agent was done and is discussed across the diaries. Some of the early conclusions came out of the pilot study, which was to transform the GUI to a 'disruptive' software agent (Attanayake, Swift, Gardner, & Sorensen, 2020), that induces or simply replaces code segments with UGens of similar functionality. This part of the study is still a work in progress.

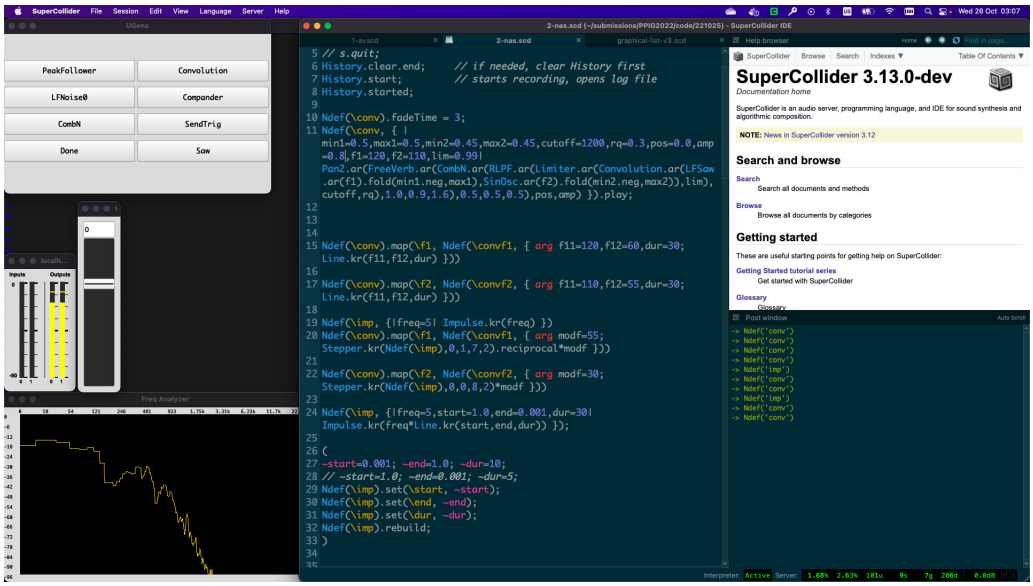

Figure 2 – Experiment proper setup in SuperCollider for the NAS and AV conditions.

3.2. Overall comments on the experimental setup

After my initial attempt to do a controlled study, I feel that I shifted to the other end of having no constraints set in place. Initially, I was inspired from the approach by Fredrik Olofsson while doing the month-long practice sessions with Nick Collins, and did not have a specific coding daily plan. Later I realised that Fredrik Olofsson had also imposed a set of constraints to another practice session that conducted with Marcus Fjellström [3]. In these sessions Olofsson used a single synth definition across all sessions. That is radically different from my experimental design, where I experimented with coding synth representations for sequencing, sound generation, control structures, machine listening among others.

4. Looking into the sound

The design of the study affords an alternative view to live coding. While playing music without listening to it sounds like a no-go, this is the main contribution of this study. Thus, I here present a reflective assessment compiled from the daily diary entries (A. F. Blackwell & Aaron, 2015).

4.1. Reflections from experience

It goes without saying that musical live coding without listening to the sound can hardly be a rewarding activity. Musicians use sound as the primary informative cue during a performance, and I have never

[3] https://fredrikolofsson.com/f0blog/pact-april/

attended a concert where the performers intentionally choose not to listen to the sound. During my non-listening practice sessions, I was sometimes curious to hear what the musical outcome sounded like, and I did sometimes unmute momentarily during the pilot study. During the experiment proper, I did not unmute the non-listening condition, instead, I listened to some of the code recordings after the end of the sessions. It is out of the question that when you live code and do not listen to the sound there are limited possibilities of what can be perceived from the musical outcome. For instance, if I program a sine oscillator and assign a specific frequency, I need perfect pitch to be able to 'hear' how the outcome may sound like. All in all, the only informative cue when non-listening to the sound can be reconstructed by imaging the musical outcome. Musical imagery is a valuable musical ability but it may be a less engaging activity in comparison to music listening. That is, we can be really engaged when listening to the music, and this may be expressed with overt bodily movements, such as dancing, whereas it is less likely to start dancing when imagining a melody of a song.

While live coding does not include any direct involvement between the sound energy and the performed actions, when listening to the sound we may be able to simulate sound actions (Jensenius, 2007, p. 19). Even if there is no direct link to any profound sound actions from daily experience or if the sounds are completely synthetic, we are still able to dance to the beat and produce synchronised bodily movements. The same cannot apply to visual percepts, as we generally perform better in audio-motor synchonisation tasks in comparison to visuo-motor tasks (Hove, Fairhurst, Kotz, & Keller, 2013). Thus, there is an embodied understanding that makes the sound and vibration an indispensable part of music making.

4.2. Reflections from the diaries

In the diaries, I found mixed feelings about the non-listening condition. It looks like I both appreciated some surprisingly appealing musical outcomes but was also annoyed when I had no idea what this may sound like. Several musical tasks are hard to do when non-listening to the sound. I found out that I often forget to apply any panning in non-listening conditions. In principle, many connectivity exercises (Nilson, 2007) can be almost impossible to control when non-listening to the sound. Certain aspects like having a percept about the tempo and the beat are impossible to extract by the visual information on the spectrum, during the non-listening condition. Furthermore, it is quite impossible to track for how long a specific musical pattern is active and produces repetitions, and difficult to understand the length of the musical patterns. I noticed that typically in the non-listening condition, the generated sound pattern had a short duration compared to the listening condition.

When listening to the sound, I sometimes experienced a feeling of performance anxiety. I feel that this can hardly be the case when non-listening to the sound. Furthermore, it felt like I spent more time and was more careful when listening to the sound. It seems logical as any programming mistakes can have significant differences in the produced sound levels, which can cause hearing damage. Thus, it looks like that the listening condition is linked to more careful actions, whereas the non-listening condition produces greater extent of experimentation and trial-and-error practices.

4.3. Psychology of programming when non-listening

Whereas I will not present a detailed analysis on the cognitive dimensions of notation between the listening and non-listening conditions, I will present certain aspects that may improve the quality of discussion (A. Blackwell & Green, 2003). I realised that *progressive evaluations* could be complex when non-listening to the musical outcome. When executing a new code chunk, it can be hard to spot differences from the spectrum. For instance, perceiving differences on the spectrum can be difficult when the musical modifications have slight frequency variations. Consequently this can cause *hard mental operations*, as the console may not show any programming errors. Furthermore, I have noticed that when I listen to the sound, I am more careful in progressive evaluations. That may indicate that the listening condition increases *viscosity*, that is resistance to change, but I think that the non-listening is more likely to cause such imbalance. This is because, *error-proneness* is also increased while non-listening to the sound which may cause *premature commitments*. For instance, I may execute a series of *progressive evaluations* and later realise I did no changes on the running program. Because of my inability to identify any differences on the spectrum, I may think that the progressive evaluations impacted the

generated sound patterns. All this description may only apply to me and amplified to some extend by my obsessive commitment on using only the `Ndef` class for sound synthesis in SuperCollider. Further, my obscure coding practice with long one-liners can also cause severe problems to progressive evaluations, error-proneness and hard-mental operations.

5. Discussion

One of my goals is to motivate other live coders to collaborate often and practice in a daily fashion. Because of the very nature of live coding, as a practice which differs substantially from traditional music performance, I have the feeling that live coders do not practice in a daily fashion (at least, this is the case for me). Carrying out a month-long practice made me realise different things, ranging from my programming skills with my favourite programming language, SuperCollider, to how musical agents can be used in live coding and what programming habits I have when live coding.

The present study began with a view more akin to a controlled experiment and developed into a first-person study based on research through design and reflective diaries. It is important to notice that while I keep an interchangable order between listening and non-listening conditions (AV and NAS) in October I cannot say whether this had any impact on the study. I did learn several things about my live coding practice, and first and foremost that I do not practice as much as I should be practicing. The month-long practice exercise imposed to me to realise my characteristic incompetence (Dahlstedt, 2012) in a live setting, but also to provide me with hope as I did make some progress. I certainly feel more confident after a month of daily practice, and I also believe I have developed a more on-the-fly feeling about the generated sound patterns. A live coder can quickly get into the labyrinth of 'coding for the sake of coding' and sometimes prioritising the code instead of the generated music could be the case. Because of the nature of typing and textual languages, live coding is far from traditional music performance. Maybe the only sensorimotor relation between live coding and instrumental performance is that both are carried out using serial skilled actions (Palmer, 1997).

The non-listening condition demonstrated how cumbersome and sometimes annoying it can be to live code for the sake of coding. When there is no anticipation of the musical outcome, it can be hard to get motivated to continue a live coding session. Sound enhances our anticipatory reward system, especially when combined with visual correspondences. On the other hand, this lack or reduced levels of anticipation because of absence of sound, may lead to unexpected musical outcomes, which can appeal to the coder. It can give a sense of being outside of one's self. Thus, a non-listening condition can be used to generate novel and creative sound patterns. For instance, I can have an educated guess of how a music pattern will sound by simply writing the code expression. An essential skill set of sound synthesis techniques can enable the coder to surprise herself positively. Thus especially in the case of online live streaming, a no-listening condition can be used as a creative technique for music-making. Given that the audience cannot understand the acoustic environment of the performer, the audience can also be unable to perceive whether the coder is listening or not to the generated sounds. Can such audience-performer dynamics bring about any novel interactions? Or do they raise any artistic concerns? Both are hard to answer and go beyond the scope of this study.

The visual helper I used during the sessions showed a limited number of UGens and was not developed to a musical agent. Its impact on my practice was minimal because it did not require attentional resources. On the other hand, I found such a simple program to be an excellent approach to familiarize myself with the large collection of UGens available in SuperCollider. I did a preliminary investigation on the potential of such visual aid, and I can see the potential of transforming this to either an on-demand agent or to a disruptive agent (Attanayake et al., 2020). Such a system would require a natural language processing component to be trained on the code and generate novel code chunks, coupled to a machine listening and learning component capable to extract similarity measures between the running musical outcome and the simulated code evaluations. The git repository can provide the coding database as it contains 98 individual live coding sessions.

It is important to notice that machine listening and learning algorithms have become more accessible.

In the seminal practice session by Collins and Olofsson only a pitch follower UGen is used in a couple of live coding sessions. More studies will follow on that aspect, given the rapid advances in the field of machine listening and machine learning. Indicatively the FluCoMa environment (Tremblay, Roma, & Green, 2021) offers a rich library for applying such computational techniques and is compatible with a variety of programming environments, like MAX/MSP, PureData and SuperCollider. In my sessions I did use some machine listening UGens, but I did not use any machine learning algorithms. This is likely the case because of the blank slate approach, which would require much effort in order to adjust such algorithms to my sessions.

While this is an ongoing research in terms of live coding practice as a disciplined endeavour, several things become apparent. First and foremost when there are no informative visual cues during the non-listening condition it is almost impossible to live code. I found myself loosing the thread of coding and had no idea how the musical outcome may sound like. Further on, from the spectrum alone it can be sometimes difficult to perceive whether command executions have actually any impact on the generated musical outcome.

A month-long practice is an excellent approach to get better on live coding. I found myself improving on several aspects, from creating naming conventions that can be informative, to how to separate triggers and control signals from sound generators. Such programming practices do have an impact on the musical outcome, as I was able to control musical structures more fluidly. Finally, I did carry the practice sessions alone and at times I was feeling demotivated and I was repeating my practices over and over. This feeling of demotivation amplifies the importance of collaboration. Whether the collaboration is as simple as turn-taking sessions or collaborative group performance, it is important to interact with more live coders and exchange ideas and knowledge. I recommend live coders that aim to commit to month-long practice sessions to find a human collaborator.

6. Conclusions

In this study, I challenged myself to do a month-long daily practice in musical live coding. I aimed to examine the relations between auditory and visual percepts and improve my live coding skills. I discussed how listening to the sound can help the coder in progressive evaluations and reduce error-proneness, while non-listening to the sound may be used as a creative coding practice technique. Further, when no informative visual cues and no sound are available to the coder, practicing musical live coding looks like an impossible task as I usually lose the thread of programming. I also experimented with a simple GUI visual helper, which can be developed into a software agent, and the documentation of the study can be used as a dataset for such developments. Practicing live coding in a daily and disciplined fashion is not easy, and several times, I repeated myself in both my coding practices and reflective diaries. The practice of musical live coding can be easier when it is a group activity, and I recommend to live coders who are determined to commit to month-long daily sessions to find at least one more coder.

7. Acknowledgements

I warmly thank the PPIG community for the valuable feedback during the conference, and special thanks to Alan Blackwell for his detailed review of the draft version of the present study. Both were catalytic to the improvement of this study.

8. References

Attanayake, U., Swift, B., Gardner, H., & Sorensen, A. (2020). Disruption and creativity in live coding. In *2020 ieee symposium on visual languages and human-centric computing (vl/hcc)* (pp. 1–5).

Blackwell, A., & Green, T. (2003). Notational systems–the cognitive dimensions of notations framework. *HCI models, theories, and frameworks: toward an interdisciplinary science. Morgan Kaufmann*, *234*.

Blackwell, A. F., & Aaron, S. (2015). Craft practices of live coding language design. In *Proc. first international conference on live coding.*

Blackwell, A. F., & Collins, N. (2005). The programming language as a musical instrument. In *Ppig*

(p. 11).
Burland, K., & McLean, A. (2016). Understanding live coding events. *International Journal of Performance Arts and Digital Media*, *12*(2), 139–151.
Collins, N., McLean, A., Rohrhuber, J., & Ward, A. (2003). Live coding in laptop performance. *Organised sound*, *8*(3), 321–330.
Dahlstedt, P. (2012). Between material and ideas: A process-based spatial model of artistic creativity. In *Computers and creativity* (pp. 205–233). Springer.
Dal Rì, F. A., & Masu, R. (2022). Exploring musical form: Digital scores to support live coding practice. In *Nime 2022.*
Davidson, J. W. (1993). Visual perception of performance manner in the movements of solo musicians. *Psychology of music*, *21*(2), 103–113.
Gamboa, M. (2022). Conversations with myself: Sketching workshop experiences in design epistemology. In *Creativity and cognition* (pp. 71–82).
Hove, M. J., Fairhurst, M. T., Kotz, S. A., & Keller, P. E. (2013). Synchronizing with auditory and visual rhythms: an fmri assessment of modality differences and modality appropriateness. *Neuroimage*, *67*, 313–321.
Jensenius, A. R. (2007). Action-sound: Developing methods and tools to study music-related body movement.
Magnusson, T. (2015). Code scores in live coding practice. In *Proceedings of the international conference for technologies for music notation and representation, paris* (Vol. 5).
Nilson, C. (2007). Live coding practice. In *Proceedings of the 7th international conference on new interfaces for musical expression* (pp. 112–117).
Palmer, C. (1997). Music performance. *Annual review of psychology*, *48*(1), 115–138.
Roberts, C., & Wakefield, G. (2018). *Tensions and techniques in live coding performance.*
Sorensen, A. C., & Brown, A. R. (2007). aa-cell in practice: An approach to musical live coding. In *International computer music conference* (pp. 292–299).
Tremblay, P. A., Roma, G., & Green, O. (2021). Enabling programmatic data mining as musicking: The fluid corpus manipulation toolkit. *Computer Music Journal*, *45*(2), 9–23.