Exploring Activity Traces from Live Coding Musicians

DANIEL MANESH, Virginia Tech, USA

Additional Key Words and Phrases: Live Coding, creativity support, creative version control, creative activity traces

1 Introduction and Background

In the creative arts realm, live coding refers to a performance practice where the performer writes, edits, and runs code live in front of an audience to create an audiovisual performance [2, 4, 8]. While live coding can be completely improvisational [2], in practice, many live coders prepare pre-composed snippets in advance, or even plan out their entire performance [1, 3, 5]. Like most creative activities, live coding is marked by dynamic exploration, which can occur live in front of an audience or in preparation as a live coder is composing a piece on their own.

Live coding is a promising area of study for digital creativity researchers because it is a creative act that naturally lends itself to digitization. For example, live coding has readily available activity traces in the form of full code replays [9]. Another approach is to use checkpoints (i.e., program versions) determined, for example, by when live coders run their code [5]. Regarding this second approach, there has been rising research interest in examining version control in creative contexts [6, 7]. Creative practitioners have different needs for version control than software engineers do—for example, creatives are more likely to draw on previous versions of their work for inspiration [7]. By studying how we can understand and make use of version histories in live coding, we may learn transferrable lessons for designing creative version control systems in general.

I propose exploring the following two research questions related to live coding.

- RQ1: What can live coding musicians learn, and what might they want to learn, by examining and reflecting on live coding activity traces?
- RQ2: How can we effectively structure and represent activity trace data to live coders to augment their creative practice?

To address RQ1, I propose a study where live coders install a plugin which records their live coding activities unobtrusively. After using the plugin for multiple sessions, the live coders and researchers will have a chance to asynchronously examine the creative activity trace data and then will meet to discuss what types of insights they found and what difficulties they had navigating the data. This initial study can inform the design of a system to address RQ2. I propose addressing RQ2 through a series of co-design workshops with the live coding community, progressing from ideation and feedback on mock-ups to the implementation and testing of a full-fledged system. I expand on these two study proposals below.

2 Proposed Study 1: Exploring Activity Traces

To address RQ1, I propose a study in three stages. In the first stage, participants will complete several open-ended live coding sessions on their own time, but using a lightweight IDE extension which records their activity traces (e.g., keystroke data). In the second stage, the participants will independently review their live coding sessions using a simple

Author's Contact Information: Daniel Manesh, danielmanesh@vt.edu, Virginia Tech, Blacksburg, Virginia, USA.

2018. Manuscript submitted to ACM

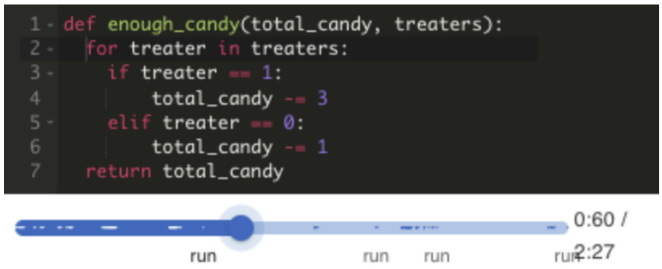

Fig. 1. A code replay system from Xie et al. [9] that includes a timeline indicating when the code was edited and when it was run. I propose using a similar, simple system in a formative study to understand what types of information live coders are interested in from their code replays.

timeline interface that replays the session. In the third and final stage, the participant and researchers will meet to discuss their experiences.

For the first stage, we will need to build a IDE extension which records live coding sessions. One potential concern is privacy: participants may be hesitant about recording their keystroke data and providing it to the research team. To mitigate this concern, we can have the extension only save the data locally, and give participants the option to share with the researchers only if they want to. Another concern is the wide variety of code editors that can be used for live coding. In order to address this concern, we will plan to develop for one or two of the more popular platforms (e.g., Pulsar or VS Code) and recruit participants who already use these tools.

For the second stage, we will need to design a simple interface for reviewing the activity traces. We propose a basic timeline interface with key events, such as code runs, displayed on the timeline (e.g., similar to prior work [9]). We intentionally opt for a familiar and easy-to-use interface, as we want to avoid biasing the participants towards specific ways of looking at the data. On their own time, participants will write down and submit their reflections. Having a written record can help jog their memories for the final interview, and the act of writing may encourage deeper reflection.

For the final stage, we will conduct interviews to discuss each participant's experience and reflections. The researchers will have already read the submitted reflection and may also have been able to review the participant's activity traces (if they opted in). During the interview, we will ask questions about the participants' reflections, aiming to understand what information they were interested in, how they might like to use that information, what frictions they had with the interface, and what other information they may have wanted that was not available.

3 Proposed Study 2: Co-designing a System for Interacting with Activity Traces

The goal of this study is to co-design and test a system for making sense of activity trace data for live coding. The target use case will be for live coders to augment their own practice, though it could also be used to reflect on other people's live coding processes.

Because study 1 is not yet complete, there are many open questions about the design goals of the system. For example, we could design our system to be used in a live setting, surfacing the activity trace data of the current session as is done in the existing system SHARP [5] (see Figure 2). In contrast, we might focus on an offline setting where, for example, live coders might draw inspiration from activity traces across several older sessions.

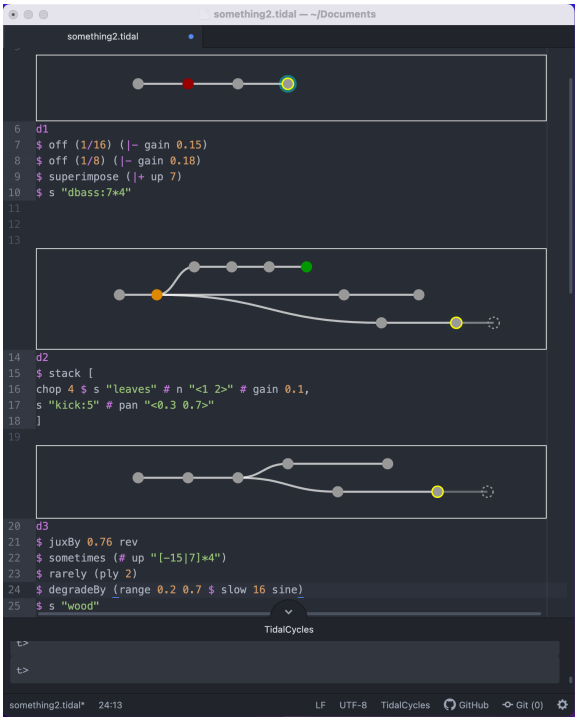

Fig. 2. A screenshot of existing system SHARP, which provides block-level version trees for the current live coding session [5]. SHARP could be used as a starting point for the system described in Study 2, but we may also want to explore version activities across sessions or organized in other ways. Ultimately, the goals of our system will be determined by Study 1 and our proposed co-design sessions for Study 2.

We propose a co-design process for this phase, where researchers work together with live coders using an iterative design approach across multiple workshops. Initial workshops would involve generating brainstorming design ideas with live coders and discussing low-fidelity mock-ups inspired from the results from the first study. From there, we can create one or two usable prototypes which can be used and discussed by participants in the next workshop. Ideally, once our system reaches maturity, we can make it widely available to the live coding community and conduct a longitudinal study based on real-world usage.

4 Conclusion

In conclusion, we propose two studies to collect and understand creative activity traces from live coding musicians. We believe live coding provides a promising opportunity to explore activity traces and version control in creative contexts, as it is a creative act that lends itself easily to digitization. The results from our studies may help inform creativity support research beyond just live coding musicians and can contribute to the wider discussion on creative version control.

References

[1] Alan F Blackwell, Emma Cocker, Geoff Cox, Alex McLean, and Thor Magnusson. 2022. *Live coding: a user's manual.* MIT Press.

[2] Nick Collins, Alex McLEAN, Julian Rohrhuber, and Adrian Ward. 2003. Live coding in laptop performance. *Organised Sound* 8, 3 (Dec. 2003), 321–330. doi:10.1017/S135577180300030X

[3] Georgios Diapoulis. 2023. Musical Live Coding in Relation to Interactivity Variations. *Organised Sound* 28, 2 (2023), 149–161. doi:10.1017/S1355771823000444

[4] Thor Magnusson. 2011. Algorithms as Scores: Coding Live Music. *Leonardo Music Journal* 21 (12 2011), 19–23. arXiv:https://direct.mit.edu/lmj/article-pdf/doi/10.1162/LMJ_a_00056/1675039/lmj_a_00056.pdf doi:10.1162/LMJ_a_00056

[5] Daniel Manesh, Douglas Bowman Jr., and Sang Won Lee. 2024. SHARP: Exploring Version Control Systems in Live Coding Music. In *Proceedings of the 16th Conference on Creativity & Cognition* (Chicago, IL, USA) *(CC '24)*. Association for Computing Machinery, New York, NY, USA, 426–437. doi:10.1145/3635636.3656195

[6] Eric Rawn, Jingyi Li, Eric Paulos, and Sarah E. Chasins. 2023. Understanding Version Control as Material Interaction with Quickpose. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 1–18. doi:10.1145/3544548.3581394

[7] Sarah Sterman, Molly Jane Nicholas, and Eric Paulos. 2022. Towards Creative Version Control. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2, Article 336 (nov 2022), 25 pages. doi:10.1145/3555756

[8] Ge Wang and Perry R. Cook. 2004. 2004: On-the-Fly Programming: Using Code as an Expressive Musical Instrument. In *A NIME Reader*, Alexander Refsum Jensenius and Michael J. Lyons (Eds.). Vol. 3. Springer International Publishing, Cham, 193–210. doi:10.1007/978-3-319-47214-0_13 Series Title: Current Research in Systematic Musicology.

[9] Benjamin Xie, Jared Ordona Lim, Paul K.D. Pham, Min Li, and Amy J. Ko. 2023. Developing Novice Programmers' Self-Regulation Skills with Code Replays. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1* (Chicago, IL, USA) *(ICER '23)*. Association for Computing Machinery, New York, NY, USA, 298–313. doi:10.1145/3568813.3600127