Turn-Taking and Chatting in Collaborative Music Live Coding

Anna Xambó[1,2], Pratik Shah[3], Gerard Roma[2,1], Jason Freeman[1] and Brian Magerko[2]
[1]Center for Music Technology, Georgia Institute of Technology
[2] Digital Media Program, Georgia Institute of Technology
[3] School of Interactive Computing, Georgia Institute of Technology
Atlanta, Georgia, USA
{anna.xambo,pratikshah,gerard.roma,jason.freeman,magerko}@gatech.edu

ABSTRACT

Co-located collaborative live coding is a potential approach to network music and to the music improvisation practice known as live coding. A common strategy to support communication between live coders and the audience is the use of a chat window. However, paying attention to simultaneous multi-user actions, such as chat texts and code, can be too demanding to follow. In this paper, we explore collaborative music live coding (CMLC) using the live coding environment and pedagogical tool EarSketch. In particular, we examine the use of turn-taking and a customized chat window inspired by the practice of pair programming, a team-based strategy to efficiently solving computational problems. Our approach to CMLC also aims at facilitating the understanding of this practice to the audience. We conclude discussing the benefits of this approach in both performance and educational settings.

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing systems and tools**; **Computer supported cooperative work**; • **Social and professional topics** → *Computing education*;

KEYWORDS

Live coding, collaboration, CSCW, pair programming

ACM Reference Format:
Anna Xambó[1,2], Pratik Shah[3], Gerard Roma[2,1], Jason Freeman[1] and Brian Magerko[2]. 2017. Turn-Taking and Chatting in Collaborative Music Live Coding. In *Proceedings of AM '17, London, United Kingdom, August 23–26, 2017,* 5 pages.
https://doi.org/10.1145/3123514.3123519

1 INTRODUCTION

Live coding and collaboration is a promising approach to the established discipline of *computer-supported collaborative work* (CSCW), but it is still unclear how to best support collaboration: shall live coders work in their different environments, or shall they work on the same? Shall they share code and mutually modify others' code, or shall they keep their code as an individual, unmodifiable contribution? Among the different potential configurations to collaborative live coding, pair programming is a well-known practice in CS education and industry that organizes the group workflow in a turn-taking fashion.

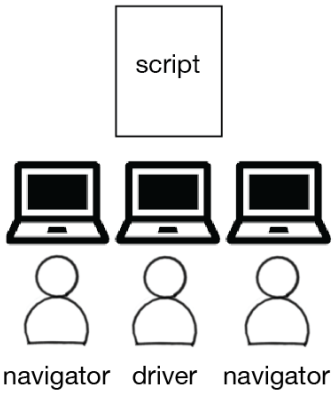

Figure 1: Approach to trio live coding.

In this paper, we explore the benefits of supporting communication features to turn-taking and pair programming in EarSketch [10], a live coding environment and educational tool. We particularly explore the use of a chat window and a shared code editor (similar to Google Docs-like collaborative editing) in co-located pair programming with a group of two and three users respectively.

Our main research question is *how do music live coders collaborate, and what design paradigms for live coding tools can effectively support these collaboration modes?* In particular, we aim at examining whether and how turn-taking supports collaborative live coding. We are interested in understanding: (1) what are the tasks undertaken by the different collaborators based on the pair programming model of driver vs navigator(s); (2) what are the differences between the configurations of one driver vs one navigator and one driver vs two navigators; (3) whether a chat window is an effective mechanism for communication and, if so, how it is used; and (4) whether using hashtags in the chat window is helpful in a collaborative session. We adopt an autoethnographic approach already used for understanding a live coding environment [17] because it can help to gain a first insight from a live coder perspective.

This study is a follow-up of the study reported in [26]. In the prior study, we discussed the benefits of co-located *collaborative music live coding* (CMLC) in the classroom and proposed three approaches

AM '17, August 23–26, 2017, London, United Kingdom

ACM ISBN 978-1-4503-5373-1/17/08…$15.00
https://doi.org/10.1145/3123514.3123519

to real-time collaboration: (1) a shared script accessed from a single terminal in turns (*live pair programming*); (2) a shared script accessed from multiple individual terminals at any time (*multiple live coding*); and (3) a shared script accessed from two individual terminals at any time (*pair live coding*). In this paper, we explore a merged version of these three approaches to CMLC (see Fig. 1). Each user has an individual terminal like in pair or multiple live coding, the roles are strictly divided into driver and navigator like in live pair programming, with the addition of a second navigator like in multiple live coding. This study is framed in a performance setting, however it still looks into the live coding synergies between performance and education.

2 RELATED WORK

2.1 Turn-Taking and Pair Programming in CSCW

The organization of *turn-taking* has been studied in conversation analysis since late 1960s and early 1970s [22]. Supporting turn-taking has been a topic of research of CSCW and groupware technology since early 1990s [18]. *Interaction analysis* has included both verbal and non-verbal communication when analyzing technology-enhanced collaborative settings [14], where both *talk-driven interaction* (e.g., 'turns at talk') and *instrumental interaction* (e.g., 'turns with bodies' and 'turns with artifacts', such as two kids taking turns with a mouse when playing a game) are examined.

Pair programming is an established practice in programming where two programmers, a *driver* and a *navigator*, collaborate on the same computational problem (e.g., designing, coding or testing) and after a certain amount of time (e.g., 10 minutes) they switch roles [25]. While the driver is in charge of writing the code, the navigator is responsible of both the long-run thinking, in particular the nitty-gritty details and potential errors, such as syntax errors [25]. Turn-taking between a small number of users is described in a seminal study on designing systems for collaborative musical experiences [4] as a useful protocol for assessing new musical instruments, notably in [2, 7]. This research is influenced by pair programming practices but supporting larger groups than pairs in a turn-taking fashion of a driver and a few navigators and applied to the domain of music computing, which is novel in the literature.

2.2 Communication Tools in CSCW

There exist a number of environments that support real-time collaboration for text editing (e.g., SubEthaEdit,[1] ShareLatex,[2] Overleaf)[3]; learning (e.g., Virtual Math [16]); and coding (e.g., CodeCircle [9]). The most common characteristics for supporting real-time collaboration, in alignment with CSCW techniques, include supporting sharing links, users' online presence, color-coding users, levels of access permissions, communication tools (e.g., chat, commenting), visual feedback of real-time multiple-interaction, version history, and error handling as a shared experience.

Facilitating networked collaboration and communication is reported as essential for CMLC [15]. Music live coding environments that support collaboration include Overtone [1], Gibber [21], Betablocker [20], the Republic quark [5] in SuperCollider, and Fluxus [20]. In Overtone, there is a text chat panel that lets the members of a session send messages separated from the code. In Gibber, users can meet in a chat room where real-time collaborative code editing is possible (like GoogleDocs yet with no colored users). Editing is in sync, but execution is independent for each user. Other environments that support co-located and remote text-based collaborative music improvisation and include a chat window are LOLC [11], urMus [15], and Lich.js [19]. A study on a CSCW music system reported that the chat was used for exchanging descriptions of the users' activities [8].

Our paper is based on the use of the communication tools traditionally used in CSCW for supporting awareness and communication among a team, such as a chat, a buddy list, and the code editor as a shared space. Turn-taking in EarSketch is a a first step towards supporting real-time collaboration using these tools. It is an open question how to use these tools effectively during a CMLC session, which is tackled next.

2.3 Language in SMS and Social Media

In the literature, we find that using short messaging system (SMS) texting and social media hashtags do not affect negatively formal writing [3, 23]. Features of SMS messaging include the use of short utterances, abbreviations, emoticons, and other symbols [12]. Syntactical and lexical reductions are applied in SMS texting to reduce effort, time and space [13].

Our approach to SMS and social media hashtags is to take advantage of their established use among students but applied to facilitate the communication between live coders during a CMLC session. Here the use of the SMS texts and social media hashtags benefits an informal learning environment where efficient communication is in the fore. Supporting an informal language style, commonly used between students, can raise interest in programming, which is a well-known challenge in CS education [10].

3 THE STUDY

EarSketch [10] is a free online tool and curriculum designed for learning to code by making music using audio samples, beats, and effects. It is inspired by a digital audio workstation (DAW) interface and supports both composition features and live coding features [26]. Next, we detail how EarSketch is also incorporating collaborative features.

3.1 EarSketch and Collaboration

Collaboration in EarSketch has been divided into three phases. Each phase introduces new features to support richer collaboration between users. Phase 1 has been implemented, while phase 2 is under development, and phase 3 is future work.

3.1.1 Phase 1. The first phase of collaboration in EarSketch introduced the capability of sharing scripts through links or users and to share the audio on SoundCloud. Another feature added was to automatically retain the metadata of the original script, such as the title and author name. Using the workflow of importing the code before allowing users to edit ensures that the authorship is attributed when the users share or publish the script.

[1] https://codingmonkeys.de/subethaedit (accessed March 27, 2017).
[2] http://sharelatex.com (accessed March 27, 2017).
[3] https://overleaf.com (accessed March 27, 2017).

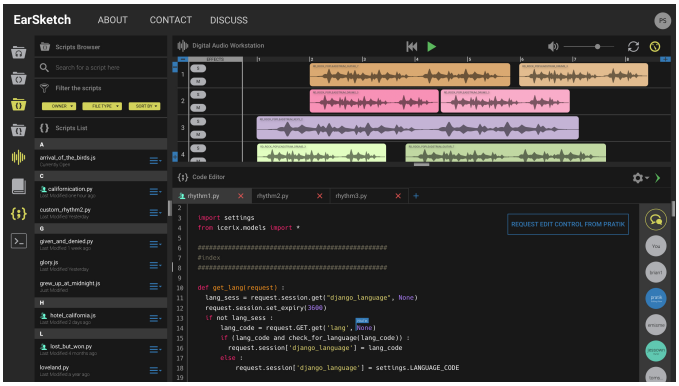

Figure 2: Turn-taking collaborative editing - Navigator View.

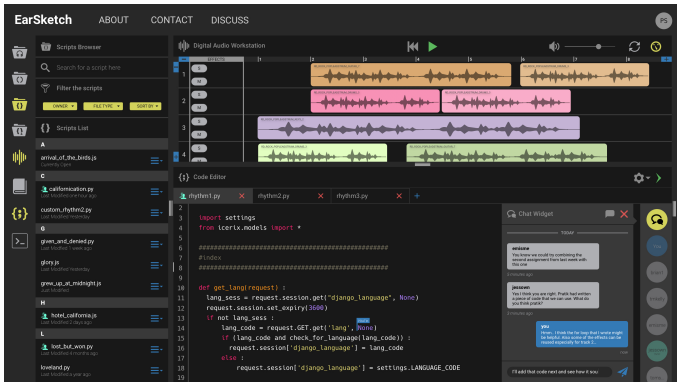

Figure 3: Instant messages between collaborators through the chat window.

3.1.2 Phase 2. The second phase of collaboration in EarSketch will allow multiple users to work on the same script, allowing for the possibility of real-time collaborative script editing. This type of collaboration follows the model of turn-taking where only one of the users is the driver and all other users are navigators. The navigators can monitor the changes as they are able to see the edits in real-time along with the cursor of the driver. There will be visual feedback to enforce the differences between driver and navigators. Editing control can be requested from the driver or the original author/owner of the script, at any time, by any navigator (see Fig. 2). The second phase will also introduce comprehensive support for notifications throughout the system. These notifications can be related to newly shared scripts, EarSketch announcements or any other aspect which the user needs to be notified of.

3.1.3 Phase 3. The final phase of collaboration in EarSketch will introduce features that support peer-to-peer communication. We identify two kinds of peer-to-peer communications in EarSketch. The first type is used to convey information about planning and other subjective aspects of the script creation and editing. The second type is used to notify users about potential errors, which serves as a debugging tool that also has pedagogical value. We envision that the first type of communication will be supported via Instant Messaging between the collaborators using a shared chat window (see Fig. 3).

3.2 Proof of Concept and Study Procedure

We tweaked a version of EarSketch to add collaborative editing and a chat window for the proof of concept discussed in this paper. As explained above, the aim was at testing some of the future features to be included in phases 2 and 3 of collaboration in EarSketch. We used Firepad[4] and Firechat,[5] which are both based on Firebase.[6]

The study procedure consisted of asking the group to create a live coding session. The group was asked to work on a shared Python script of EarSketch using their own individual laptops. The group was expected to write code by turn-taking and to communicate between the team members (e.g., taking decisions) via a chat window included in EarSketch. Each of the participants was suggested to be at least once in the role of writing code. The group was encouraged to communicate only using the chat window, as if it was a performance setting. One laptop was connected to two loudspeakers and the respective live coder was responsible to run the script when one of the live coders asked for it. The other live coders were wearing headphones that allowed them to preview sounds or scripts independently, if necessary.

The use of the following hashtags was suggested to the group for the communication in the chat window: *#req* for requesting a turn, *#go* for giving the turn, and *#run* for running the code in case the laptop's person was not connected to the loudspeakers. Direct messages were encouraged using the username preceded with an ampersand symbol (e.g., *@username*). Both scripts and comments in the chat window were stored for data analysis. The session was video recorded using the screencast software SnagIt.[7] The screencast software captured the screen from the computer that was connected to the loudspeakers while the group was working with EarSketch.

4 EXPERIENCE AND DISCUSSION

We informally tried two sessions of CMLC with a trio and a duo. The three are expert musicians, and one (who performed in the trio) is an expert in EarSketch.[8] Each session lasted around 20 minutes. In this section, we report and compare each of the two experiences from an autoethnographic perspective. We discuss the two approaches within the broader picture of live coding in CSCW and highlight next steps.

4.1 Trio Live Coding

Two navigators and one driver allowed us to have more specialist roles. One task of the navigator was to preview sounds with their headphones and suggest audio samples' names that could suit well. This task was done autonomously or requested by another live coder. If there was an error when executing the code from the main terminal, previewing the error from another terminal was also used to solve the problem. As the piece was using algorithmic elements,

[4]https://firepad.io (accessed April 10, 2016).
[5]https://firechat.firebaseapp.com (accessed April 10, 2016).
[6]https://firebase.google.com (accessed April 10, 2016).
[7]https://techsmith.com/screen-capture.html (accessed April 10, 2016).
[8]A video sample of both sessions can be found here: https://vimeo.com/212639022.

every time that we pressed the "run" button there was a change and thus executing the code was found aesthetically interesting. We ended the piece with a combination of quick changes and executions of the code that resulted in a compelling glitch style.

We as a group constantly used the chat window to explain unclear code snippets (e.g., "master track would be MASTER_TRACK instead of track number"), share plans of individual actions (e.g., "I want to make a longer beat string algorithmically"), or suggest actions that others could do (e.g., "maybe mute the tracks one by one?"). Having three on board, allowed us to tell others when we were done or tired of driving the session (e.g., "and now ready for someone else with fresh ideas..."). The three suggested tags were used for requesting, granting and asking the live coder of the master terminal to execute code. However, direct or group messages were assumed without making them explicit. For example, *@username* was not used. The chat window was also used to plan parts of the piece (e.g., "btw we can start doing an ending?"), ask for opinion or help (e.g., "are any of the groove sounds good here? e.g. HOP_DUSTYGROOVEPART_001"), or comment about the musical results (e.g., "this is nice"). We requested an equal frequency of turns during the session. One of the live coders is an expert of EarSketch, so he helped the team when using other functions than the common `fitMedia()` or `makeBeat()`. For example, the expert live coder explained the `makeBeatSlice()` function via the chat window (e.g., "each successive number is an index to a different timestamp in the sound"). This saved us time of looking at the EarSketch curriculum and was an instance of *situated peer learning* [6].

4.2 Duo Live Coding

If compared to the trio live coding, we also equally requested turns during the session, but the pace was faster. From a driver perspective, it was more participative, but it also felt more linear, like in a Ping Pong game of back and forth with the ball. The session felt more predictable. In particular, the roles were less specialized, focusing on alternating between driving and navigating. Often times, as a navigator we were planning our next move as drivers and paying less attention to the other's actions. The chat window was used similar to how we used it in the trio live coding session: as a tool for communication to discuss about the code and the musical output.

4.3 Lessons Learned

As live coders, we found the model of turn-taking a little bit slow for a performance setting. It would be interesting to combine it with Google Docs-like collaborative editing, discussed as the multiple live coding approach in [26]. It is still an open question how to combine the structure provided by turn-taking with a more freestyle format provided by simultaneous multiple editors so that both performers and audience understand what is going on.

We found the preview feature was useful to the group, which allowed us to preview sounds before adding them to the shared script or solve an error in the code while others where working with other tasks in parallel. Providing personal spaces as essential for collaboration aligns with previous research on CSCW applied to music [8].

The two members who experienced both duo and trio live coding found it more interesting to work as a trio live coding than as a duo. Partly the fact that one of the live coders is an expert of EarSketch helped. In addition, the combination of multiple specialized tasks made the improvisation more varied and serendipitous. We discussed that the live coder needs to be skillful with the programming language, as the musician needs to master the musical instrument, even more if it is only a group of two live coders. Within larger groups, an expert in the team can help to make worthwhile the turn-taking approach, as someone who has the broader picture. In an educational context, combining expert students with novice students seems sensible.

Turn-taking in live coding reminded us of the *cadavre exquis* collaborative game, where the more the merrier for creative discovery, or hip hop concerts with several MCs. The experience of turn-taking recalls what Wessel and Wright [24] describe as the *catch and throw* metaphor in musical control, where there is a dialogue between a group of musicians, in which the musical material is received, modified and sent in real time. In this case there is a shared space where the musical material is changed under request.

Using a chat window and hashtags was an effective tool of communication. Connecting hashtags to visible notifications that go to either a user or the group seems like the next step for improving the communication in EarSketch. This aligns with the planned implementation of adding the turn-taking feature, a chat window, and roles associated to a script (e.g., owner and collaborators).

5 CONCLUSION AND FUTURE WORK

This paper looked into turn-taking and chatting in CMLC using the CS educational online platform EarSketch. We compared duo and trio live coding from an autoethnographic stance. We discovered that a trio live coding can be more interesting in performance because the roles of a driver and two navigators, borrowed from pair programming, can specialize and adapt easily during the musical improvisation act. We found that the role of a chat window is an important tool for supporting communication in CMLC, and that using hashtags was promising. However, we also foresee that an hybrid form of turn-taking combined with multi-editing a shared script can be the most interesting result in performance. We speculate that turn-taking and chatting in collaborative live coding between groups of two, three, or larger groups, can be useful in the classroom for pedagogical purposes. We plan to continue this research towards improving collaboration in EarSketch, supporting both performance and education settings. Future work also includes gathering audience response and exploring group dynamics.

ACKNOWLEDGMENTS

The EarSketch project receives funding from the National Science Foundation (CNS #1138469, DRL #1417835, DUE #1504293, and DRL #1612644), the Scott Hudgens Family Foundation, the Arthur M. Blank Family Foundation, and the Google Inc. Fund of Tides Foundation.

REFERENCES

[1] Samuel Aaron and Alan F. Blackwell. 2013. From Sonic Pi to Overtone: Creative Musical Experiences with Domain-Specific and Functional Languages. In *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling*

& Design (FARM '13). 35–46.

[2] Hansen Anne-Marie, Jørgen Andersen Hans, and Raudaskoski Pirkko. 2012. Two Shared Rapid Turn Taking Sound Interfaces for Novices. In *Proceedings of the 12th International Conference on New Interfaces for Musical Expression (NIME '12)*. 470–473.

[3] Shazia Aziz, Maria Shamim, Muhammad Faisal Aziz, and Priya Avais. 2013. The Impact of Texting/SMS Language on Academic Writing of Students: What Do We Need to Panic About? *Elixir Linguistics and Translation* 55 (2013), 12884–12890.

[4] Tina Blaine and Sidney Fels. 2003. Contexts of Collaborative Musical Experiences. In *Proceedings of the 3rd International Conference on New Interfaces for Musical Expression (NIME '03)*. 129–134.

[5] Alberto de Campo. 2014. Republic: Collaborative Live Coding 2003-2013. Dagstuhl, 152–153.

[6] Pierre Dillenbourg. 1999. What Do You Mean by 'Collaborative Learning'? In *Collaborative Learning: Cognitive and Computational Approaches*, P. Dillenbourg (Ed.). Vol. 1. Elsevier, Oxford, UK, 1–19.

[7] Sidney Fels and Florian Vogt. 2002. Tooka: Explorations of Two Person Instruments. In *Proceedings of the 2002 Conference on New Interfaces for Musical Expression*. 1–6.

[8] Robin Fencott and Nick Bryan-Kinns. 2010. Hey Man, You're Invading my Personal Space! Privacy and Awareness in Collaborative Music. In *Proceedings of the 10th International Conference on New Interfaces for Musical Expression (NIME '10)*. 198–203.

[9] Jakub Fiala, Matthew Yee-King, and Mick Grierson. 2016. Collaborative Coding Interfaces on the Web. In *International Conference of Live Interfaces (ICLI 2016)*. 49–58.

[10] Jason Freeman, Brian Magerko, Tom McKlin, Mike Reilly, Justin Permar, Cameron Summers, and Eric Fruchter. 2014. Engaging Underrepresented Groups in High School Introductory Computing through Computational Remixing with EarSketch. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. 85–90.

[11] Jason Freeman and Akito Van Troyer. 2011. Collaborative Textual Improvisation in a Laptop Ensemble. *Computer Music Journal* 35, 2 (2011), 8–21.

[12] Robert Godwin-Jones. 2005. Emerging Technologies: Messaging, Gaming, Peer-to-Peer Sharing: Language Learning Strategies & Tools for the Millennial Generation. *Language Learning & Technology* 9, 1 (2005), 17–22.

[13] Mohd. Sahandri Gani B. Hamzah, Mohd. Reza Ghorbani, and Saifuddin Kumar B. Abdullah. 2009. The Impact of Electronic Communication Technology on Written Language. *Online Submission* 6, 11 (2009), 75–79.

[14] Brigitte Jordan and Austin Henderson. 1995. Interaction Analysis: Foundations and Practice. *The Journal of the Learning Sciences* 4, 1 (1995), 39–103.

[15] Sang Won Lee and Georg Essl. 2014. Communication, Control, and State Sharing in Collaborative Live Coding. In *Proceedings of the 14th International Conference on New Interfaces for Musical Expression (NIME '14)*. 263–268.

[16] Rachel M Magee, Christopher M Mascaro, and Gerry Stahl. 2013. Designing for Group Math Discourse. In *2013 Conference on Computer Supported Collaborative Learning (CSCL 2013)*. 312–319.

[17] Thor Magnusson. 2011. Confessions of a Live Coder. In *Proceedings of the International Computer Music Conference 2011 (ICMC '11)*. 609–616.

[18] Andy McKinlay, Rob Procter, Oliver Masting, Robin Woodburn, and John Arnott. 1994. Studies of Turn-Taking in Computermediated Communications. *Interacting with Computers* 6, 2 (1994), 151–171.

[19] Chad Mckinney. 2014. Quick Live Coding Collaboration in the Web Browser. In *Proceedings of the 14th International Conference on New Interfaces for Musical Expression (NIME '14)*. 379–382.

[20] Alex McLean, Dave Griffiths, Nick Collins, and G. A. Wiggins. 2010. Visualisation of Live Code. In *Visualisation and the Arts*. 1–5.

[21] Charles Roberts and JoAnn Kuchera-Morin. 2012. Gibber: Live Coding Audio in the Browser. In *Proceedings of the International Computer Music Conference 2012 (ICMC '12)*. 64–69.

[22] Harvey Sacks, Emanuel A. Schegloff, and Gail Jefferson. 1974. A Simplest Systematics for the Organization of Turn-Taking for Conversation. *Language* 50, 4 (1974), 696–735.

[23] Latisha Asmaak Shafie, Norizul Azida, and Nazira Osman. 2010. SMS Language and College Writing: The Languages of the College Texters. *International Journal of Emerging Technologies in Learning* 5, 1 (2010), 26–31.

[24] David Wessel and Matthew Wright. 2002. Problems and Prospects for Intimate Musical Control of Computers. *Computer Music Journal* 26, 3 (2002), 11–14.

[25] Laurie Williams and Robert Kessler. 2002. *Pair Programming Illuminated*. Addison-Wesley Longman Publishing Co., Inc.

[26] Anna Xambó, Jason Freeman, Brian Magerko, and Pratik Shah. 2016. Challenges and New Directions for Collaborative Live Coding in the Classroom. In *International Conference of Live Interfaces (ICLI 2016)*. Brighton, UK.