Challenges and New Directions for Collaborative Live Coding in the Classroom

Anna Xambó[1,2], Jason Freeman[1], Brian Magerko[2] and Pratik Shah[3]

1 Center for Music Technology, Georgia Institute of Technology, Atlanta, GA, USA
anna.xambo@gatech.edu
jason.freeman@gatech.edu
2 Digital Media Program, Georgia Institute of Technology, Atlanta, GA, USA
magerko@gatech.edu
3 School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, USA
pratikshah@gatech.edu

Abstract. This paper focuses on the potential of collaborative live coding in educational settings. In particular, it draws from our experience with EarSketch, a free online platform for algorithmic composition and computational music remixing that allows students to learn Computer Science Principles (CSP) by making music using either Python or JavaScript. We argue that collaborative live coding is a promising approach to learning CSP through computer music in the classroom. We draw on interviews with teachers and observations in schools. We discuss how collaboration can be better supported in educational settings when learning CSP using EarSketch, and the challenges and potential for learning music and code using a computer-supported collaborative environment.

Keywords: Collaboration, Live Coding, Peer-to-Peer Communities of Practice, STEAM Education.

Introduction

Knowing how to code is an essential skill of the 21st century. National programs, such as the recent American "Computer Science (CS) for All",[1] evidence the emergent interest of providing the required resources to allow students to learn to code early on. These programs aim to broaden participation in Science, Technology, Engineering, and Math (STEM) fields, in which certain populations are underrepresented, such as women, Afro-Americans, and Latinos (Burge and Suarez 2005; Freeman et al. 2014). A successful approach is including the arts when teaching STEM subjects, also known as Science, Technology, Engineering, Art, and Math (STEAM) education. In particular, initiatives such as EarSketch (Freeman et al. 2014) have proven to engage underrepresented populations in STEM disciplines by bringing musical artistic practices with low barrier of entry into STEM.

EarSketch[2] is a free online learning platform and curriculum for algorithmic composition and computational music remixing.[3] EarSketch allows students to easily create music by combining audio samples, using either Python or JavaScript, within a Digital Audio Workstation (DAW) inspired interface. The educational platform has been deployed in a number of high schools, summer camps, and college courses. There is an existing debate of authentic learning in music education (cf. Green 2008), to which the experience with EarSketch contributes. Freeman et al. (2014) explain that part of the success of the initiative, seconded by informal conversations with teachers who deploy EarSketch and our observations in the classroom, is that working with EarSketch has an authentic[4] character: it is personally meaningful (i.e. students can create their own music using different electronic music genres, such as hip hop, dubstep, or techno), as well as professionally relevant (i.e. students learn modern programming languages, and how to operate DAWs). The

[1] http://nsf.gov/csforall (accessed February 18, 2016).
[2] http://earsketch.gatech.edu (accessed February 18, 2016).
[3] *Computational music remixing* refers here to program a different version of a song by combining existing sound samples.
[4] An *authentic* activity or practice refers here to an action or set of actions that connect to a real-world situation.

EarSketch curriculum aligns with the Advanced Placement (AP) Computer Science Principles (CSP) curriculum framework.[5] A highlighted aspect of the AP CSP curriculum framework is to support collaboration during CS learning.

This paper aims to explore the nature of collaboration using EarSketch in educational settings, and to point to future directions, with no intention to assess the musical outcome at this stage of research. *Live coding* practices are based on the use of scripting languages for real-time music improvisation (Brown 2006; Collins et al. 2007; Freeman and Van Troyer 2011; Rohrhuber et al. 2007). The potential of live coding as an educational tool with an impact on both musical and computational learning has been discussed in Freeman and Magerko (2016) using EarSketch. With live coding, the modification and execution of the code, or composing and performing, are produced as simultaneous activities in real time. As a follow-up of Freeman and Magerko (2016), this paper speculates the potential of collaborative live coding in educational environments, particularly from a performative point of view. The aim is to investigate how to better support collaborative practices using computers in the classroom. In the next section, we overview musical practices of live coding and collaboration, as well as live coding practices in CS education, that inform this work.

Background

Collaborative Live Coding

Collaborative live coding is an umbrella term that refers to performing live coding in group, which includes from small to larger groups. Barbosa (2003) proposed a classification space for computer-supported collaborative music based on the user's location (co-located vs remote) and group's interaction time (synchronous vs asynchronous). This research is interested in a classroom setting in which groups are located in the same space, interacting at the same time, working either on individual or shared interfaces. Next, we provide illustrative examples, as opposed to all-inclusive, of approaches to co-located collaborative live coding.

Small group collaboration using individual screens, and working as a network with a shared clock is reported by McLean (2015). A common collaborative live coding configuration is working in pairs. 2ˆn[6] is a project started in 2012 by the electronic music duo Pulso (Gerard Roma and Anna Xambó). The project uses a highly constrained environment written in SuperCollider, which is a real-time audio synthesis environment and programming language (McCartney 2002). The project is inspired by the ixilang SuperCollider environment (Magnusson 2011). The setup consists of two laptops, each running a shared customized patch that shows two synchronized code editors, one for each of the performers (see Fig. 1).

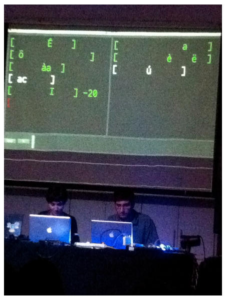


Figure 1: Pulso's 2ˆn presented in Live Coding Sessions at Niu, Barcelona (March 13, 2012)

[5] https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-curriculum-framework.pdf (accessed February 18, 2016).
[6] https://soundcloud.com/pulso-2-n (accessed February 18, 2016).

There exist a number of laptop ensembles and orchestras all over the world, of which some of them explore egalitarian approaches of the ensemble, whilst others explore more hierarchical structures.[7] Collaborative live coding in large groups, under democratic settings, is investigated by the Republic (de Campo 2014), a project that started in 2003 using laptops and their built-in speakers. The Republic is based on an available extension library written in SuperCollider. The Republic's principle is to create a symmetrical and multi-directional network, in which each player can access and modify each other's code (Rohrhuber et al. 2007). The two ensembles powerbooks_unplugged and Republic 111 use the Republic's ideas and technology. The recent advent of the Web Audio API,[8] in JavaScript, has enabled a number of projects to explore the possibilities of participatory audience with their mobile devices using their browser, and connected to a network. With a few exceptions, such as the UrMus live coding language for mobile phones used extensively by the Michigan Mobile Phone Ensemble (Essl 2010), there is little research on live coding on mobile devices, particularly participatory live coding, which seems an interesting direction to computationally support collaboration among large groups.

Live Coding in Learning Environments

Interdisciplinary experiences between CS and music have been explored in the classroom. For example, Scratch[9] is a blocks-based programming language, which can be used to create a variety of multimedia applications, such as games. Scratch has been particularly used in music-related projects, for example, for teaching computational thinking through music in the classroom (Ruthmann et al. 2010). Accordingly, computational thinking in Scratch includes concepts such as synchronization, looping, initialization, use of variables, changing variables algorithmically, modularization, and event processing. Live coding comes into play when using infinite loops and algorithmic composition (cf. Edwards 2011). Changes can be done in real time with no disruption to the musical outcome, a key feature of live coding. The Scratch courses on computational thinking through real time music are called Performamatics (Ruthmann et al. 2010), in which the understanding of computational and musical concepts are tightly linked to the nature of a musical live coding event.

Another example is Sonic Pi,[10] an open source live coding environment for also teaching computing concepts through music, which runs on the tiny computer Raspberry Pi (Aaron and Blackwell 2013). Sonic Pi is based on imperative commands, built on top of the Ruby and SuperCollider languages. Educational collaborative exercises using Sonic Pi are reported by Aaron and Blackwell (2013). For example, students enact imperative programming by triggering sounds in turns. To our knowledge, platforms for live coding in learning environments have been designed and researched focusing on individual use. This paper is an initial step to fill the gap between collaboration as a key component in CS learning and live coding in education.

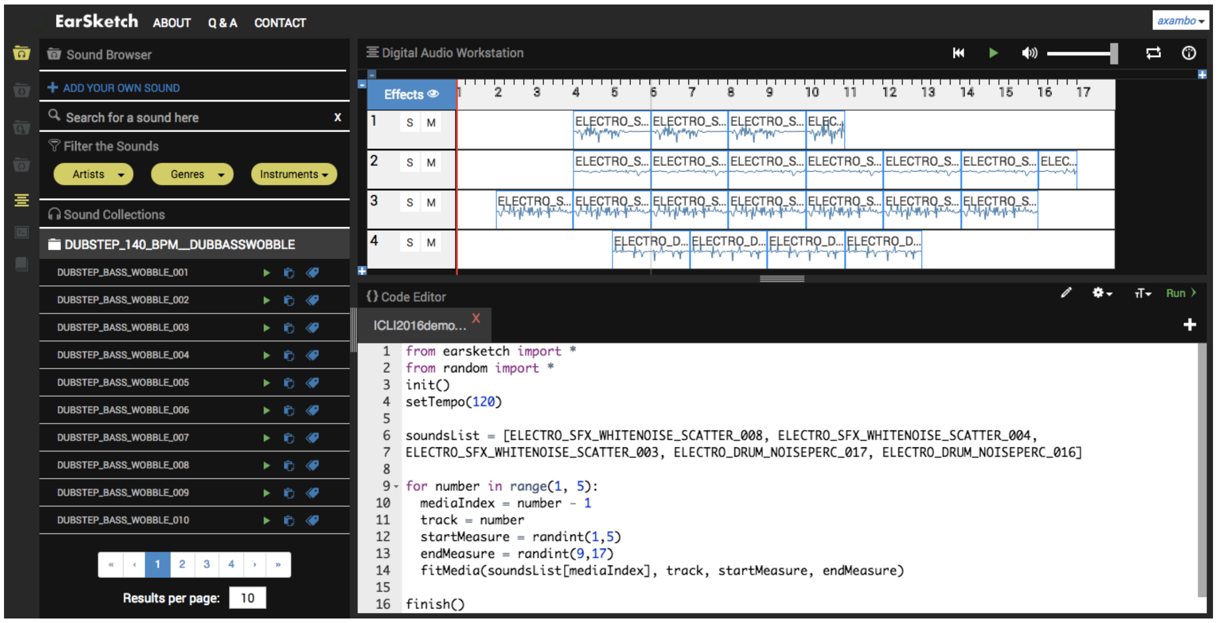
EarSketch: Live Coding Features

EarSketch is a free browser-based DAW environment built with Web Audio API that allows students to create their own algorithmic compositions and computational music remixes while learning computing principles (Mahadevan et al. 2015). The EarSketch API is built on top of JavaScript and Python so that EarSketch can be easy to use by students. Users can work either with their own uploaded sounds, or with 4000 available music samples that were designed by Richard Devine, who is an Atlanta electronic music composer, and Young Guru, who is Jay-Z's audio engineer and tour DJ. EarSketch supports CSP through promoting creative practice and self-expression linked to computational learning; allowing for the use of computational concepts, such as layers of abstraction (e.g. the DAW timeline), loops, variables, different data types, and procedural algorithms; providing the Internet medium as both an individual and a collaborative space in which songs and scripts can be shared; and allowing for social impact of computing practices.

In the DAW view, users can play the music they created in code, and perform basic transport and mixing operations. The common workflow is to write lines of code in the code editor; execute the code by pressing the run button, which

[7] http://www.ialo.org/doku.php (accessed February 18, 2016).
[8] https://www.w3.org/TR/webaudio (accessed February 18, 2016).
[9] https://scratch.mit.edu (accessed February 18, 2016).
[10] http://sonic-pi.net (accessed February 18, 2016).

renders the tracks on the DAW view; and press the play button to listen to the results (see Fig. 2). Users are not able to edit the multi-track audio and effects content directly, instead they must edit the code to change the music. Some of the DAW features of EarSketch can be used within a live coding context (Freeman and Magerko 2016). For example, the looping playback feature in the transport controls allows for changing the code in real time with no disruption to the musical output. Loop points is another live coding feature, which allows students to set shorter ranges within the song with start and end points. Soloing and muting tracks can also be combined with the previous features to create more richness and variation of the musical outcome. However, a DAW interface, such as EarSketch, is generally designed for individual use. In the next section, we report how EarSketch is used in the classroom as part of collaborative practices, and how EarSketch teachers envision it could be improved for supporting collaboration in educational settings.



```
from earsketch import *
from random import *
init()
setTempo(120)

soundsList = [ELECTRO_SFX_WHITENOISE_SCATTER_008, ELECTRO_SFX_WHITENOISE_SCATTER_004,
ELECTRO_SFX_WHITENOISE_SCATTER_003, ELECTRO_DRUM_NOISEPERC_017, ELECTRO_DRUM_NOISEPERC_016]

for number in range(1, 5):
  mediaIndex = number - 1
  track = number
  startMeasure = randint(1,5)
  endMeasure = randint(9,17)
  fitMedia(soundsList[mediaIndex], track, startMeasure, endMeasure)

finish()
```

Figure 2: EarSketch with a Python script that includes a looping function and a random function

Collaboration in the Classroom with EarSketch

Here we present relevant topics that emerged during informal conversations around EarSketch and collaboration that evidences the need of computer-supported educational environments for collaborative live coding. We had casual interviews with some teachers of EarSketch, of which some of them had teaching experience, whilst some of them had training experience. We were interested in getting a sense of current practices in the classroom and potential needs for improving communication between students, and communication between students and teacher. We also visited three schools and conducted in-the-wild and unobtrusive observations by taking notes.

We identified a number of practices for sharing code: for example, Google Docs is used for sharing the EarSketch code between students and the teacher; some students share between them the development process (e.g. techniques, composition strategies), or the musical outcome; some students share their code with other students, and Learning Management Systems are used by some students to post their code and comment on it. These examples illustrate asynchronous sharing practices occurring after completing and saving individual scripts. As potential needs for sharing code, it was highlighted that sharing a script file, like with a Google Doc, would be useful because multiple people can edit the same document at the same time and access from different computers; and some variation of Google Docs with EarSketch would be a huge improvement as Google Docs is used in other CS subjects for group projects. This points to the need of real-time collaboration support in the classroom, during the process of coding and music making. However, some issues were raised, such as to what extent sharing code can go against CSP: teachers need to make sure the work's authorship, which relates to how to grade a collaborative piece; and also sharing would require different levels of permissions.

We recognized a number of practices around collaboration in the classroom: for example, organic and informal collaboration occurs; students naturally team up; groups work in pairs; and groups of size three with at least one musician have been observed, which points to small, casual and interdisciplinary teams. It was also reported that students prefer to be independent and collaborative, which indicates that keeping individual terminals for each student during collaboration seems useful. Activities around collaboration include: informal presentations in which students share their content, explain what they do, and receive feedback from the teacher or other students; sessions in which everyone plays their piece as a performance and the other students give their opinions openly; gallery walks, in which the classmates visit and listen to others' work from one side of the class, and then they switch side; pair programming activities in which students alternate roles between driver and navigator; competitions of songs using a voting system; and collaborations in alignment with the AP CSP curriculum framework. In particular, the group activities that occur during the process of coding, such as in pair programming, are based on turn-taking teamwork. As potential collaborative features, we identified the need of more real-time collaboration support, including the ability to chat to each other; the ability to modify the same script from different terminals by multiple students; and an environment that supports freedom of choice during teamwork (as opposed to constrained roles).

We also identified the nature of the teaching environment and teaching practices, including: a casual environment; students talking to each other before they ask to the teacher; and the promotion of interaction with the environment almost every moment according to the AP CSP curriculum framework. As potential features for supporting teaching practices, we identified that it should be easier to see the code of each other, and to make comments from a small team (teacher, and a few peers) so that students could receive meaningful feedback. This indicates the importance to support collaboration and peer learning in a CS class within informal contexts, in particular sharing and commenting code in real time as part of the learning process. Collaborative coding as a musical performance event can promote peer-to-peer learning within real-world situations, as discussed next.

Approaches to and Challenges of Collaborative Live Coding

In this section, we discuss potential group configurations suitable for collaborative live coding based on the notion of communities of practice; three promising approaches to real-time collaboration based on working in either pairs or larger groups using the same script; and final remarks on how we can best support collaborative live coding in the classroom.

Peer-to-Peer Communities of Practice

In situated learning, knowledge is shared and co-constructed within a context and in a community of practice (CoP), understood as a group that shares an activity (Lave and Wenger 1991), yet this literature has typically focused on examples of beginners learning from experts. *Collaborative learning*, understood as a situation in which more than one person learn or try to learn together (Dillenbourg 1999), has been a frequently studied aspect of computer-supported collaborative work (CSCW) systems (Dillenbourg 1999). These concepts allows us to understand the action as a situated collaborative activity in which the knowledge is transferred by doing in a group and mediated by technologies. In such learning, people gradually construct a shared, convergent meaning, which is situated (Roschelle 1992; Suchman 1987), and which depends on the people involved and the particular technology used. Green (2008) identifies *group learning* through music-making in informal learning practices within music education, in which students learn through watching, imitating and listening to each other. The above ideas resonate with the notion of collaborative live coding, in which students can learn from viewing and interacting with other students' practices, as opposed to only from the teacher, and taking advantage of real-time collaboration features. Potential configurations and roles in the classroom can include working in interdisciplinary pairs or small groups by combining levels of expertise (e.g. beginners and experts), or domains of expertise (e.g. coders and musicians).

Approaches to Real-Time Collaboration

Live *Pair Programming* Approach

This approach is built upon the existing practice of pair programming in the classroom observed with EarSketch. Combining pair programming with live coding is highlighted in Freeman and Magerko (2016) as a form of collaboration where each student can add, in turns, a layer of complexity, computationally and musically, to an ongoing loop. The periodic switching roles of driver and navigator can be kept, and the changes on the EarSketch script are produced in real time in a performance setting. The screen can be projected, and so the other students can see the code and understand two approaches to computational thinking that are clearly organized in turns. In live coding, the musical outcome changes as part of a process, a process of changing by doing, which connects with process music ideas (Reich 1965), as well as with an iterative coding workflow (Freeman and Magerko 2016). Even though the clarity on who is doing what, turn-taking adds constraints to collaborative live coding because it limits the flexibility of modifying a script synchronously as in the case of pair live coding (which is explained later in this section), due to students are working with only one shared terminal. This approach can be useful in a CS class to strengthen learning of CSP through pair programming within a creative context. The selection of a set of audio samples from two people can also create interesting hybrid music styles, unless a music style is planned beforehand, or the music style is constrained by an assignment, or both students like the same type of music.

Multiple *Live Coding* Approach

This approach is based on the Google Docs approach of allowing for multiple editors, who can work simultaneously on the same file. With EarSketch, a planned future feature is that the script can be shared simultaneously, everyone can modify it, and the musical outcome results from a sum of individual cursors. In an educational context, this approach can be applied to teach democratic approaches to music making in real time, as well as learning to intervene in due course, similar to the cycle of listening and playing in group musical improvisation. Computational concepts can be learned using a more interactive approach than just watching a projected screen of other's code as in the case of live pair programming, where the knowledge can be spread throughout the whole class, increasing the information flow within the class. This approach can also become chaotic, but nothing prevents the teacher and the students to decide for certain roles and rules. For example, an interesting avenue to explore is a conversation with the teacher as a live coder showing a computational and musical concept, with a script that can be shared and occasionally modified by students, either by replying a teacher's question, or by writing questions or suggestions as comments that the teacher or other students can address, or by modifying code in an organized form. Another example is exploring interactive programming in group, where there can be a division of labor of creating different functions for different parts of the song and test them as they are written, in real time. As a potential challenge, it is unclear how to know when to re-excute the code in a state that has no syntax or logical errors when multiple people are editing the same code document. Another challenge is, if students want to create a symmetrical network as in the Republic's example, identifying how to know what is each other's code, as well as how to know who has modified it, assuming that it matters to know about code authorship.

Pair *Live Coding* Approach

This approach is an instance of the multiple live coding approach, but constrained to two students. It is also based on duos as in pair programming working on a single EarSketch script, but each student has an individual terminal so both can be drivers who contribute to the script simultaneously. In an educational context, this approach can be useful for simultaneous sharing of computational and musical strategies on the same script, as well as for team solving of computational errors. This approach's flexibility for real-time group work is in detriment of knowing who is doing what beyond the pair of students, for example, when projecting their script. An interesting challenge is working with two simultaneous scripts in EarSketch, as in Pulso's 2ˆn, that would require both to be synchronized.

Live Coding Suitable for Learnability and Collaboration

According to Ruthmann et al. (2010), a suitable environment for live coding in the classroom needs to allow for the exploration of real-time manipulation of code, in which both musical and computational understanding are important. Furthermore, the environment needs to be easy to use (e.g. EarSketch, Scratch, Sonic Pi), including domain-specific languages that are easy to understand (Aaron and Blackwell 2013). We argue here that the benefits of collaborative live coding in the classroom include that computational thinking moves from an individual to a social plane, in which problems and progress are shared as part of the learning process, in real-time and real-life situations. This approach includes existing practices observed in the classroom, such as showcasing students' work, or discussing the code in group. It also spans to potential new practices, such as performing in pairs of live coders, or manipulating in real time the code as part of a participatory, multi-directional musical piece involving the whole class. An open question is whether the sessions should be recorded to reflect on them pedagogically. Another open question is how collaborative live coding would be screencasted and understood in certain live coding educational contexts, such as Livecoding.tv,[11] a livestream platform that generally focuses on learning from watching individual live coders. Future work includes assessing the musical outcome of collaborative live coding in the classroom, as well as designing a set of exercises based on this practice.

Conclusion and Future Work

This paper discussed the challenges and potential of collaborative live coding in the classroom. We presented existing collaborative practices in live coding, and live coding environments in education. Drawn from the literature review and empirical work, we discussed three challenging and potential approaches to collaborative live coding using EarSketch, as part of peer-to-peer communities of practice. Finally, we proposed how to best support collaborative live coding in educational settings. As future work, we plan to assess the suggested approaches to collaborative live coding in the classroom.[12]

Acknowledgements. We are thankful to the teachers who participated in the interviews. The EarSketch project receives funding from the National Science Foundation (CNS #1138469, DRL #1417835, and DUE #1504293), the Scott Hudgens Family Foundation, and the Arthur M. Blank Family Foundation.

References

Aaron, Samuel, and Alan F. Blackwell. 2013, September. "From Sonic Pi to Overtone: Creative Musical Experiences with Domain-Specific and Functional Languages." *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design*. New York, USA: ACM, 35–46.

Barbosa, Álvaro. 2003. "Displaced Soundscapes: A Survey of Network Systems for Music and Sonic Art Creation." *Leonardo Music Journal* 13:53–59.

Brown, Andrew R. 2006. "Code Jamming." M/C Journal 9, no. 6.

Burge, Jamika D., and Tiki L. Suarez. 2005, October. "Preliminary Analysis of Factors Affecting Women and African Americans in the Computing Sciences." Edited by Pamela J. Williams and Mark A. Friedman, *Proceedings of the 2005 Conference on Diversity in Computing.* New York, USA: ACM, 53–56.

Collins, Nick, Alex McLean, Julian Rohrhuber, and Adrian Ward. 2007. "Live Coding Techniques for Laptop Performance." *Organised Sound* 8 (3): 321–330.

[11] https://www.livecoding.tv (accessed February 18, 2016).
[12] Proof of concept: https://vimeo.com/167653232 (accessed May 22, 2016).

de Campo, Alberto. 2014, September. “Republic: Collaborative Live Coding 2003–2013.” Edited by Alan Blackwell, Alex McLean, James Noble, and Julian Rohrhuber, *Collaboration and Learning through Live Coding (Dagstuhl Seminar 13382).* Dagstuhl, Germany, 152–153.

Dillenbourg, Pierre. 1999. “What Do You Mean by ’Collaborative Learning’?” In *Collaborative Learning: Cognitive and Computational Approaches*, edited by P. Dillenbourg, Volume 1, 1–19. Oxford, UK: Elsevier.

Edwards, Michael. 2011. “Algorithmic Composition: Computational Thinking in Music.” *Communications of the ACM* 54 (7): 58–67.

Essl, Georg. 2010, June. “The Mobile Phone Ensemble as Classroom.” Proceedings of the 2010 International Computer Music Conference. New York: *International Computer Music Association*, 506–509.

Freeman, Jason, and Brian Magerko. 2016. “Iterative Composition, Coding and Pedagogy: A Case Study in Live Coding with EarSketch.” *Journal of Music Teacher Education* 9 (1): 37–54.

Freeman, Jason, Brian Magerko, Tom McKlin, Mike Reilly, Justin Permar, Cameron Summers, and Eric Fruchter. 2014, March. “Engaging Underrepresented Groups in High School Introductory Computing through Computational Remixing with EarSketch.” Edited by J.D. Dougherty, Kris Nagel, Adrienne Decker, and Kurt Eiselt, *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. New York: ACM, 85–90.

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

Green, Lucy. 2008. *Music, Informal Learning and the School: A New Classroom Pedagogy.* Hampshire, UK: Ashgate Publishing Limited.

Lave, Jean, and Etienne Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge, UK: Cambridge University Press.

Magnusson, Thor. 2011, July–August. “ixi lang: A SuperCollider Parasite for Live Coding.” Edited by Monty Adkins and Ben Isaacs, *Proceedings of the 2011 International Computer Music Conference*. San Francisco: International Computer Music Association, 503–506.

Mahadevan, Anand, Jason Freeman, Brian Magerko, and Juan Carlos Martinez. 2015. “EarSketch: Teaching Computational Music Remixing in an Online Web Audio Based Learning Environment.” *Proceedings of the Web Audio Conference.*

McCartney, James. 2002. “Rethinking the Computer Music Language: SuperCollider.” *Computer Music Journal* 26 (4): 61–68.

McLean, Alex. 2015. “Reflections on Live Coding Collaboration.” Skyler and Bliss Original Citation Adkins, Monty and Segretier, Laurent (2015) Skyler and Bliss. In: xCoAx 2015: *Proceedings of the Third Conference on Computation, Communication, Aesthetics and X*. Universidade do Porto, Porto, pp. 213–220.

Reich, Steve. 1965. “Music as a Gradual Process.” *Writings on Music*, 1965–2000, pp. 34–36.

Rohrhuber, Julian, Alberto de Campo, Renate Wieser, Jan-Kees van Kampen, Echo Ho, and Hannes Hölzl. 2007, September. “Purloined Letters and Distributed Persons.” *Music in the Global Village Conference*.

Roschelle, Jeremy. 1992. “Learning by Collaborating: Convergent Conceptual Change.” *The Journal of the Learning Sciences* 2 (3): 235–276.

Ruthmann, Alex, Jesse M. Heines, Gena R. Greher, Paul Laidler, and Charles Saulters II. 2010, March. “Teaching Computational Thinking through Musical Live Coding in Scratch.” Edited by Gary Lewandowski, Steven Wolfman, Thomas J. Cortina, and Ellen L. Walker, *Proceedings of the 41st ACM Technical Symposium on Computer Science Education.* 351–355.

Suchman, Lucy. 1987. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge, UK: Cambridge University Press.