Cooperative Live Coding of Electronic Music with Troop

Ryan Kirkbride
University of Leeds
sc10rpk@leeds.ac.uk

Abstract. Live coding is the process of generating audio or visuals using algorithms that are written, and re-written, live in front of an audience. In a typical performance the live coder will project their screen and share their code and creative processes with the audience. When playing together, a group of live coders will often share resources over a network, such as a tempo clock or lines of code, but rarely do they work together on the same material concurrently. Troop is an interactive text editor that allows multiple users to edit a text buffer simultaneously, evaluate portions of code, and create a shared and cooperative musical experience.

Collaborative Live Coding Environments

An ensemble of live coders work on separate laptops but share resources over a network, such as timing systems, to better coordinate their performance. This can be quite an arduous task when done manually and many groups will opt to use a program that specifically facilitates collaboration and synchronization. Such tools tend not to be programming languages themselves but software that has been developed to improve the communication between multiple computers that are using an existing Live Coding language. A good example of this is the browser-based system EXTRAMUROS, which allocates each connected performer a small text box on a web page into which they can each write code (Ogborn, 2014). These text boxes are visible and can be edited by any other connected performer during a performance. EXTRAMUROS is “language-neutral”, which means it can
be used with any language that has an interpreter that allows commands to be “piped” into it. This improves the accessibility of cooperative performance to a wider range of Live Coding practitioners. Another browser-based tool for collaborative Live Coding is GABBER, which is an extension to the live coding library, GIBBER (Roberts et al, 2015). It combines a chat-room interface with shared text buffers similar to EXTRAMUROS, but the code in each user’s text buffer is only executed on that user’s machine. Rohrhuber et al. (2007) developed an interface within the SuperCollider environment similar to that of a chat-room that allows performers to share small blocks of code called “codelets”. In contrast to EXTRAMUROS and GABBER, the “codelets” are shared with, but not executed on, each connected machine. Performers either use the “codelets” or modify them and re-submit them to the chat-room interface. Rohrhuber has since gone on to develop another popular SuperCollider extension called THE REPUBLIC that allows performers to access and modify each other’s code without the chat-room style interface (de Campo and Rohrhuber, 2011). The “textual performance environment” LOLC also focusses on a conversational style of communication by allowing performers to share shorthand musical patterns, which are then played or transformed and re-shared by other performers, aiming to facilitate methods of practice common to both improvisation and composition (Freeman and Van Troyer, 2011). As opposed to sending text between performers, IMPROMPTU SPACES uses a tuple-space that acts as a “remote bulletin board” for posting and retrieving information across a network (Sorenson, 2010). This creates a shared and distributed memory that is accessible to each connected client and allows users to manipulate global variables such as tempo while avoiding any read or write clashes.

**Concurrent Collaboration with Troop**

Multiple Live Coders working on individual portions of code will usually have their work on separate screens. If audience members are unable to see all of the code due to the lack of space or projectors, for example, then this can result in a non-optimal audience experience. The live coding system, EXTRAMUROS, addresses this by allocating each connected performer a text box on a single web page so that the audience can see all the active coding that is occurring and performers can quickly and easily share or modify each other’s work. There is still a degree of separation in the workflow, however, and modifications of other performers’ work have to be requested and accepted as opposed to being an integral part of the shared creative process. Single shared text buffers have seen much mainstream success recently, most notably in Google Docs, and this has prompted me to create a similar tool for concurrent Live Coding collaboration called TROOP.
Where the previously discussed live coding environments allowed users to share portions of code that have already been written, TROOP shares keystroke data between connected clients to build code together. Each user is allocated a different coloured font to differentiate the contributions made so that each user can leave traces of their own coloured code throughout the shared text buffer, which is demonstrated in Figure 1. Editing someone else’s code interweaves their colours and thought processes, creating a temporary visual testament to the cooperative process until the code is changed once more.

Figure 1. A screenshot of the Troop interface with three connected users.

Not only does TROOP allow live coders to work in a more interactive and cooperative way than was previously possible but it also minimises the amount of technical equipment required for performance as only one laptop needs to be connected to a projector and PA system. TROOP is written in Python and designed to work with the Python-based live coding language, FOXDOT, as its interpreter can easily be imported as a standard Python library (Kirkbride, 2016). It now also fully supports TIDALCYCLES, which is a pattern creating language embedded in the Haskell programming language (McLean, 2014). TROOP can also be considered language-neutral as it can be used with any application that accepts text being piped to it through the standard input from the terminal in a similar manner to EXTRAMUROS.

Troop is open source and is available at https://github.com/Qirky/Troop.
Acknowledgments

I’d like to thank Dr. Alex McLean for the conversation that led to the idea for Troop and his continued support of my work and also my PhD supervisors Dr. Luke Windsor, and Prof. Guy Brown.

References