Live Coding Console with Remote Audience into the web

Guilherme Martins Lunhani[1], **Flávio Luiz Schiavoni**[2]

[1]Rua Abolição 403 – 18044-070, Sorocaba, SP

[2]Federal University of São João Del Rei (UFSJ)
São João Del Rei, Minas Gerais, Brazil
lunhanig@gmail.com, fls@ufsj.edu.br

Abstract

Live coding is a (not so) novel form of performance based on computer programming languages. Since the emergence of Web Audio, several initiatives managed to create applications and programming environments on the web for music programming and, why not, live coding with music in web browsers. Even though running on the web, these environments run an individual live coding application without audience capability. This article describes the context, reflections and the development of a live coding web environment and three approaches to create a live coding environment on the web with audience including an initial implementation and some code development. These approaches are conceptual and can be applied on other tools and expand the live coding capability on the web.

1. Introduction

In the context of musical productions made by artist-programmers using the textual programming languages [1], Giovanni Mori [2, p.197] defined the British term *livecoding* as a polyvalent technique of improvisation.

In academic context, the term appeared in 2003 [3], and in 2004, a set of rules was formalized by British artist-programmers[4]. This set of rules do not restrict any language expression. At that time, the artistic artifacts fluctuated between music, the audiovisual and dance.

Many *livecoding* into the web start to emerging from a largely discussion since the W3C added the capability of real time audio processing to web browsers [5]. There are many web applications that can generate, process and analyze audio directly using JavaScript, hiding the basic process into classes and functions.

Under this context, we developed a live DSP console as a single *client-side application*, or in other words, a prototype of a web CLI (Command Line Interface). This work aims to share the current development thoughts and works focusing the creation of a web live coding environment with remote audience.

2. Web live coding environment with remote audience

A basic webaudio application is developed in Javascript and works only in the browser. This approach was used in in most applications existents. A step forward to live coding on the web is to try to create a programming environment where the performance could be listened by an audience like a streaming.

1st approach: Audio streams

Our first approach to a Live coding web environment with audience is to keep a connection with the server and create an audio stream from the application output. This approach could use a stream server, like Icecast or ShoutCast to distribute the live performance audio stream. The advantage of using one of these servers is that the audience could use any application that connects to these stream servers and does not have to use a web browser to listen the live coding performance. On the live coding environment, a small change is necessary adding a stream object on the audio route to make an audio stream to the server.

2nd approach: Audio Stream + code sharing

Our second approach is to send the code developed on the environment with the audio parameters and configuration to the server. The server would run the code locally and send the audio output to audio stream instead of sending it to audio output.

Thus, clients could connect to server audio stream, that also can be an Icecast / Shoutcast

instance, and hear the living performance. The main difference here is a reduced bandwidth between the live Coder and the server.

3rd approach: Code Sharing

Our last approach is to send the code to the server to be shared with the audience. In this approach, the audience computer should run the same application as the live coder and audience computer would synthesize locally the audio from the shared code. This approach uses less bandwidth and allow the audience to interact with the source code and learn how the code was written. The server implementation is close to a white board, a server to share text among different clients. Since the server does not need to run Javascript code, it can be implemented in any serve-side language like PHP or JSP.

3. Approaches comparison

The first comparison among the three approaches is the shared data type. On the first approach, it is shared audio, the second approach shares code between the live coder and the server and audio between the server and audience and the third approach shares only code.

Since the shared data type is different, on the first approach the code is private and the live coder does not need to share it. She second approach is semi private because it is sent to the server and the server could save the code, publish it to the audience during the performance or later. The audience could read the code but a change on the code would not change the local sound since it is not processed locally but received already processed. On the third approach the code is shared and real time published to the audience. The audience could change the code and affect the hearing experience and also re share the code on the web.

The presence of the code can change the hearing experience. Both, first and second approach leads to a passive hearing while the third approach allows the code reading with the listening. Audience can notice programming changes, code errors and typos and it can affect the hearing experience.

Lastly, the client program influences the hearing experience but can simplify the audience listening. First and second approaches can use a stream cast application, like VLC, while the third approach is more restrictive and demands a browser as the client application.

4. Conclusion

This paper presented 3 approaches to create a live coding web environment with audience. These approaches are being developed our work in progress, referring to the works of Pietro Grossi and Max Mathews, considered by these authors as premature live coding cases.

We planned to stream or pipe the audio to a streaming server, by a streaming server channel choose by the user, or by WebRTC, but this feature is incomplete. The presented application is under development[1] and has, among the complete and incomplete features, a server that saves a code and processes it into an audio file. Our current implementation reflects an agreement between the second and third approaches.

It is part of Future works to implement the three approaches here presented to have a better framework to test and validate user's experience in each scenario.

References

[1] Alex McLean. *Artist-Programmers and Programming Languages for the Arts*. PhD thesis, Department of Computing, Goldsmiths, University of London, October 2011.

[2] Giovanni Mori. Analysing live coding with etnographical approach. In *ICLC2015 Proceedings*, pages 117–124, 2015.

[3] Nick Collins, Alex McLean, Julian Rohrhuber, and Adrian Ward. Live coding in laptop performance. *Organised Sound*, 8(3):321–330, 2003.

[4] Adrian Ward, Julian Rohrhuber, Fredrik Olofson, Alex Mclean, Dave Griffiths, Nick Collins, and Amy Alexander. Live algorithm programming and temporary organization for its promotion, 2004.

[5] Lonce Wyse and Srikumar Subramanian. The viability of the web browser as a computer music platform. *Computer Music Journal*, 37(4):10–23, 2013.

[1] Available on: `https://github.com/lunhg/termpot`.