Streaming objects and strings

André Damião
University of São Paulo
São Paulo,
Brazil
andredamiao@usp.br

Flávio Luiz Schiavoni
Federal University of São João Del Rei
Minas Gerais,
Brazil,
fls@ime.usp.br

Abstract

In this paper we outline the development of a tool for live coding which enable users to share Pure Data patches in real time. The development outcome is a piece called Tanto that mix live coding with shared code in an audiovisual performance/installation.

Keywords

Live Coding, Network Music, Pure Data, BYOB

1 Introduction

Since the computer became a common tool to musicians, computer networks became an alternative to cooperation and collaboration in music. Initiatives in music interaction over computer networks has included network music performances [Lazzaro and Wawrzynek, 2001][Young, 2001], Master classes [Young and Fujinaga, 1999], audio through networks [Carôt and Werner, 2008], distributed ensemble [Hajdu, 2003], instrument mapping [Malloch et al., 2008] and laptop ensembles[Rohrhuber et al., 2007].

Generally these interactions are all based on musical content streaming. OSC, MIDI, audio and video are the kind of content normally shared through computer networks and several tools were developed to do this job. Artistics approaches have been developed since the 1970's with The League of Automatic Music Composers[Rohrhuber, 2007].

A common practice involving computer music is live coding, as code could possibly be the most direct interface to the logic of computers. Thus, sharing code became a common practice, as it could enable a deeper and faster level control of sound and other types of data. Applications such as Republic[1] (PowerBooks Unplugged) and Co-audicle[2] (Wang and Cook) made this paradigm accessible to musicians. This paper introduces a way to share code in a visual programming language (VPL). Our research includes a Pure Data[3][Puckette, 2007] (aka PD) external object developed to realize the task of realtime code sharing. We also introduce the piece Tanto, which was developed with the tools that will be described, and modelled to it's possibilities.

[1]`https://github.com/supercollider-quarks/Republic`

[2]`http://audicle.cs.princeton.edu/`

2 An external to Code sharing in Pure Data

Once we decided to extend PD as a code sharing tool, we took some project decisions to develop it.

Our first development decision was what to share. Since PD Pure Data programming can be seen as a structured languages where subpatchs are like functions we decided to share subpatchs. Thus, it is possible to keep a main patch with local objects and connections and instantiate the object sending and receiving with a well defined code scope.

The second decision was about how to share. We decided to use multicast addressed UDP packets. We chose UDP because it is a little bit faster than TCP [Schiavoni et al., 2013] and Multicasting packets to reach every machine planned to receive the messages without message copying.

We decided to divide the implementation into two main objects: **sendpatch** and **receivepatch**. Both objects accepts network port to send / receive packets configuration. This port configuration can be used to send different data to different computers and brings to the user the possibility to have separated network communication channels.

Our code is based on some PD functions that are not exported on PD main library. The first function is the patch analyser to find a target

[3]`http://puredata.info`

subpatch to send. The second one is the copy / paste PD internal functions that permits one to catch a subpatch source code.

Once the subpatch is found and the code is copied, we marshall it on network packets to send it. Receiver unmarshall the received packet, remove local subpatch if it exists, create a new subpatch and paste the received code on it.

3 TANTO: BYOB+LiveCoding/Network

TANTO is a network live coding audio visual piece inspired by the visual aesthetics of Bring your Own Beamer (BYOB) events. The elements which define TANTO are the characteristics of the objects that we developed (sendpatch and receivepatch), a minimum of four computers, each computer needs a sound and visual output of any type, and the composition made of screens and projections where the code can be seen. All other elements are open. It could be played by one or many performers, which could be in the same space or remote spaces, improvised or notated, presented on a stage or in a installation format and etc. One of the questions behind the development of TANTO was how to mix the visual aesthetics BYOB with the paradigms of live coding and network music. BYOB became a common practice for video art producers, such as Rafaël Rozendaal, in which a group of people bring their own beamers and show their material simultaneously with dozens of other works, generating a vibrant and chaotic landscape, in which there is also a strong aspect of community. Visually it resembles the painting exhibitions from the late XIX century, where numerous pictures were accumulated on the walls, and the photography exhibitions by Wolfgang Tillmann, in which larger landscapes are formed by individual pictures.

Visualisation is central to live coding[McLean et al., 2010], the use of projection in live coding is prevailing in performances, as an extension of the instrument. The appropriation of the BYOB paradigm is a form of creating a "visual polyphony", which can be understood syntactically, by reading the code, and as visual music, because as the software does not just send the content of each subpatch, but the window position and size, it is possible to create movements and different settings of brightness for each projection.

As it was mentioned before the audio may be reproduced by any sort of output: laptop speakers, studio monitors, guitar amplifiers, headphones, etc. Each computer should be close to it's sound source, so it would possible to create a clear connection between sound, image and code. The disposition of the equipment in space may vary considering the situation in which TANTO is presented, but if there are sound sources that are very still, too contrasting in volume, or even with use of headphones, it would be recommended that the audience could circulate around the work during performance. The suggestion of using different types of speakers was influenced by the original Acousmonium, in which there was a great variety of speakers, and composers could create different timbres through the spacialization of mono or stereo files. We believe that a live coder could do the same using this configuration. The piece was created concomitantly with the development of the external objects, what enabled us to adapt the performance and the code to necessities and problems that appeared during the process.

4 Conclusion

Sharing code can have much more potential than sharing other types of data related to sound. Thus, this process contributes to the creation of more complex and permeable algorithmic music practices. Implementing this tool in a visual programming environment gives a wider access to artists and musicians. We understand that the objects sendpatch and receivepatch could be used not only for artistic purposes but also for educational and collaborative code development situations, making the process of exchanging patches much quicker. Concerning the performance TANTO, we think that the blend between live coding sessions and BYOB events can lead to very amusing code jams.

The developed PD library is an open source project available on the project website[4].

References

Alexander Carôt and Christian Werner. 2008. Distributed network music workshop with soundjack. In *In proceedings of the Tonmeistertagung*, Leipzig, Germany.

Georg Hajdu. 2003. Quintet.net - a quintet on the internet. In *Proceedings of International Computer Music Conference*, page 315–318, Singapore.

[4]`http://sourceforge.net/p/pdsendpatch/`

John Lazzaro and John Wawrzynek. 2001. A case for network musical performance. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '01, pages 157–166, New York, NY, USA. ACM.

Joseph Malloch, Stephen Sinclair, and Marcelo M. Wanderley. 2008. Computer music modeling and retrieval. sense of sounds. chapter A Network-Based Framework for Collaborative Development and Performance of Digital Musical Instruments, pages 401–425. Springer-Verlag, Berlin, Heidelberg.

Alex McLean, Dave Griffiths, Nick Collins, and Geraint Wiggins. 2010. Visualisation of live code. In *Proceedings of the 2010 International Conference on Electronic Visualisation and the Arts*, EVA'10, pages 26–30, Swinton, UK, UK. British Computer Society.

Miller Puckette. 2007. *The Theory and Technique of Electronic Music*. World Scientific Publishing Company, Incorporated, Hackensack, N.J.

Julian Rohrhuber, Alberto de Campo, Renate Wieser, Jan-Kees van Kampen, and Hannes Ho, Echo Hoand Hölzl. 2007. *Music in the Global Village: International Conference on Network Music Composition and Performance*. Budapest.

Julian Rohrhuber. 2007. *Network Music*. Cambridge University Press Cambridge ; New York.

Flávio Luiz Schiavoni, Marcelo Queiroz, and Marcelo Wanderley. 2013. Alternatives in network transport protocols for audio streaming applications. In *Proceedings of the International Computer Music Conference*, Perth, Australia.

John Young and Ichiro Fujinaga. 1999. Piano master classes via the internet. In *In Proceedings of International Computer Music Conference*, pages 519–522, Beijing, China.

John P. Young. 2001. Using the Web for live interactive music. In *Proceedings of International Computer Music Conference*, pages 302–305, Habana, Cuba.