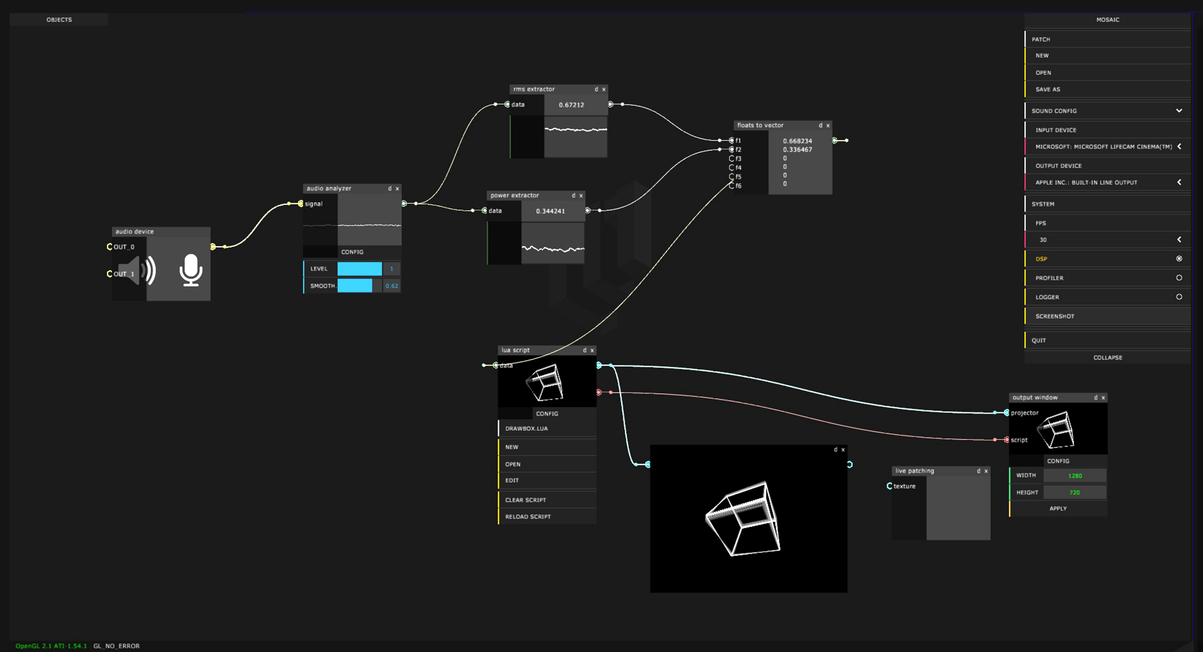
M O S A I C

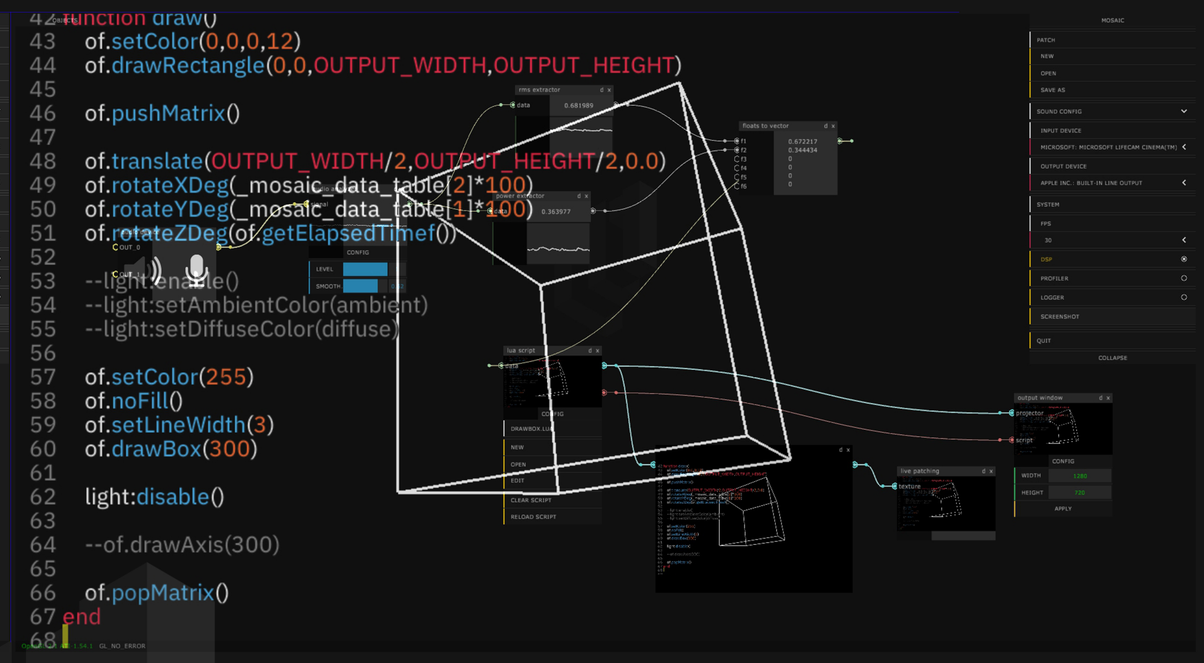
AN OPENFRAMEWORKS BASED VISUAL PATCHING CREATIVE-CODING PLATFORM

Emanuele Mazza & María José Martínez de Pisón, Laboratorio de Luz, Universitat Politècnica de València

Mosaic is an open source multiplatform (osx, linux, windows) live coding and visual programming application based on openFrameworks. This paper describes the initial ideas that have driven its development, its internal structure and the basic libraries it is comprised of, the levels of complexity that can be developed with Mosaic, (from beginner, to very complex developments) and the main contributions to the field of live coding/visual programming and also to the field of creative coding teaching/learning.

We present the development framework, which integrates two paradigms: visual programming (diagram) and live coding (scripting), to show its features and potential, most significantly, the learning feedback generated through to the relationships it establishes between human-machine. In other words, amplifying access routes to that relationship/interrelation promotes augmented-thinking through feedback.

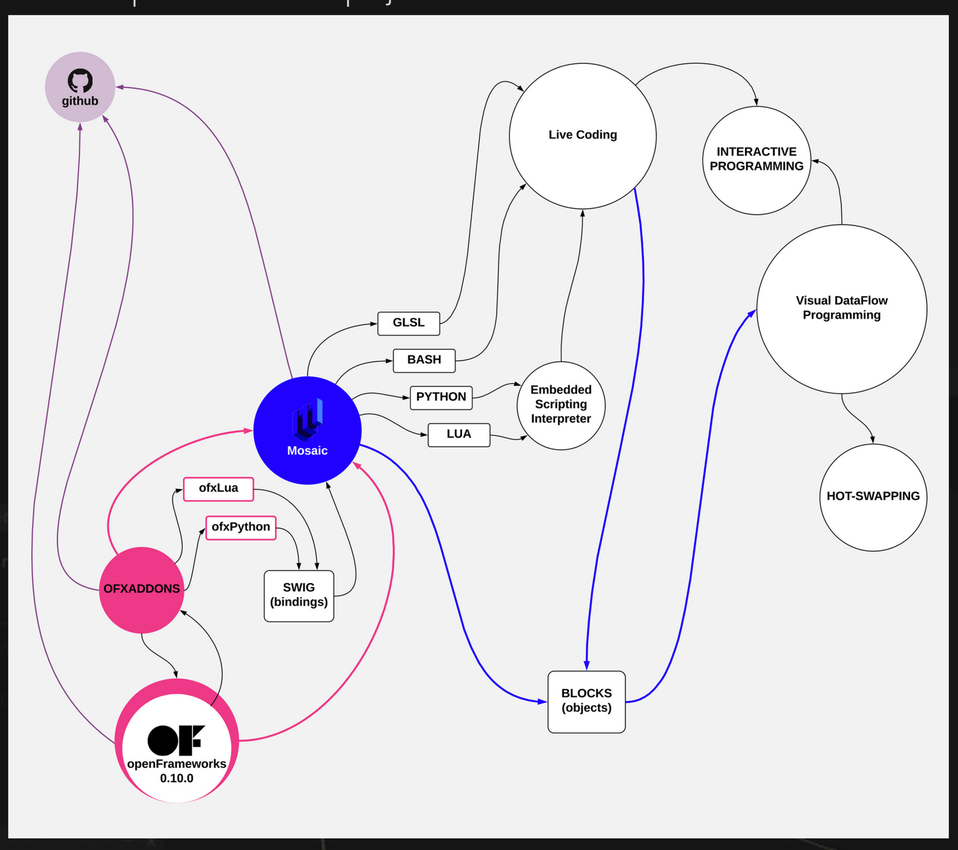




Introduction

Mosaic is a visual programming and live coding environment, based on openFrameworks. In the data flow environment, the possibility of programming in different scripting languages (Lua, Python, BASH, GLSL) is inserted through specific objects that can be activated simultaneously. This hybridization depicts a fertile two-way cross-pollination situation that amplifies the human-machine communication, providing something similar as the benefits introduced by the Dual Coding Theory (Paivio 1990, 57).

The project, at alpha stage right now, is being developed through a modular structure of independent ofxAddons. For example, ofxVisualProgramming is the central code for visual programming, and is kept isolated to encourage contributions, simplify error correction and improve the quality of the code, but most of all to encourage a collaborative community-driven development of Mosaic project.



Goals

- To submit a new development application in the field of creative programming, hybridizing live-coding and visual-programming paradigms.

- To facilitate learning and use of programming environments, promoting a human-machine interrelation that encourages augmented-thought through the learning feedback that this application provides.

- To make the software and the related documentation more widely known to promote a community-driven collaborative development.

Context

The general objective of visual programming languages is to make programming more accessible for beginners (Medlock-Walton 2014, 545) (Tanimoto 2013), and to provide advanced users with added support at three different levels:

Syntax, visual programming languages use blocks and a flow communication system (cords), which reduces and/or completely removes the possibility of syntactic errors (a cord cannot connect where it does not touch, the environment prevents the error).

Visual semantics, it is plausible to consider the inclusion of visual compression mechanisms embedded in the blocks themselves (visual feedback of comprehension), or in other words, documentation of the language hybridized in the language itself.

Pragmatic, the structure of visual programming allows the generation of case studies to analyze the interaction systems internal to the process (internal data flow/reaction/interaction) and related to the result (input -> output). Amplification of the relationship/interrelation between human-machine, boosting augmented-thought through feedback.

Today's visual programming environments also incorporate the dataflow programming concept, which translates into adding real-time live-coding capabilities, or in this case live-visual-coding, a programming environment that allows auto-parallelization (Johnston, Hanna, and Millar 2004), a programming concept that increases opportunities for shared development and multiple levels of learning (fragmentation/de-fragmentation of a program in real time)
The fundamental elements of an environment with these characteristics can be derived from the basic concepts that constitute the formulation of Reactive Functional Programming of Continuous Semantics (Elliott and Hudak 1997), which aims to abstract all the operational details not important for the programming itself (simplification of interface/use).
The key points of this formulation are related to the previously introduced dataflow programming, the dynamic modeling of values in real-time (dataflow, the cords system that we call signal transmission system), and to the possibility of controlling/programming reactive events that alter the system structure (program -> code -> visual process).

Mosaic integrates live-coding (Lua, Python, GLSL, Interactive Programming without Hot-swapping) with visual programming and data flow in real time (Fully Interactive programming), combining all its elements to reinforce its native structure of auto-parallelization, leveling out the layers of complexity and equalizing the possible user

Conclusions

We have introduced a new application hybridizing a live coding environment with visual programming, based on openFrameworks. We have explained the characteristics of its design based on independent articulated cores and the contributions it offers to the field of creative programming, teaching and self-learning. We have presented the philosophical and computer concepts on which the concept and the technology of the project is built.

Future work will involve the implementation of a distributed network communication system to work remotely on the same projects files in a collaborative way, as well as creating an on-line community to foster collaboration with the project.

References

Elliott, Conal y Hudak, Paul. 1997. “Functional Reactive Animation”. International Conference on Functional Programing. Amsterdam.
Johnston, Wesley M.; Hanna, J. R. Paul y Millar, Richard J. 2004. “Advances in Dataflow Programming Languages”. ACM Computing Surveys, 36 (1): 1-34.
Medlock-Walton PAUL. 2014. “Blocks-based Programming Languages: Simplifying Programming for Different Audiences with Different Goals”. Proceedings of the 45th ACM technical symposium on Computer science education. SIGCSE ’14. New York: ACM
Paivio, Allan. 1990. Mental Representations: A Dual Coding Approach. Oxford University Press