FlowLissajous - An environment for *meta* live code performance

Elizabeth Wilson & Andrew Thompson

Keywords: Meta-programming, Web Audio API, programming environment paradigms

Live coding languages operate by constructing and reconstructing a program designed to create sound. These languages often have domain-specific affordances for sequencing changes over time.

When examining existing languages that support temporal sequencing we have identified **two key limitations** of their ability to sequence changes over time:

1) Non-general Sequencing: These languages only allow a ***subset*** of their features or properties to be sequenced in time.

2) Lack of Shorthand: Arbitrary properties *can* be sequenced in time but ***lack powerful shorthand notation*** provided to other properties.

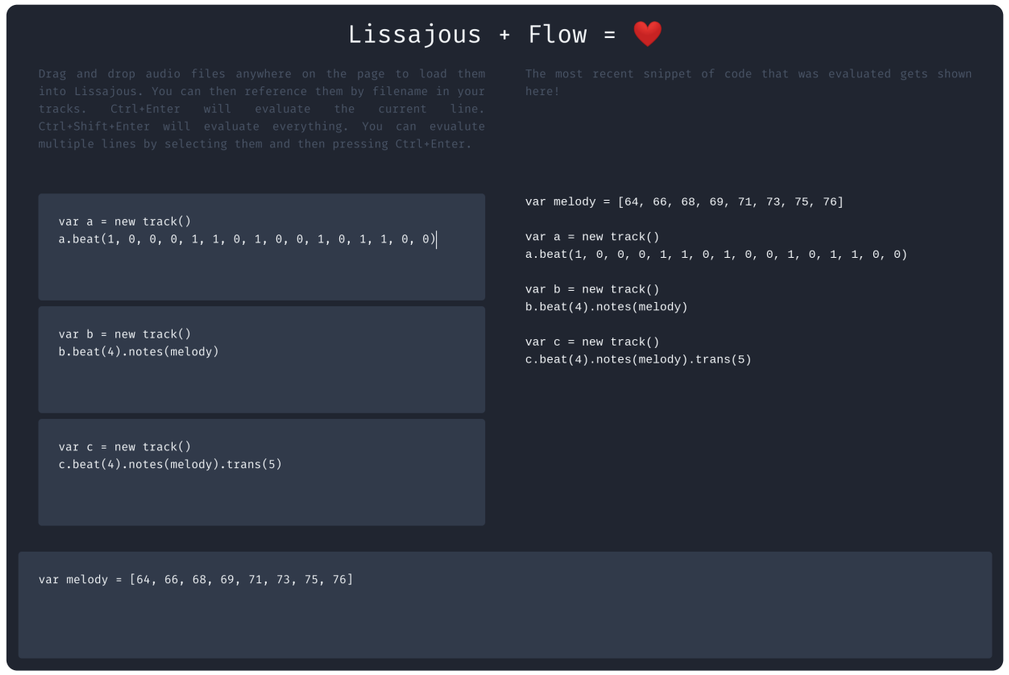
In a more broad review of music programming languages, Dannenberg notes that adding general-purpose programming features to a music programming language greatly benefits it[1]. Similarly, McLean identifies that live coding languages often focus on brevity and syntactic shorthand to describe complex ideas quickly and concisely[2].

Live coding languages based on **Lisp** exemplify these two ideas with powerful metaprogramming constructs built into the language. Similarly, the **ChucK** language treats time as a first-class data type allowing arbitrary computations to run in, and manipulate, time.

Scan the QR code with your mobile's camera to access the full paper.

Additionally, a prototype of the system can be found here: **b.link/flow-lissajous**

[1] Dannenberg, R. B. (2018) Languages for Computer Music, in Frontiers in Digital Humanities
[2] Petrol: Reactive Pattern Language for Improvised Music, in Proceedings of the 2010 International Computer Music Conference



We present a novel new approach by introducing a live coding environment with the ability to sequence the ***input source code*** itself. We acheive this while also making only minimal modifications to the host language, **Lissajous**.

Individual buffers of code may be set programmatically set to record. While recording, these buffers store their contents whenever they are evaluated. These can then be played back using standard Lissajous track methods such as **`Track.beat()`**.

```
var buf = new track()
var a = new track()

buf.record()

// evaluate each line separately
a.beat(4).notes(64, 66, 68)
document.body.style.backgroundColor = 'red'
a.beat(3).notes(71, 68, 71, 73)
document.body.style.backgroundColor = 'blue'

buf.stop().beat(16)
```

Above, each line of code between **`buf.record()`** and **`buf.stop()`** is evaluated at one bar intervals (16 16th). Although making use of Lissajous' sequencing abilities, the code is evaluated ***as if the user had typed it*** making it possible to sequence changes such as modifying the background colour when it was not possible to do so before.

MEDIA & ARTS TECHNOLOGY

EPSRC+AHRC Centre for Doctoral Training in Media and Arts Technology EP/L01632X/1

Arts & Humanities Research Council
EPSRC Engineering and Physical Sciences Research Council
Queen Mary University of London