Functional Live Coding vs. DAWs and VSTs

Dr. Nick Rothwell
Ravensbourne University London
n.rothwell@rave.ac.uk

Creative Context: Sitar, Live Coding, VSTs

- Khyal Geometries: Shama Rahman (sitar), Nick Rothwell (software and processing)
- Non-linear environment, combining live coding with commercial VSTs and a matrix mixer
- VSTs address audio-processing tasks without re-inventing the wheel
- VST control via code seems like a fertile area of creative exploration

Tools: Max, VSTs, Node.js, ClojureScript

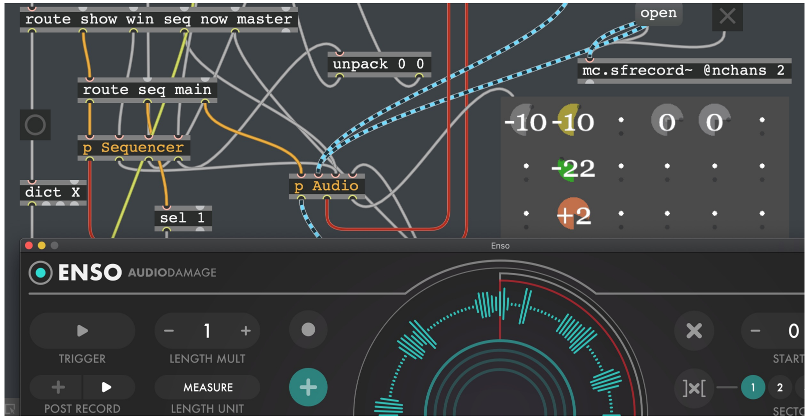


Figure 2

- Max as VST host and audio mixer
- Embedded Node.js acting as a parameter controller for VSTs and mixer
- ClojureScript live-coded via NREPL from Emacs

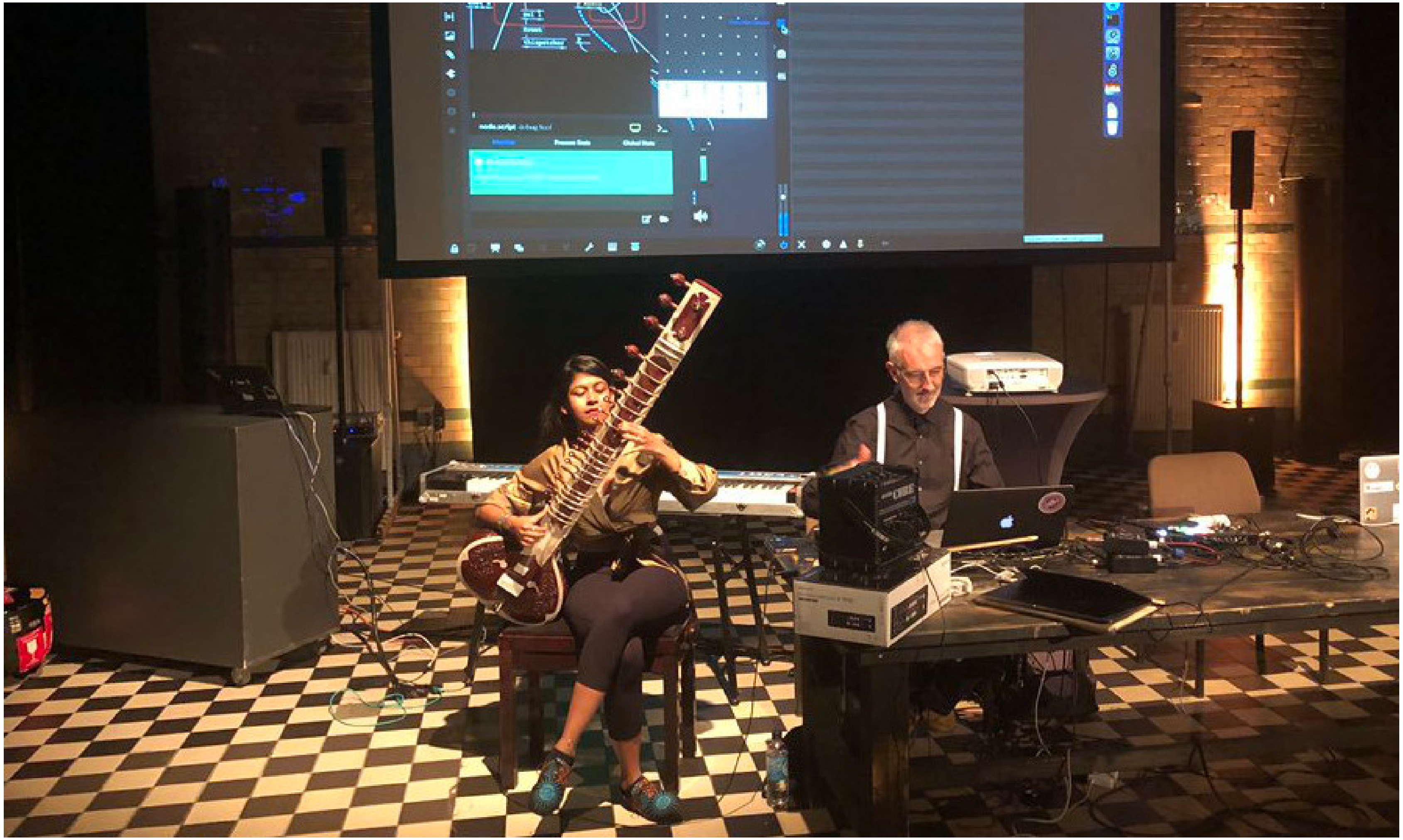
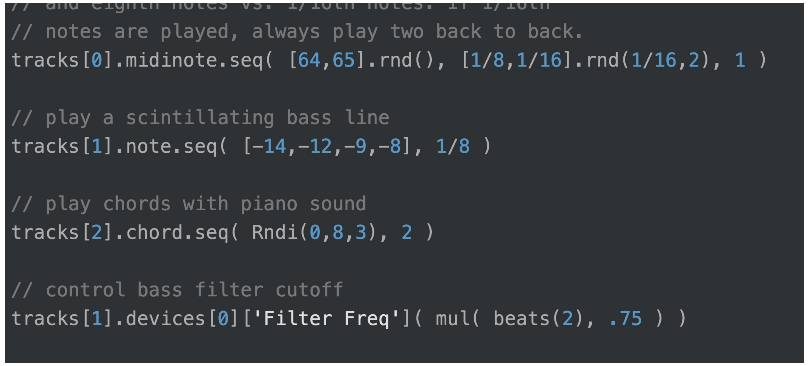


Figure 1

- No direct interaction with Max patchers or instruments–just on-screen for visual monitoring

Inspiration: Overtone, Gibberwocky



```
// and eighth notes vs. 1/16th notes. If 1/16th
// notes are played, always play two back to back.
tracks[0].midinote.seq( [64,65].rnd(), [1/8,1/16].rnd(1/16,2), 1 )

// play a scintillating bass line
tracks[1].note.seq( [-14,-12,-9,-8], 1/8 )

// play chords with piano sound
tracks[2].chord.seq( Rndi(0,8,3), 2 )

// control bass filter cutoff
tracks[1].devices[0]['Filter Freq']( mul( beats(2), .75 ) )
```

Figure 3

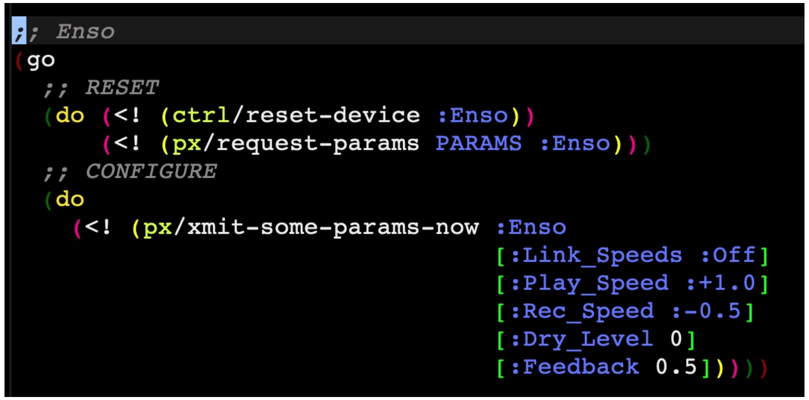
- Step sequencing from generative pattern expressions

- Table/map lookup for device parameters

- Gibberwocky's visual feedback (spark-lines etc.) replaced by Max's graphical interface

- Choice of Clojure for immutable data structures: ability to reverse changes or jump between cues, useful for improvisation and rehearsal

Issues: Resetting and Recalling Devices



```
;; Enso
(go
  ;; RESET
  (do (<! (ctrl/reset-device :Enso))
      (<! (px/request-params PARAMS :Enso)))
  ;; CONFIGURE
  (do
    (<! (px/xmit-some-params-now :Enso
                                 [:Link_Speeds :Off]
                                 [:Play_Speed :+1.0]
                                 [:Rec_Speed :-0.5]
                                 [:Dry_Level 0]
                                 [:Feedback 0.5]))))
```

Figure 4

- Conceptual clash between immutable data of Clojure and editable state of virtual instruments and effects

- Edits can be tracked, but are not always reversible–cannot emulate immutability

- Complete resets to known states (initial, preset)–discontinuous (sudden audio changes), so need to track levels of signal paths (when it is safe to recall a state?)

- Future directions: more sophisticated state tracking needed

References

Grosse, Darwin (2019). Node For Max Intro – Let's Get Started! January 2019. Available at `https://cycling74.com/articles/node-for-max-intro-%E2%80%93-let%E2%80%99s-get-started` (accessed [2019-09-22 Sun]).

FARM (2019). ACM SIGPLAN International Workshop on Functional Art, Music, Modelling and Design, Berlin, 18-23 August. Available at https://icfp19.sigplan.org/home/farm-2019 (accessed [2019-09-22 Sun]).

Roberts and Wakefield (2016). Live Coding the Digital Audio Workstation. International Conference on Live Coding, Hamilton, Canada.

McGranaghan (2012) Mark McGranaghan. ClojureScript: Functional Programming for JavaScript Platforms. 2012 IEEE Internet Computing 15(6):97 - 102.

Hauman (2015) Developing ClojureScript with Figwheel. Clojure/West, Portland, Oregon, April 20-22 2015.

Brannigan, Erin (2007) Music Makes the Moves. RealTime issue 76, Dec 2006-Jan 2007. Available at http://www.realtimearts.net/article/76/8266 (accessed [2019-09-22 Sun]).

Hickey, Rich (2016). Clojure core.async. Strange Loop, November 2013.

Hoare, C.A.R. (1978). Communicating Sequential Processes. Communications of the ACM 21/8, August 1978.

Redux: A predictable state container for JavaScript apps. Available at https://redux.js.org/ (accessed [2019-09-22 Sun]).

Transient Data Structures. Available at https://clojure.org/reference/transients (accessed [2019-09-22 Sun]).