# Intelligent Beheamoths: live coding with Ltt

Konstantinos Vasilakos

2023-10-19

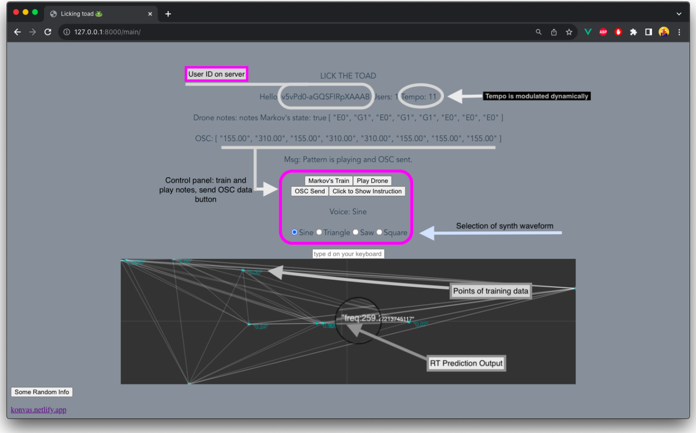
# Outline

1. The Idea
2. Interaction
3. Network Interaction
4. Sound Making
5. Live Coding
6. Control from SC
7. Demo
8. Reflections
9. Development

DESCRIPTION: Live coding with networked music system using Neural Network's regression algorithm trained by multiple users and sonified using SuperCollider's custom made sound synthesis modules.

# The Interface

The system is a client/server application and communicates with OSC with other sound design environment:



LICK THE TOAD

Hello v5vPd0-aGQSFIRpXAAAB Users: 1 Tempo: 11

Drone notes: notes Markov's state: true [ "E0", "G1", "E0", "G1", "G1", "E0", "E0", "E0" ]

OSC: [ "155.00", "310.00", "155.00", "310.00", "310.00", "155.00", "155.00", "155.00" ]

Msg: Pattern is playing and OSC sent.

Voice: Sine

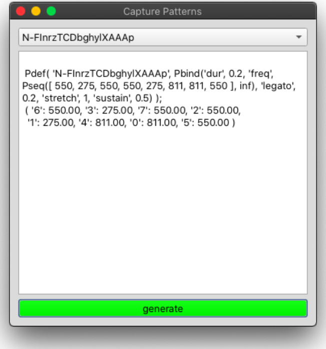
● Sine ○ Triangle ○ Saw ○ Square

"freq:259.2213745117"

Figure: Ltt interface and control GUI.

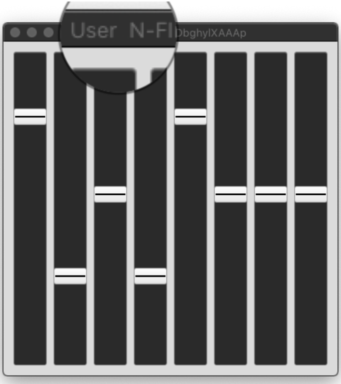
# Selection of Data on the Fly

SuperCollider's part of the system supports selection of each event through the user's ID in a GUI.



N-FlnrzTCDbghylXAAAp

```
 Pdef( 'N-FlnrzTCDbghylXAAAp', Pbind('dur', 0.2, 'freq',
Pseq([ 550, 275, 550, 550, 275, 811, 811, 550 ], inf), 'legato',
0.2, 'stretch', 1, 'sustain', 0.5) );
 ( '6': 550.00, '3': 275.00, '7': 550.00, '2': 550.00,
  '1': 275.00, '4': 811.00, '0': 811.00, '5': 550.00 )
```



# Interoperabillity

[ "A0", "C4", "C4", "A0", "G2", "G2", "A0", "G2" ] OSC:[ "214.00", "601.00", "601.00", "214.00", "428.00", "428.00", "214.00", "428.00" ]
The performance is shaped by the data of each user which is interpreted live while the performance unfolds. This way the sounds both from the decentralized domain of the user's together with the live electronics are creating an unpredictable sonic environment which enables the communication between audience and performer.

# Performance method

- The live coding performance is focussing on the real time sonification of the incoming data.
- Each user is sending their data in an array of notes.
- The coder is mapping this values in running processes with the help of streams and patterns of SuperCollider.

# Events Patterns and Streams

Performance elaborates using Patterns as random or sequential streams, and mapped to any sound parameter: Examples of patterns include: *Prand*, *Pseq*, *Pser*, *Pxrand* and more...

```
Pdef(\user_ID,
Pbind(\dur, 0.1,
\freq, Pseq([121.0, 220.0, 340.0, 280], inf)
));
```

# OSC Message Trigs

A small command line is also available from SC to set all users’ new states of the Markov chain and trigger back new data.

```
n=NetAddr("localhost", 57122);
n.sendMsg(’/kv’, "markov_train");
n.sendMsg(’/kv’, "osc_trigger");
```

# Live Coding.

Live coding in SC -> play some now. . .

# Points to consider

The project aims to serve as an interconnector between coder and audience:

- Incoming data is the stimuli for real time arbitrary sonifications forming a dynamic soundscape.
- An unpredictable sound result generated by the algorithm that is trying to connect the gaps by sending proximal values between X and Y coordinates from the user's device.
- The input serves as the seeding input for a larger chain of processing and interconnection of modules, such as the Markovian chains and the machine learning modules bound together to offer a great paradox of a calculated surprise of unpredictable sonifications.

# Technical Details

- Project is a client/server system encapsulating training/predictions.
- Communication client/server is bound through WebSockets, and Express server.
- Data from clients is using OSCJS library.
- It is hosted locally with NodeJS and online -> `https://lick-the-toad.netlify.app/`
    - Connection with users and OSC is supported only locally now.
- Sound synthesis of clients is using Tone.js and Markov chains extension for JS.