Code Concert: Writing live music with TidalCycles

Jason Medcoff

September 22 2018

Introduction

A little bit about me.
I'm a programmer, a musician, and a mathematics enthusiast.
I'm very excited today to give a talk involving all three.
If this excites you too, you're in the right place.

Introduction

What is this talk about?

- TidalCycles, a pattern based language
- Haskell, the programming language it's built on
- How to put them together to have a lot of fun

Functional programming is really awesome, therefore Haskell is awesome, though neither are the main focus here. We'll talk about it enough to understand what's going on.

Lists and You: the joy of FP

Lists are the quintessential data type. You can't do much without them.

- plain old arrays
- linked lists
- `java.util.ArrayList`
- `'()` or `nil`
- etc.

Lists and You

FP is the best way to work with lists, because no matter the language, abstract lists are recursively defined.

```
<List> = <[]> | <Object><List>
```

By the way, this is basically all of lisp.

```
(function first-arg second-arg etc...)
```

Lists and You

We can build lists by hand.

- `1 : []` gives `[1]`
- `1 : [2, 3]` gives `[1, 2, 3]`
- `[1, 2] : [[3]]` gives `[[1, 2], [3]]`
- `'h' : 'e' : 'l' : 'l' : 'o' : []` gives `"hello"`

Lists and You

Some takeaways:

- Haskell is strongly typed, and thus wants homogeneous lists
- Haskell treats strings as lists of characters

Enter TidalCycles: the Pattern

Let's start with a notion of tempo.
Tidal measures cycles per second (cps), but we can use some *DIMENSIONAL ANALYSIS*^{TM} to put this in more familiar terms.

$$\texttt{cps} = (120/60/4)$$

where 120 is beats per minute (bpm), 60 is seconds per minute, and 4 is beats per cycle. We are not restricted to this division.

The Pattern

What is a "list" in music?

- a lick
- a riff
- a melody
- a beat
- a bassline
- etc.

An example: the L I C C

An example: the L I C C

```
d1 $ slow 4 $ up
"[2 4 5 7 4 ~ 0 2] ~" # s "arpy"
```

- sample references
- rests
- sequences
- functions, of course

Extreme hand waving

To summarize:

```
<Sample-ref> = <Sample-name>":"<Sample-index>
        <Note> = <~> | <Sample-ref>
       <Notes> = < > | <Note><Notes>
<Pattern> = <Notes> | <Notes><Pattern><Notes>
```

The TidalCycles Architecture

Audio applications that aren't workstations tend to have a simple and familiar paradigm.

- A client, interfaced by an editor
- A server, listening for messages

The TidalCycles Architecture

Here's how Tidal does it.
The client is a Haskell interactive session running TidalCycles.
The server is a sampler running on SuperCollider.

Baby Steps

Let's make some noise.

```
d1 (pattern-effect (...
(sound "some-pattern"
# synth-effects)))
```

Make it concise:

```
d1 $ pattern-effect $
sound "some-pattern" # synth-effects
```

Examples

Examples

What can programs do that human performers or sequencing software can't easily do?

```
d1 $ s "rad1*8" # n $ run 8
```

Pay close attention to the `n` function. Then try `irand`.

Samples

Where do the samples come from?
I like to prepare my own, and combine them in interesting ways.

Where to go from here

Interested? Check this out.

- tidalcycles.org
- TOPLAP (toplap.org)
- bit.ly/codeconcert