TidalCycles workshop

[2 hours duration]

What is TidalCycles?

TidalCycles is a language (DSL[1]) to create musical and sonic patterns.

TidalCycles cooperates with other software in order to work.

[1] Domain-Specific Language - a programming language which is created based on a more a general purpose programming language. For this matter **Haskell**.

Our Goals

→ Make it sound

→ Create musical patterns

→ Transform patterns programmatically

→ Add expressiveness with effects

"The Trio"

→ Tidalcycles

→ Supercollider

→ Superdirt

and a text editor (emacs, vim, atom or sublimetext).

Supercollider

Supercollider is a language, sclang, to

→ create audio synthesis

→ audio effects

→ manipulate audio samples

→ midi/osc communications

→ has its own editor

SuperDirt

An engine/synthesizer that bridges Supercollider and Tidalcyles, with custom synths, effects and audio sample libraries.

TydalCycles does not make any sound by itself.

Install & Config

Follow the steps to Install & Config

A video for I&C in the Mac

Start SuperDirt

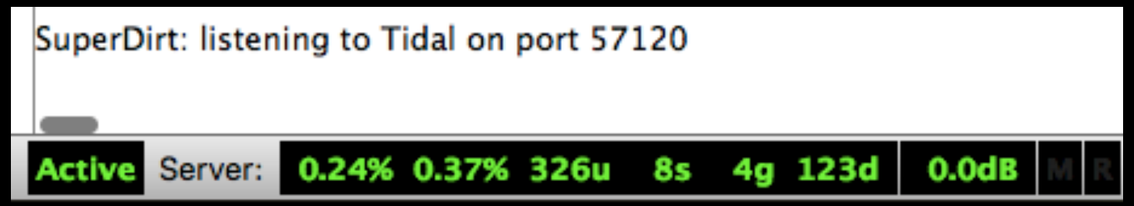
Now open Supercollider and in the left pane type:

```
SuperDirt.start;
```

Start SuperDirt

Keep text cursor in-line and press **[Shift] + [Enter]**

Then if SuperDirt is paired with Tidal, SuperDirt is on. In the right pane you'll have this,



Musical and Sonic Patterns

TidalCycles bases your musical ideas in **patterns**. Something you've heard before and, copy-pasted or with changes, will keep its character in future listenings.

How to create a pattern

Open the text editor you have chosen to work with Tidal and type the following:

```
d1 $ sound "bd"
```

How to create a pattern

```
d1 $ sound "bd"
```

d1* is a connection to Superdirt.

$ what is on its right side will be evaluated first

sound "bd" is a pattern of sound samples.

"bd" is a sample's or synth's name

* d1 to d9 connections available.

How to play a pattern

Keep text cursor in-line and press **[Shift] + [Enter]**.
This works like a ▶️ button.

How to play a pattern

Are you hearing something other than your breathing?

Yes?! You're On!

Play/Stop on the keyboard

Remember

1 line of code, place the cursor in-line and press **[Shift] + [Enter]**

1+ lines of code, place the cursor in-line anywhere and press **[CTRL] + [Enter]**

How to silence a pattern

Ok, having a blast and everything, but now it is time stop it.

To stop **d1** from playing

```
d1 silence
```

To stop all connections at once

```
hush
```

How to have silence among sounding samples

~ is a rest (silence). The absence of a sounding sample.

```
d1 $ sound "~ bd"
```

Identifying Samples

```
d1 $ sound "bd"
```

bd is the name of the folder where the sample is.

Identifying Samples

```
d1 $ sound "bd:1 bd:3 bd:4 hh"
```

Will play three different "bd" samples. The **:** followed by a number will id the sample and represents the file order within its folder. So,

bd:0 is the same as **bd** representing the first sample. **bd:4** representing the fifth sample, and so forth.

Identifying Samples

This samples naming logic applies to all samples you have access to.

'sound' pattern duration

You noticed that as soon as you played the **sound** pattern it started looping. TidayCycles predefines **1 cycle per second** or **120 beats per minute**.

```
cps(1) -- cycles per second

bps(120/60) -- 2  beats per second

d1 $ sound "bd:1" -- Will play one "bd:1" per cycle.

bps(90/60) -- 90 bps

d1 $ sound "lighter"
```

'sound' pattern duration

```
d1 $ sound "bd bd bd bd"
```

Will play one **bd** per 1/4 of a cycle, and four **bd** samples per cycle.

Patterns of patterns

Groups of patterns

[pattern1 [pattern2] ... pattern n]

```
d1 $ sound "bd [bd bd] bd bd" -- [1/4 [1/8 1/8] 1/4 1/4]
-- the same as
d1 $ sound "bd . bd bd . bd bd"

d1 $ sound "808bd:1 [bd:5 bd:3] bd [bd bd:2]"

d1 $ sound "bd:3 [jvbass jvbass] bd:4"
```

Patterns and polyrhythms

Polyrhythms

[pattern 1, pattern 2, ..., pattern n]

```
d1 $ sound "[bd, sn:3, numbers, hh:1]" -- play all at once

d1 $ sound "[bd bd bd, flick:6 flick:4 flick:6 flick:4]" -- polyrhythmic

d1 $ sound "[bd, [sn:3 sn:3], numbers, hh:1]" -- play all at once with grouping
```

Patterns varying per cycle

Varying per cycle

[<pattern 1 pattern 2 ... pattern n>]

```
d1 $ sound "<bd sn:3>" -- play each once per cycle

d1 $ sound "<bd sn:3 numbers hh:1>"
```

* to repeat more often a sample or pattern

```
d1 $ sound "808bd:1 808bd:1 808bd:1 808bd:1"
-- the same as
d1 $ sound "808bd:1*4"

d1 $ sound "bd [sn sn]"
-- the same as
d1 $ sound "bd sn*2"
```

/ to repeat less often a sample or pattern

```
d1 $ sound "house:5/2" -- once per two cycles

d1 $ sound "house house:5/2"
```

Mix and match [ ] < > * /

Be more creative with your patterns.

```
d1 $ sound "[bd ~, sn:1*2, ~ <co co*2>, hh:1/4]"
```

```
d1 $ sound "[bd ~, ~ sn:1, <jvbass jvbass:1 bass3:2>*4]" -- wicked!
```

Knowing this, we are set to create great grooves!

Programmatically selecting samples

Saving time while typing samples id's

```
-- way 1. Longer typing and error prone
d1 $ sound "arpy arpy:4 arpy:7 arpy:10" -- Major 7 chord arpeggio
-- way 2
d1 $ s "arpy*4" # n "0 4 7 10"
-- way 3
d1 $ n "0 4 7 10" # s "arpy"

-- sequencing one octave up and down
d1 $ n "0..7 6..1" # sound "supercomparator"
```

Transforming a pattern with coding

To transform a pattern programmaticaly we use **functions**.

```
d1 $ sound "[bd*2, bass3:2*4]"

-- now twice as fast. 'fast' is the function and '2' is the amount of speed
d1 $ fast 2 $ sound "[bd*2, bass3:2*4]"
```

Transforming a pattern with coding

Functions

slow • fast • rev • every

```
d1 $ slow 2 $ sound "[bd*2, bass3:2*4]" -- plays 2 times slower

-- 2, 3, or 4 times faster, each value for each cycle
d1 $ fast "<2 4 6>" $ sound "[bd*2, bass3:2*4]"

-- for every 4 cycles, twice as fast
d1 $ every 4 (fast 2) $ sound "[bd*2, bass3:2*4]"

-- one cycle forward, one cycle backwards
d1 $ every 2 (rev) $ sound "bd jvbass bd:1 jvbass:3"
```

Using Effects

To add effects we use **functions** and **#** operator.

```
d1 $ sound "arpy*16" # gain "0.8"
```

Using Effects

Functions

gain • pan • vowel • speed

```
-- default and maximum volume is 1, 0.5 to set to middle
d1 $ sound "arpy*16" # gain "0.5"

-- stereo panning from left to right
d1 $ sound "arpy*8" # pan "[0.0 0.5 1.0]/2"

-- vowels timbers of pattern sounds
d1 $ sound "[bd jvbass]*2" # vowel "[a o]"

-- speeds up, down and reverse sample playing
d1 $ sound "numbers:1" # speed "<4 1 0.125 -1.3>"
```

Using Effects

Functions

cutoff • coarse • room • size

```
--cuts off above 1000 Hz
d1 $ n "[0..7][6..1]" # sound "supercomparator" # cutoff 1000

-- downsizes audio resolution
d1 $ n "[0..7]" # sound "supercomparator" # coarse 20

-- chaining effects to add room and size to pattern's sound
d1 $ n "[7..0]" # sound "supercomparator" # room 2 # size 0.5
```

Some extra examples

```
-- 'sine1' is a sinusoidal curve sequence of decimal values from 0.0 to 1.0
-- 'scale' rescales values, here from 0.0 - 1.0 to 100 - 3000
d1 $ sound "[bd <jvbass:0*2 jvbass:1*2>]*2" # slow 1.75 (cutoff (scale 100 3000 sine1))


d1 $ n "[0..7]" # s "auto" # pan sine1


d1 $ n "[0..7]" # s "house" # slow 1.75 (cutoff (scale 140 3000 sine1))

-- 'tri1' is a triangle curve sequence of decimal values
d1 $ sound "[padlong, linnhats*3]"
# crush "<4 10>/4"
# slow 8 (bpf (scale 500 5000 tri1))
# slow 4 (bandq (scale 1 30 tri1))
```

Where to go for more

→ **TidalCycles** Website - tidalcycles.org **Manual**

→ **slack** Live coding team - slack.com **Community**

→ **LURK** - lurk.org **Community**

→ **toplap** - toplap.org **Live Coding Organization**

→ Youtube

→ Vimeo

Enjoy the awsomeness of making music while programming

by @pd3v_