Workshop 04:
TidalCycles Practical Tutorial 1
Rhythm Creation

Review: Sequence

Sequence

- Sequence: A repeating progression
- Repeating the same group of sounds over and over.
- In Strudel, sequences can be constructed in various ways.

Basic Sequence

```
-- TidalCycles
d1 $ sound "bd hc sd ho"
```

```
// Strudel
$: s("bd hh sd oh")
```

- The basics of a sequence.
- You can play multiple sounds within a sequence by separating them with spaces.

Basic Sequence

Main Drum Sounds

- bd: Bass drum “thud”
- sd: Snare drum “snap”
- hc (hh): Hi-hat “ch”
- ho (oh): Open hi-hat “chee”
- rim: Rimshot “clack”

- lt: Low tom “doh”
- mt: Mid tom “toh”
- ht: High tom “poh”
- cr: Crash cymbal “crash”
- cp: Clap “clap”

Playing Many Sounds

```
-- TidalCycles
d1 $ sound "bd bd hc bd sd bd hc bd"
```

```
// Strudel
$: s("bd bd hh bd sd bd hh bd")
```

- The longer the sequence (the more sounds it has), the faster it gets.
- This is because the contents of the sequence are crammed into one cycle.

Changing the Tempo

```
-- TidalCycles
d1 $ sound "bd bd hc bd sd bd hc bd" # cps(100/240)
```

```
// Strudel
$: s("bd bd hh bd rim bd hh bd").cpm(40)
```

- One way is to use cpm().
- cycles per minute
- The unit is how many cycles per minute.
- There are other methods, which will be described later.

Rests

```
-- TidalCycles
d1 $ sound "bd ~ hc ~ cp"
```

```
// Strudel
$: s("bd ~ hh ~ cp")
```

- ~ represents a rest (a part where no sound is produced).

Sub-sequence

```
-- TidalCycles
d1 $ sound "bd [hc hc] sd [hc ~ bd]"
```

```
// Strudel
$: s("bd [hh hh] sd [hh ~ bd]")
```

- Enclosing in [] creates a sub-sequence.
- It’s like further dividing an element within a sequence.

Multiplication - Speed Up

```
-- TidalCycles
d1 $ sound "bd hc*2 cp hc*3"
```

```
// Strudel
$: s("bd hh*2 cp hh*3")
```

- The “*” (multiplication) symbol divides and repeats within that sequence.
- It speeds up by the number of multiplications.

Speeding Up a Sequence

```
-- TidalCycles
d1 $ sound "bd [hc ~ cp]*2"
```

```
// Strudel
$: s("bd [hh ~ rim]*2")
```

- Multiplication of sub-sequences is also possible.
- It speeds up and repeats the part enclosed in [].

Super-Fast Multiplication

```
-- TidalCycles
d1 $ sound "bd ~ hc*16 ~ cp ~ hc*8"
```

```
// Strudel
$: s("bd ~ hh*16 ~ rim ~ hh*8")
```

- Increasing the number of multiplications makes the division finer.
- As you speed it up, it gradually starts to sound like a pitch.

Sub-sub-sequence

```
-- TidalCycles
d1 $ sound "bd [cp [hc hc] cp] hc"
```

```
// strudel
$: s("bd [ rim [hh hh] rim] hh")
```

- By nesting [], you can create a sub-sequence of a sub-sequence.

Today’s Topic: Let’s create some weird rhythms!

How to create weird rhythms

- Play in parallel
  - Polyrhythm
  - Polymeter
- Euclidean Rhythm

Playing in Parallel

```
-- TidalCycles
d1 $ sound "hc hc hc, cp cp"
```

```
// Strudel
$: s("hh hh hh, cp cp")
```

- Write two sequences separated by a comma (,).
- The two sequences are played in parallel.

Playing in Parallel 2

```
-- TidalCycles
d1 $ sound "hc ~ hc ~ hc, cp cp, ~ hc, bd"
```

```
// Strudel
$: s("hh ~ hh ~ hh, cp*5, ~ ht, bd")
```

- You can increase the number of sequences played in parallel as much as you want.
- Just keep separating them with commas (,).

Playing in Parallel 3

```
-- TidalCycles
d1 $ sound "hc ~ hc ~ hc, bd [cr, cp*4]"
```

```
// Strudel
$: s("hh ~ hh ~ hh, bd [cr, cp*4]")
```

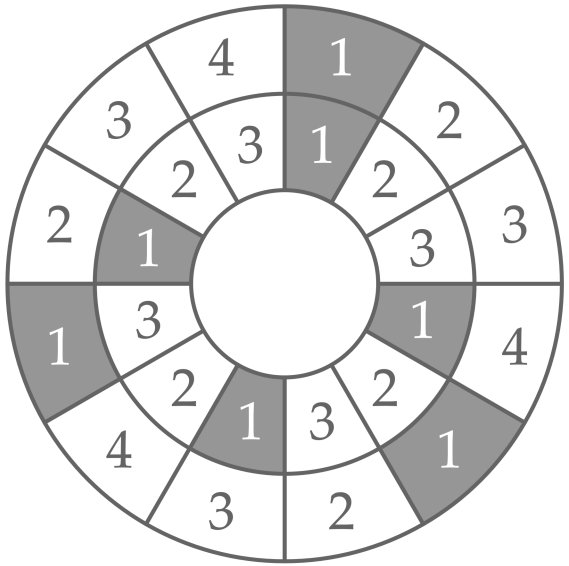
- It is also possible to play in parallel within a sub-sequence.

Polyrhythm and Polymeter

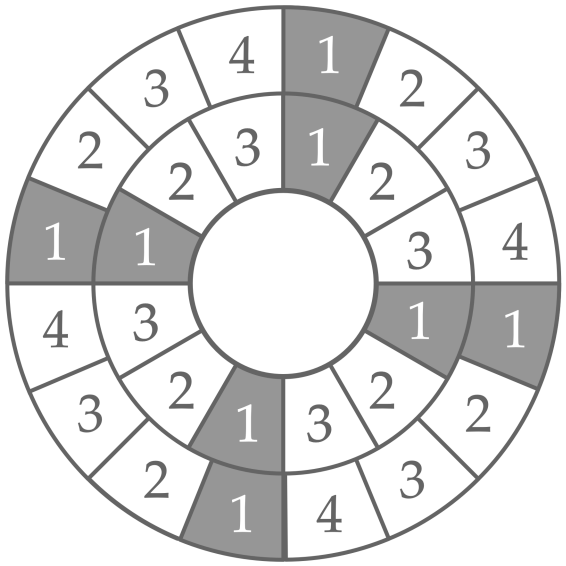
Polyrhythm and Polymeter

- The difference between Polyrhythm and Polymeter
  - Polyrhythm: The first beat is aligned, and the other beats shift.
  - Polymeter: All beats are aligned, and the first beat shifts.

3:4 polymeter



3:4 polyrhythm



Polyrhythm

```
-- TidalCycles
d1 $ sound "bd sd, hc hc hc"
```

```
// Strudel
$: s("bd sd, hh hh hh")
```

- Try playing rhythms with different time signatures in parallel.
- A unique twisted rhythm is generated → Polyrhythm
- Let’s try various combinations of time signatures!

```
-- TidalCycles
d1 $ sound "bd cr, hc*4, ~ cp cp ~ cp*2, ~ sd sd"
```

```
// Strudel
$: s("bd cr, hh*4, ~ cp cp ~ cp*2, ~ sd sd")
```

- Let’s try playing various polyrhythm patterns!

Polymeter

```
-- TidalCycles
d1 $ sound "{bd sd, cp hc ho}"
```

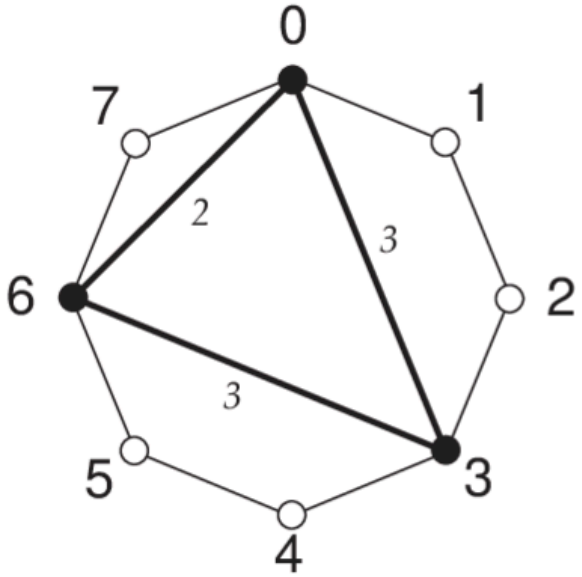
```
// Strudel
$: s("{bd sd, rim hh oh}")
```

- This time, try enclosing both ends of the pattern written in parallel with {}.
  - How does it change?
- The time signature is common, and the first beat shifts → Polymeter
- Let’s try various things with this too!

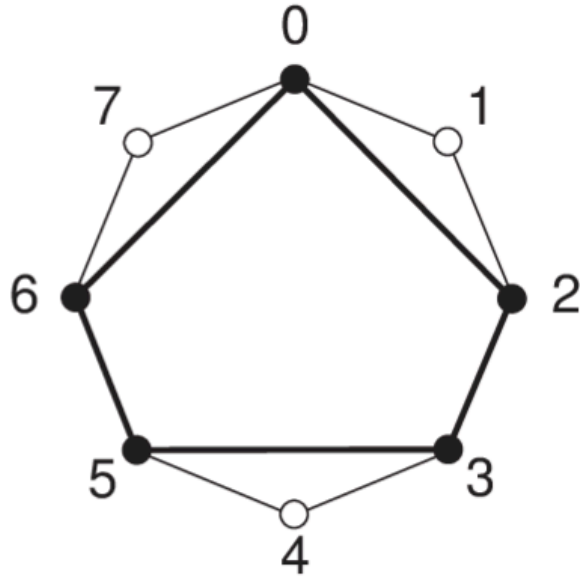
Euclidean Rhythm

What is Euclidean Rhythm?

- A rhythm pattern generated based on the Euclidean Algorithm.
- left: (3, 8), right: (5, 8)



(a)



(b)

- Found in various folk music rhythm patterns around the world.
- Examples: Cuban Tresillo, Brazilian Samba, Agadez in West Africa, etc.

How to make a Euclidean rhythm

- In the case of (3,8)
  - [11100000]
  - [10] [10] [10] [00]
  - [100][100][10]
  - [10010010]
- In the case of (5,8)
  - [11111000]
  - [10][10][10][11]
  - [101] [101] [10]
  - [10110110]

Euclidean Rhythm

```
-- TidalCycles
d1 $ sound "bd(3, 8), hc(9, 16)"
```

```
// Strudel
$: s("bd(3, 8), hh(9, 16)")
```

- An example of a Euclidean rhythm.
- Various complex rhythm patterns can be created.
- Let’s try it!

Euclidean Rhythm

```
-- TidalCycles
d1 $ sound "bd(3, 7), hc(9, 14, 5), cp(5, 7, 2)"
```

```
// Strudel
$: s("bd(3, 7), hh(9, 14, 5), cp(5, 7, 2)")
```

- If you make the denominator an odd number, you can generate a rhythm with an odd time signature!
- It is possible to imitate the rhythm structure of various folk music from around the world.

Effects

Effects

```
-- TidalCycles
d1
  $ sound "bd sd [~ bd] hc"
  # delay 0.8
  # delaytime (5/8)
  # lock 1
```

```
// Strudel
$: s("bd sd [~ bd] hh")
   .bank("tr909")
   .delay(0.8).delaytime(5/8)
```

- TidalCycles / Strudel has various effects available.
- Delay, reverb, flanger, etc.
- Try applying a delay to the rhythm.
- By devising the delay time, a groove is born!

Effects

```
-- TidalCycles
d1
  $ sound "bd cp [~ bd] hc*2, bd:2(3,8), hc*8, cp(7,8)"
  # delay 0.8 # delaytime (9/8) # delayfeedback 0.5
  # lock 1
  # lpf (range 400 8000 $ slow 16 $ sine)
  # resonance 0.2
  # room 0.5 # size 0.5
```

```
// Strudel
$: s("bd cp [~ bd] hh*2, bd:2(3,8), hh*8, cp(7,8)")
   .bank("tr909")
   .delay(0.8).delaytime(9/8).delayfeedback(0.5)
   .lpf(sine.segment(128).range(400,8000).slow(16)).resonance(10)
   .room(0.5).size(1.5)
   .punchcard().color("pink")
```

- Let’s add more various effects!
- Delay + Reverb + Low-pass filter

Exercise: Let’s make a weird rhythm!

Exercise: Let’s make a weird rhythm!

Today’s exercise!

- Let’s make a “weird” rhythm using Strudel!
- Odd time signatures, polyrhythms, polymeters, etc.
- Let’s try to devise a subtle adjustment that is not normal but also not too messy!
- Reference
- I’ve put together a playlist of songs with weird rhythms on Spotify!
- https://open.spotify.com/playlist/6J9H9ErKXep2xjPqvi09OB