

# TIDAL LOOPER

An introduction into live sampling with



tidalcycles

TidalCycles meetup #2, 2021-05-08

**MRREASON**  
AKA  
**THOMAS GRUND**

Tidalist, electric guitarist and  
software engineer



# INTRODUCTION

# MOTIVATION AND INSPIRATION



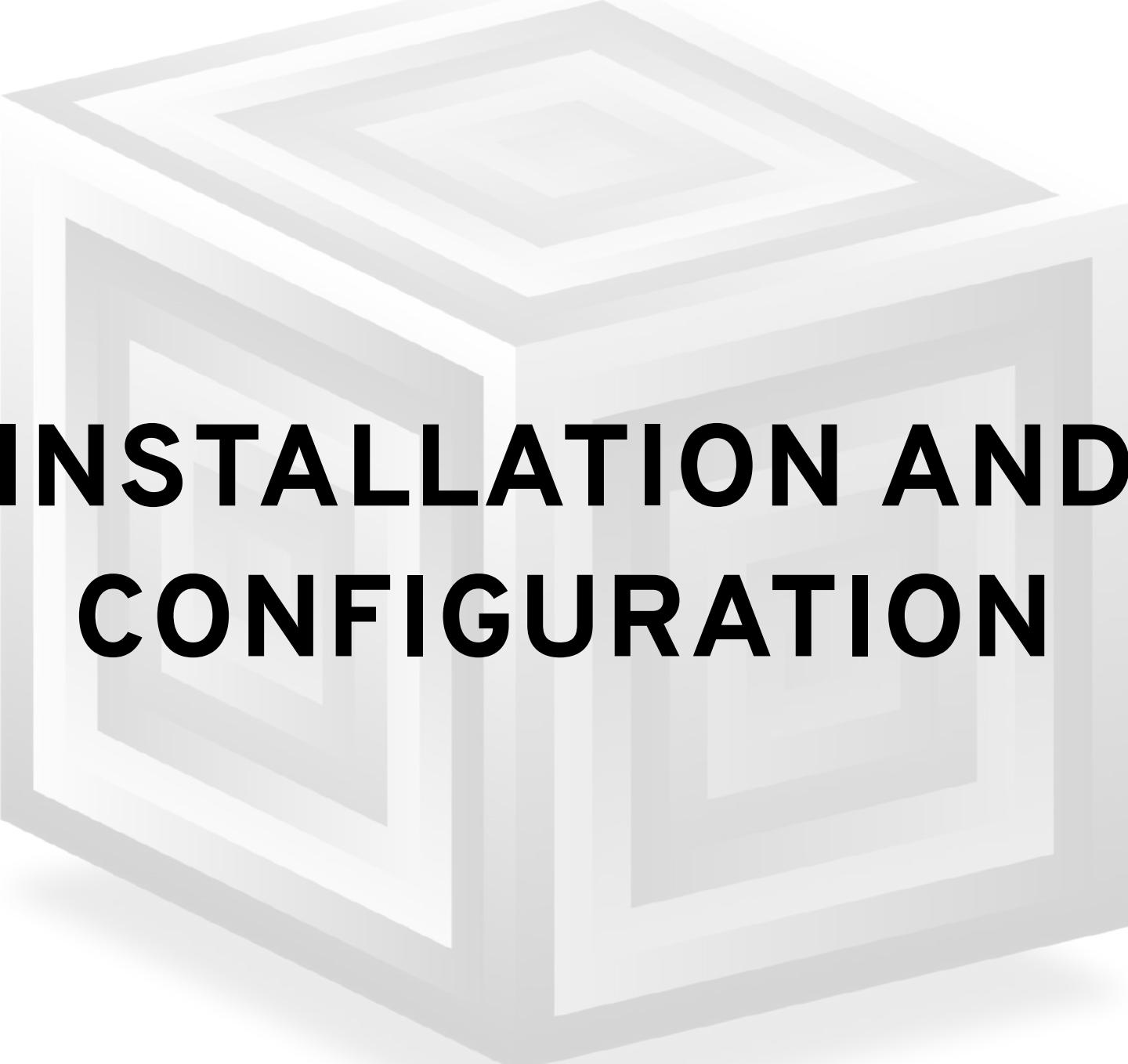
```
1 d1 $ s "looper" # n "<0 1 2 3>"  
2  
3 d2 $ s "loop" # n "[0,1,2,3]"
```

# BASIC REPLACE MODE EXAMPLE

```
1 d1 $ s "rlooper"
2     # n "< 0 1 2 3 >"
3     # linput 14
4
5 d2 $ s "loop"
6     # n "[0, 1, 2, 3]"
```

# BASIC OVERDUB MODE EXAMPLE

```
1 d1 $ s "olooper"
2      # linput 14
3
4 d2 $ s "loop"
```



# **INSTALLATION AND CONFIGURATION**

# DOWNLOAD FROM GITHUB

The screenshot shows a GitHub repository page for 'thgrund / tidal-looper'. The 'Code' dropdown menu is open, and the 'Download ZIP' option is highlighted with a red box. The repository details include 2 branches and 0 tags. The 'About' section describes the project as providing different looper variants for SuperDirt in TidalCycles. Tags listed include osc, live-coding, supercollider, hydra, looper, obs-studio, and tidalcycles. The 'Readme' and 'GPL-3.0 License' files are also listed. The 'Languages' section shows SuperCollider at 100.0%.

thgrund / tidal-looper

Code Issues 5 Pull requests Actions Projects Wiki Security Insights Settings

master 2 branches 0 tags Go to file Add file Code

**Clone**  
HTTPS SSH GitHub CLI  
https://github.com/thgrund/tidal-looper Use Git or checkout with SVN using the web URL.

**Open with GitHub Desktop**

**Download ZIP**

**About**

Different looper variants for SuperDirt to provide live sampling in TidalCycles.

osc live-coding supercollider  
hydra looper obs-studio  
tidalcycles

Readme  
GPL-3.0 License

**Releases**

No releases published Create a new release

**Packages**

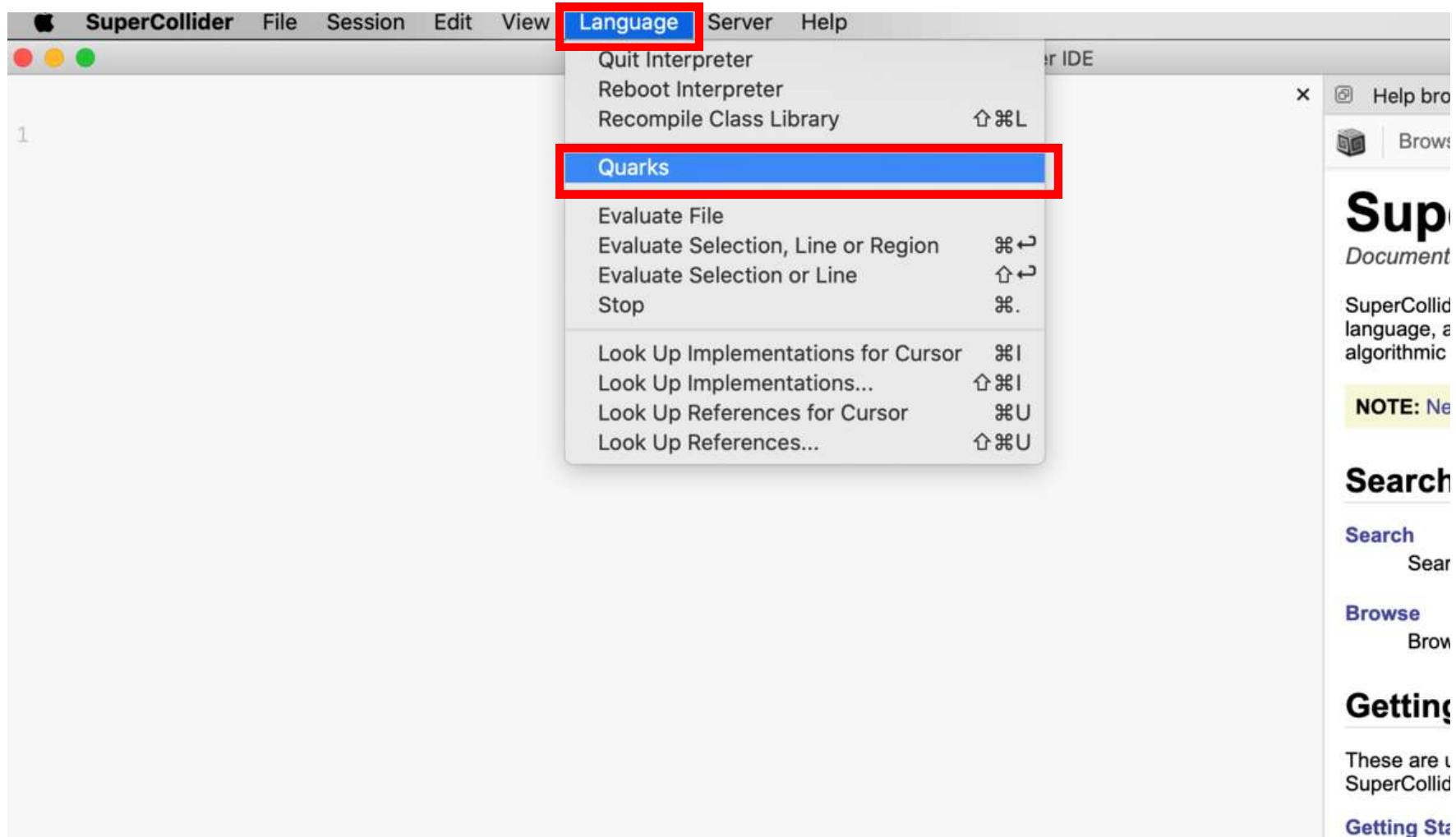
No packages published Publish your first package

**Languages**

SuperCollider 100.0%

URL: <https://github.com/thgrund/tidal-looper>

# INSTALL THE QUARK !



# INSTALL THE QUARK II

SuperCollider 3.11

Browse Search Indexes ▾

Quarks

Check for updates **Install a folder** Uninstall all Save Quarks set Load Quarks set Help Recompile class library

Name	Version	Summary
Dirt-Samples		Set of samples used in Dirt
KDTree		A kd-tree data structure, for efficiently querying large collections of points in...
PitchShiftPA	1.0.1	Phase-Aligned Pitch and Formant Shifter
SuperDirt		SuperCollider implementation of the Dirt sampler for the Tidal programming ...
Vowel		Convenience Class for Vowel Creation
XML	0.21	XML parsing and formatting according to the DOM specification.
wslib	0.31	various language additions, GUI classes and more Main features: SimpleMID...
SuperSampler	0.2.5	Polyphonic concatenative sampler synthesizer that combineing differnet sou...
+ 3Dj		
+ APCmini		
+ API		
+ AlGui		
+ AmbiEM		
+ ArdourOSC		
+ ArduinoQuaternion		
+ AudacityLabels		
+ AudioMulchClock		
+ AudioUnitBuilder		
+ AuditoryAugmentation		
+ Automation		
+ BBCut		
+ BandSplitter		
+ BatLib		
+ Bending		
+ Riorlund		

Documentation indexes

Documents Alphabetical index of all documents

# INSTALL THE QUARK III

Untitled - SuperCollider IDE

Untitled

x Help browser Home ← → ↻

Check for updates Install a folder Uninstall all Save Quarks set Load Quarks set Help Recompile class library

Quarks

Name	Version	Summary
✓ tidal-looper		Set of samples used in Dirt
✓ Dirt-Samples		A set of samples used in Dirt
✓ KDTree		A kd-tree data structure, for efficiently querying large collections of points in...
✓ PitchShiftPA	1.0.1	Phase-Aligned Pitch and Formant Shifter
✓ SuperDirt		SuperCollider implementation of the Dirt sampler for the Tidal programming ...
✓ Vowel		Convenience Class for Vowel Creation
✓ XML	0.21	XML parsing and formatting according to the DOM specification.
✓ wslib	0.31	various language additions, GUI classes and more Main features: SimpleMID...
+ SuperSampler	0.2.5	Polyphonic concatenative sampler synthesizer that combines different sou...
+ 3Dj		
+ APCmini		
+ API		
+ AllGUI		

**SuperCollider**  
SuperCollider is an audio server, programmatic audio engine, and IDE for sound synthesis and music composition.

**News in SuperCollider version**

**Search and browse**  
Search all documents and methods  
Browse all documents by category

**Getting started**  
Useful starting points for getting started with SuperCollider:

**Getting Started tutorial series**  
Get started with SuperCollider

**Glossary**  
Glossary

# SUPERCOLLIDER STARTUP SCRIPT

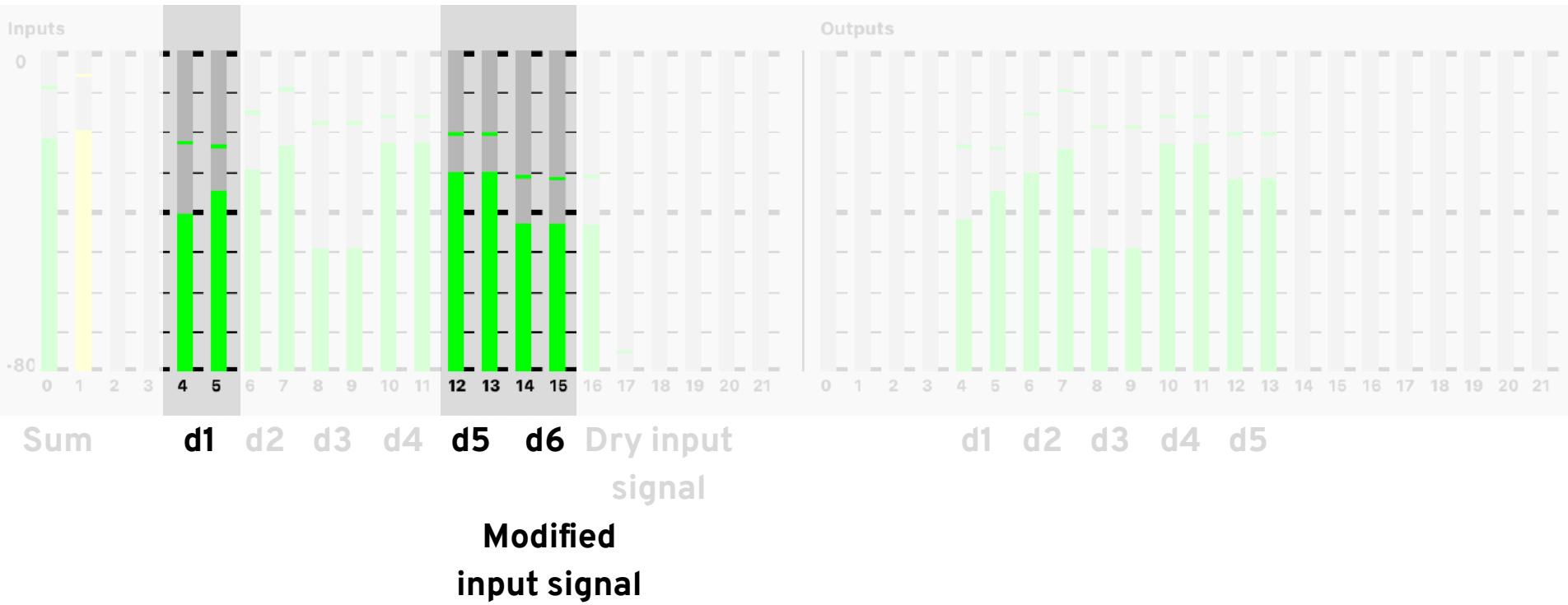
```
1  (
2      s.reboot {
3          s.options.numBuffers = 1024 * 256;
4          s.options.memSize = 8192 * 32;
5          s.options.numWireBufs = 512;
6          s.options.maxNodes = 1024 * 32;
7          s.options.numOutputBusChannels = 22;
8          s.options.numInputBusChannels = 22;
9
10     s.waitForBoot {
11         ~dirt = SuperDirt(2, s);
12         ~dirt.loadSoundFiles;
13
14         ~dirt.start(57120, [4, 6, 8, 10, 12, 14, 16, 18, 20]);
15
16         // optional, needed for convenient access from sclang:
17         (
18             ~d1 = ~dirt.orbits[0]; ~d2 = ~dirt.orbits[1]; ~d3 = ~dirt.orbits[2];
19             ~d4 = ~dirt.orbits[3]; ~d5 = ~dirt.orbits[4]; ~d6 = ~dirt.orbits[5];
20             ~d7 = ~dirt.orbits[6]; ~d8 = ~dirt.orbits[7]; ~d9 = ~dirt.orbits[8];
21             ~d10 = ~dirt.orbits[9]; ~d11 = ~dirt.orbits[10]; ~d12 = ~dirt.orbits[11];
22         );
23     };
24
25     s.latency = 0.2;
26
27     ~looper = TidalLooper(~dirt);
28 }
29 )
```

# SUPERCOLLIDER STARTUP SCRIPT

```
1  (
2      s.reboot {
3          s.options.numBuffers = 1024 * 256;
4          s.options.memSize = 8192 * 32;
5          s.options.numWireBufs = 512;
6          s.options.maxNodes = 1024 * 32;
7          s.options.numOutputBusChannels = 22;
8          s.options.numInputBusChannels = 22;
9
10     s.waitForBoot {
11         ~dirt = SuperDirt(2, s);
12         ~dirt.loadSoundFiles;
13
14         ~dirt.start(57120, [4, 6, 8, 10, 12, 14, 16, 18, 20]);
15
16         // optional, needed for convenient access from sclang:
17         (
18             ~d1 = ~dirt.orbits[0]; ~d2 = ~dirt.orbits[1]; ~d3 = ~dirt.orbits[2];
19             ~d4 = ~dirt.orbits[3]; ~d5 = ~dirt.orbits[4]; ~d6 = ~dirt.orbits[5];
20             ~d7 = ~dirt.orbits[6]; ~d8 = ~dirt.orbits[7]; ~d9 = ~dirt.orbits[8];
21             ~d10 = ~dirt.orbits[9]; ~d11 = ~dirt.orbits[10]; ~d12 = ~dirt.orbits[11];
22         );
23     };
24
25     s.latency = 0.2;
26
27     ~looper = TidalLooper(~dirt);
28 }
29 )
```

# AUDIO BUSES IN SUPERCOLLIDER

```
1 s.options.numOutputBusChannels = 22;  
2 s.options.numInputBusChannels = 22;  
3 ~dirt.start(57120, [4, 6, 8, 10, 12]);
```



# CONFIGURE THE DEFAULT VALUES

```
1 // You can adjust these parameter even in runtime
2 ~looper.rLevel = 1.5; // Default 1.0
3 ~looper.pLevel = 1.2; // Default 1.0
4 ~looper.linput = 15; // Default 0; Set this to your main input port.
5 ~looper.lname = "mybuffer"; // Default "loop"
6
7 // Recording synthdef with envelope
8 SynthDef(\buffRecordEnv, {|input = 0, pLevel = 0.0, rLevel = 1.0, buffer|
9     var in = SoundIn.ar(input);
10    var env = EnvGen.ar(Env.asr(0.01, 1, 1, 1), 1, doneAction:2);
11
12    RecordBuf.ar(in * env, buffer, recLevel: rLevel, preLevel: pLevel,
13    loop: 0, run: 1, doneAction: Done.freeSelf);
14
15 }).add;
16
17 ~looper.looperSynth = "buffRecordEnv";
```

# CONFIGURE THE DEFAULT VALUES

```
1 // You can adjust these parameter even in runtime
2 ~looper.rLevel = 1.5; // Default 1.0
3 ~looper.pLevel = 1.2; // Default 1.0
4 ~looper.linput = 15; // Default 0; Set this to your main input port.
5 ~looper.lname = "mybuffer"; // Default "loop"
6
7 // Recording synthdef with envelope
8 SynthDef(\buffRecordEnv, {|input = 0, pLevel = 0.0, rLevel = 1.0, buffer|
9     var in = SoundIn.ar(input);
10    var env = EnvGen.ar(Env.asr(0.01, 1, 1, 1), 1, doneAction:2);
11
12    RecordBuf.ar(in * env, buffer, recLevel: rLevel, preLevel: pLevel,
13    loop: 0, run: 1, doneAction: Done.freeSelf);
14
15 }).add;
16
17 ~looper.looperSynth = "buffRecordEnv";
```

# CONFIGURE THE DEFAULT VALUES

```
1 // You can adjust these parameter even in runtime
2 ~looper.rLevel = 1.5; // Default 1.0
3 ~looper.pLevel = 1.2; // Default 1.0
4 ~looper.linput = 15; // Default 0; Set this to your main input port.
5 ~looper.lname = "mybuffer"; // Default "loop"
6
7 // Recording synthdef with envelope
8 SynthDef(\buffRecordEnv, {|input = 0, pLevel = 0.0, rLevel = 1.0, buffer|
9     var in = SoundIn.ar(input);
10    var env = EnvGen.ar(Env.asr(0.01, 1, 1, 1), 1, doneAction:2);
11
12    RecordBuf.ar(in * env, buffer, recLevel: rLevel, preLevel: pLevel,
13    loop: 0, run: 1, doneAction: Done.freeSelf);
14 }
15 }).add;
16
17 ~looper.looperSynth = "buffRecordEnv";
```

# CONFIGURE THE DEFAULT VALUES

```
1 // You can adjust these parameter even in runtime
2 ~looper.rLevel = 1.5; // Default 1.0
3 ~looper.pLevel = 1.2; // Default 1.0
4 ~looper.linput = 15; // Default 0; Set this to your main input port.
5 ~looper.lname = "mybuffer"; // Default "loop"
6
7 // Recording synthdef with envelope
8 SynthDef(\buffRecordEnv, {|input = 0, pLevel = 0.0, rLevel = 1.0, buffer|
9     var in = SoundIn.ar(input);
10    var env = EnvGen.ar(Env.asr(0.01, 1, 1, 1), 1, doneAction:2);
11
12    RecordBuf.ar(in * env, buffer, recLevel: rLevel, preLevel: pLevel,
13    loop: 0, run: 1, doneAction: Done.freeSelf);
14
15 }).add;
16
17 ~looper.looperSynth = "buffRecordEnv";
```

# PERSIST AND RESET

```
1 // Resets buffers "loops"
2 ~looper.freeLoops()
3
4 // Persist buffers "loops"
5 ~looper.persistLoops()
6
7 // Configure persistance path
8 ~looper.persistPath = "/your/root/path/"
```

# PERSIST AND RESET

```
1 // Resets buffers "loops"
2 ~looper.freeLoops()
3
4 // Persist buffers "loops"
5 ~looper.persistLoops()
6
7 // Configure persistance path
8 ~looper.persistPath = "/your/root/path/"
```

# PERSIST AND RESET

```
1 // Resets buffers "loops"
2 ~looper.freeLoops()
3
4 // Persist buffers "loops"
5 ~looper.persistLoops()
6
7 // Configure persistance path
8 ~looper.persistPath = "/your/root/path/"
```

# CONFIGURATION IN TIDALCYCLES

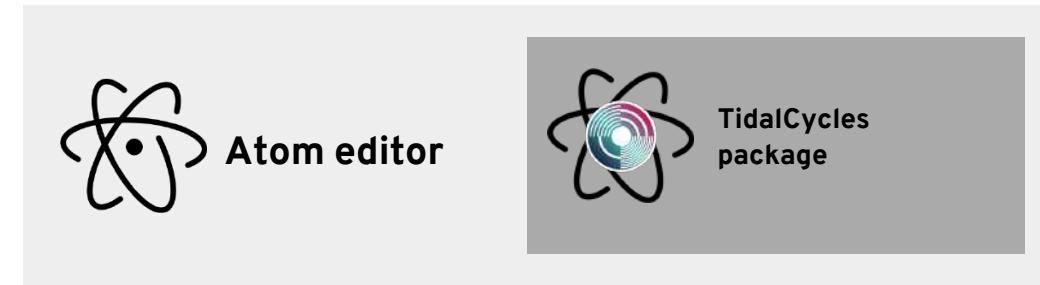
```
1 -- You could add this in BootTidal.hs
2
3 -- Change input bus
4 linput = pI "linput"
5
6 -- Change the buffer name
7 lname = pS "lname"
```

# CONFIGURATION IN TIDALCYCLES

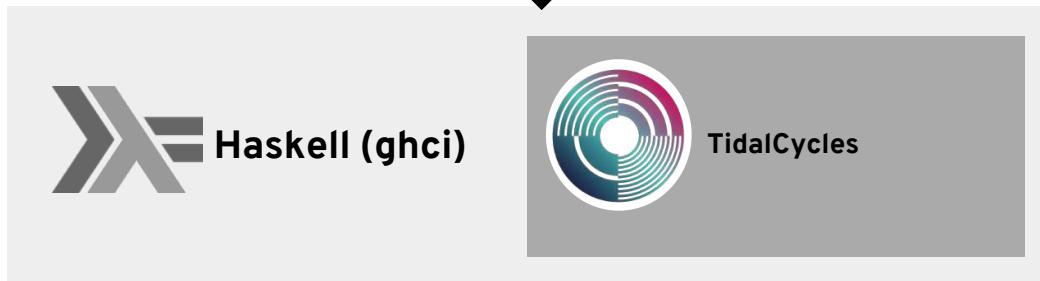
```
1 -- You could add this in BootTidal.hs
2
3 -- Change input bus
4 linput = pI "linput"
5
6 -- Change the buffer name
7 lname = pS "lname"
```

# TECHNICAL SETUP

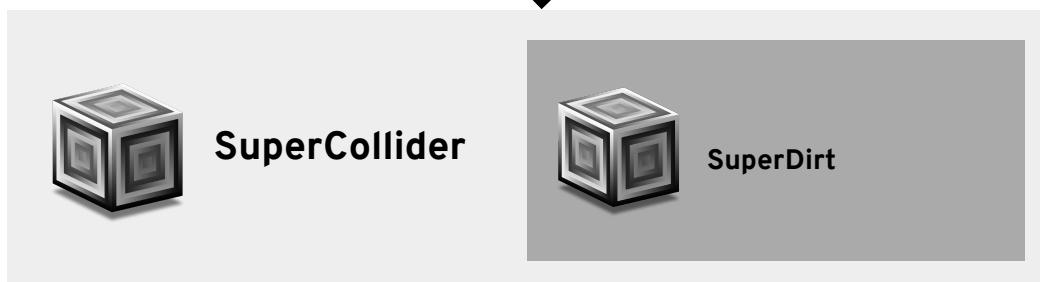
# CONTROL THE RECORDING



```
1 d1 $ s "looper"  
2 # n "<0 1 2 3>"
```

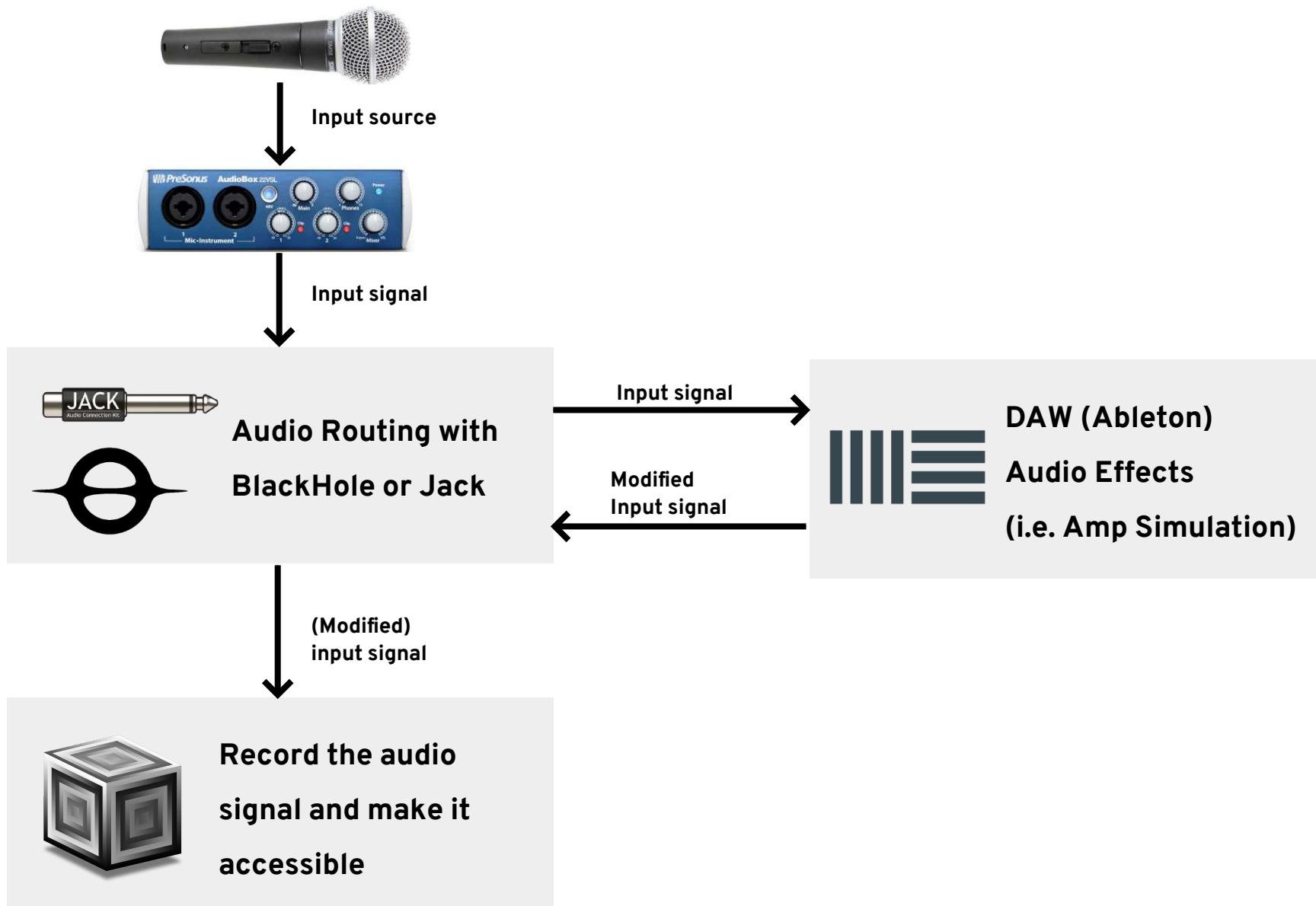


```
1 [ /play2, cps, 0.5625,  
2 cycle, 23.0, orbit, 0,  
3 delta, 0.59259271,  
4 n, 2.0, s, looper ]
```

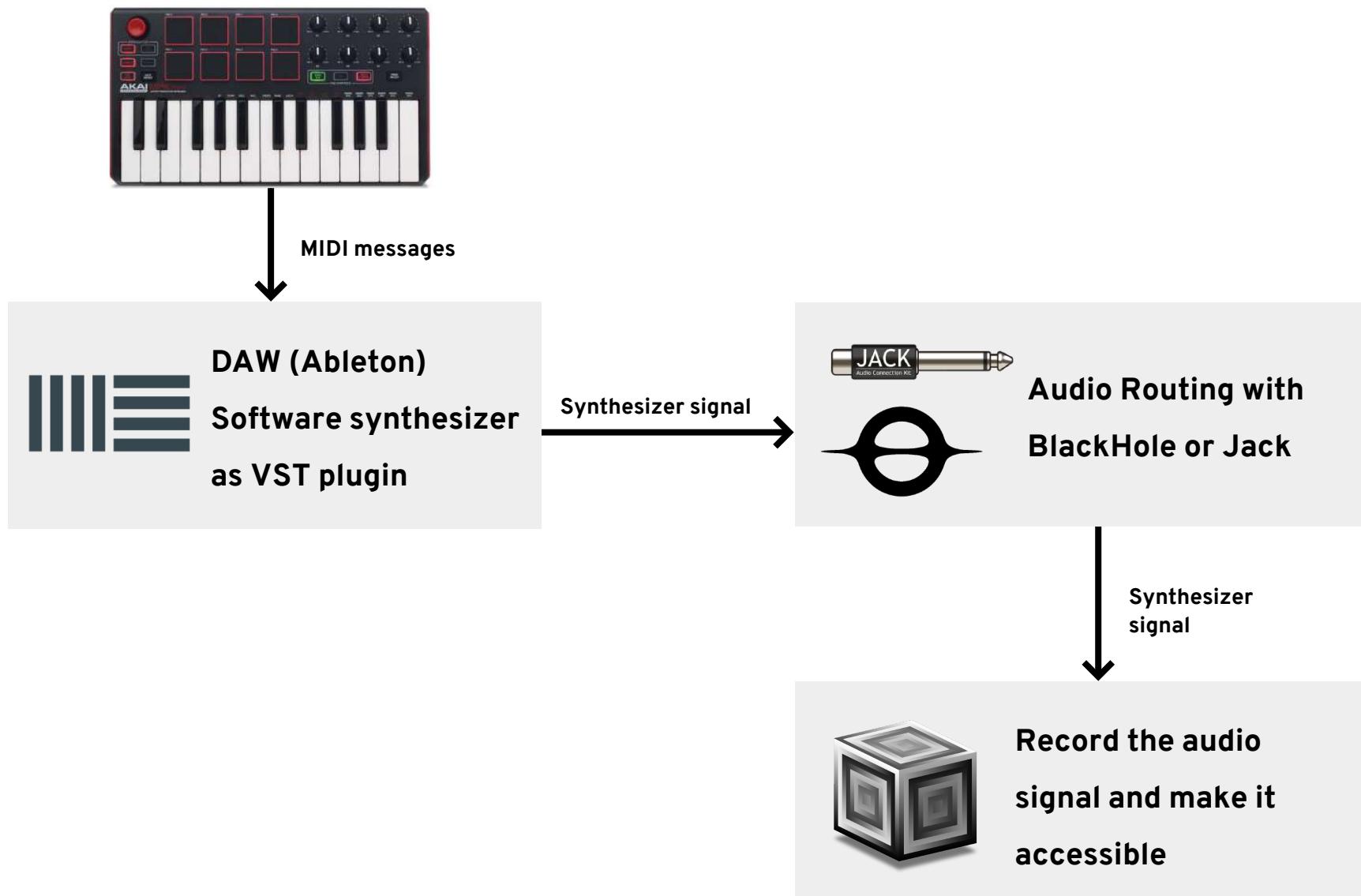


```
1 /*  
2 Includes the looper and  
3 buffer memory.  
4  
5 Processes the requests.  
6 */
```

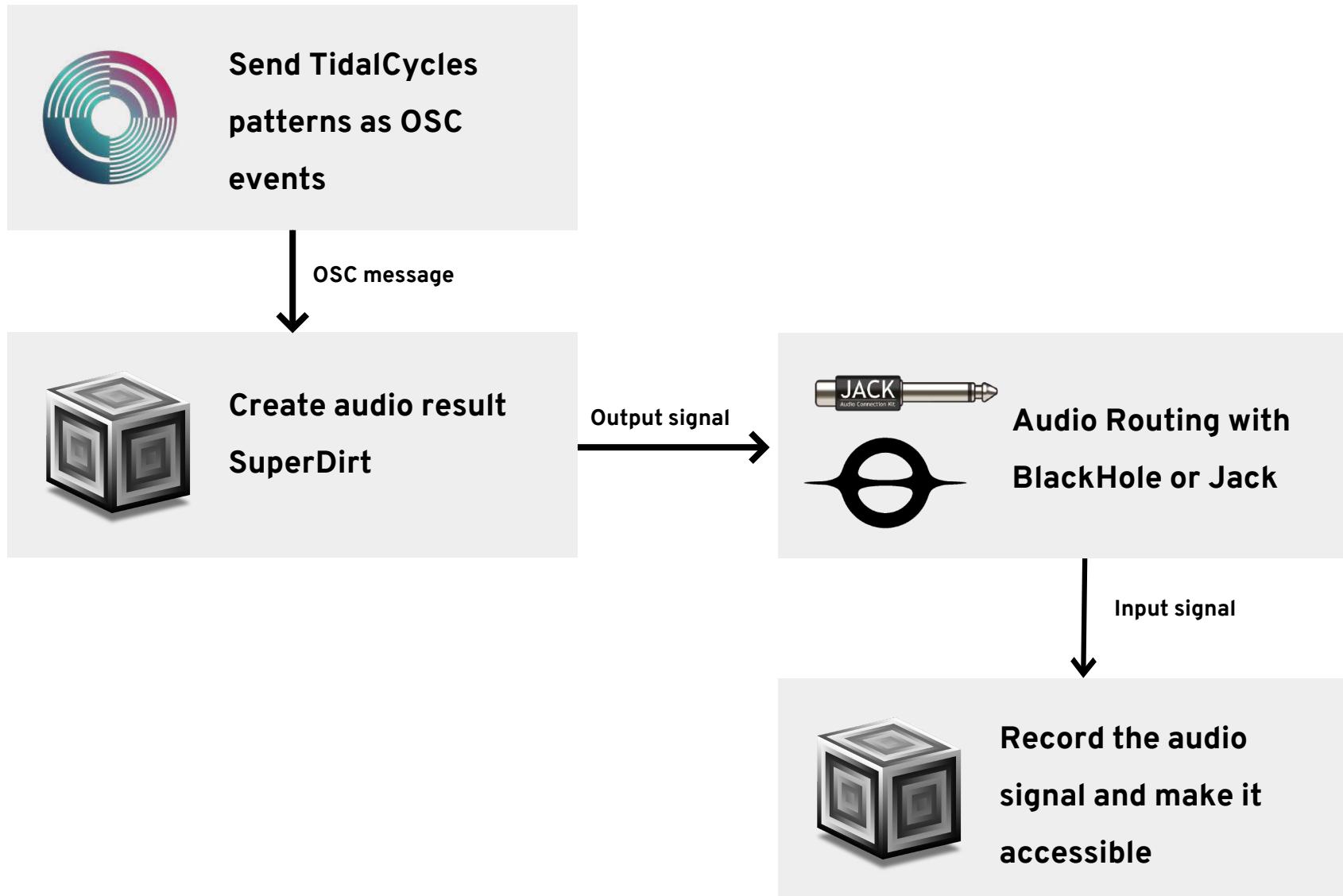
# RECORD AN INPUT SIGNAL



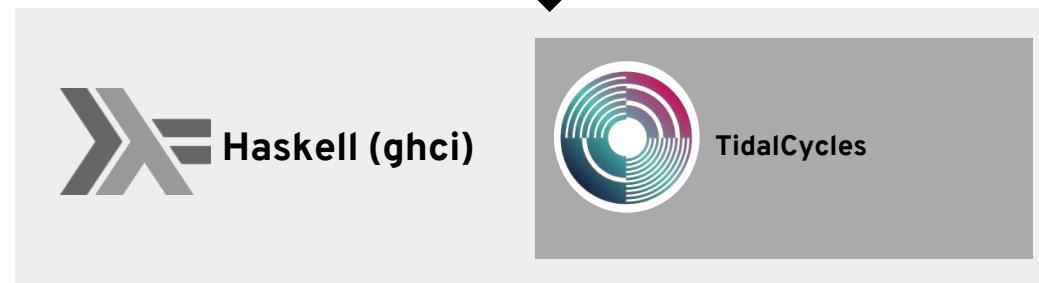
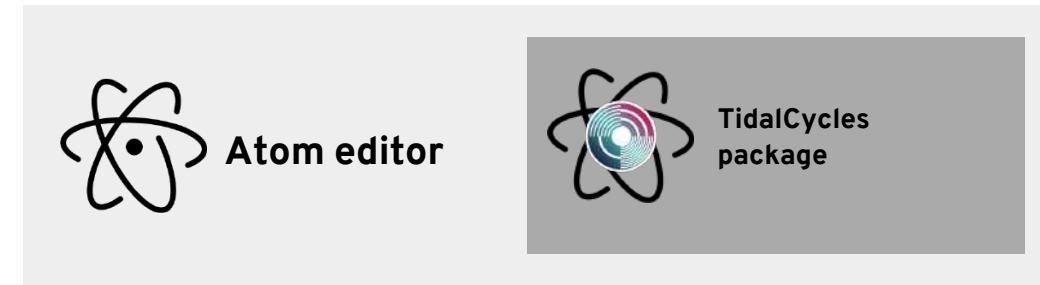
# RECORD A VST SYNTHESIZER



# RECORD A TIDAL PATTERN



# CONTROL THE PLAYBACK



Code

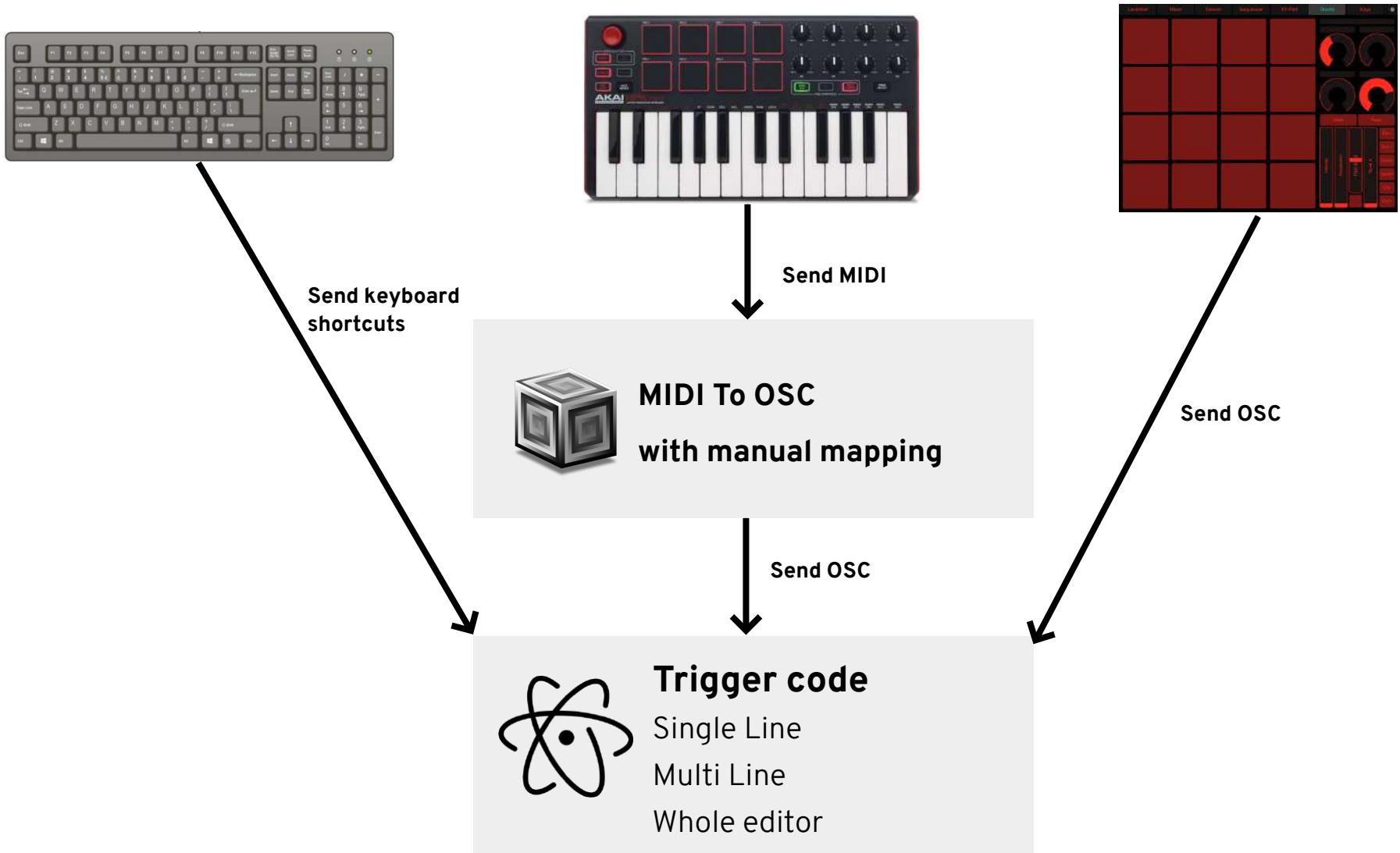
OSC messages  
(Pattern Events)

```
1 d1 $ s "loop"
2 # n "[0,1,2,3]"
```

```
1 [ /play2, cps, 0.5625,
2   cycle, 23.0, orbit, 1,
3   delta, 0.59259271,
4   n, 3.0, s, loop ]
```

```
1 /*
2 Includes the looper and
3 buffer memory.
4
5 Processes the requests.
6 */
```

# TRIGGER THE CODE



# TRIGGER THE CODE

```
1  (
2      s.waitForBoot {
3          ~dirt = SuperDirt(2, s);
4          // More SuperDirt ...
5
6          ~atomOSC = NetAddr("127.0.0.1", 3333);
7
8          MIDIFunc.cc({arg src, chan, num, val;
9              if (num == 60, {
10                  ~atomOSC.sendMsg("/atom/eval", \type, "line");
11              });
12
13              if (num == 61, {
14                  ~atomOSC.sendMsg("/atom/eval", \type, "multi_line");
15              });
16
17              if (num == 62, {
18                  ~atomOSC.sendMsg("/atom/eval", "whole_editor")
19              });
20          });
21      }
22  )
```

# TRIGGER THE CODE

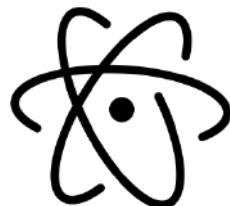
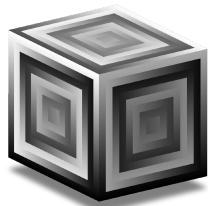
```
1  (
2      s.waitForBoot {
3          ~dirt = SuperDirt(2, s);
4          // More SuperDirt ...
5
6          ~atomOSC = NetAddr("127.0.0.1", 3333);
7
8          MIDIFunc.cc({arg src, chan, num, val;
9              if (num == 60, {
10                  ~atomOSC.sendMsg("/atom/eval", \type, "line");
11              });
12
13              if (num == 61, {
14                  ~atomOSC.sendMsg("/atom/eval", \type, "multi_line");
15              });
16
17              if (num == 62, {
18                  ~atomOSC.sendMsg("/atom/eval", "whole_editor")
19              });
20          });
21      }
22  )
```

# TRIGGER THE CODE

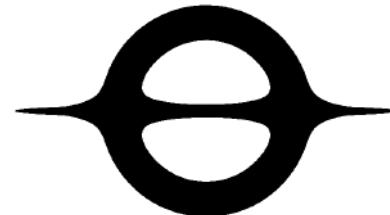
```
1  (
2      s.waitForBoot {
3          ~dirt = SuperDirt(2, s);
4          // More SuperDirt ...
5
6          ~atomOSC = NetAddr("127.0.0.1", 3333);
7
8          MIDIFunc.cc({arg src, chan, num, val;
9              if (num == 60, {
10                  ~atomOSC.sendMsg("/atom/eval", \type, "line");
11              });
12
13              if (num == 61, {
14                  ~atomOSC.sendMsg("/atom/eval", \type, "multi_line");
15              );
16
17              if (num == 62, {
18                  ~atomOSC.sendMsg("/atom/eval", "whole_editor")
19              });
20          });
21      }
22  )
```

# MINIMAL AND EXTENDED SETUP

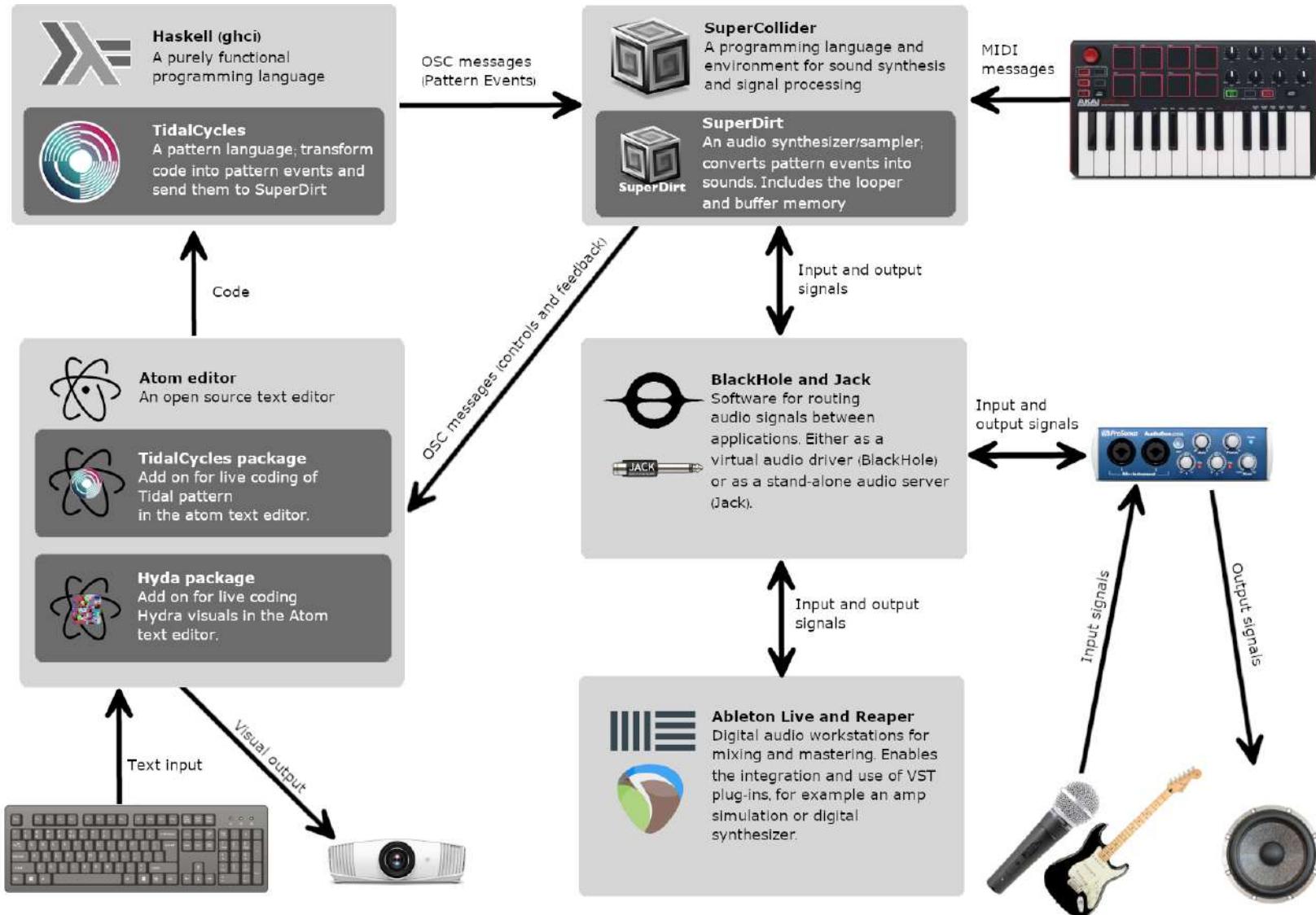
Minimal setup



Extended setup



# COMPLETE ARCHITECTURE



# COMPLETE SETUP





# EXAMPLES AND HINTS

# SWITCHING THE INPUT BUS

```
1 d1 $ s "olooper"
2   # n "< 0 1 2 3 >"
3   # linput "< 14 12 >"
4
5 d2 $ s "loop"
6   # n "< 0 1 2 3 >"
```

# CONTROL THE BUFFER LENGTH

```
1 d1 $ s "rlooper"
2      <| n "< [0 1@3] >"
3      # linput 14
4
5 -- (0>¼) |n 0.0, s: "rlooper"
6 -- (¼>1) |n 1.0, s: "rlooper"
7
8 -- The overdub looper remembers
9 -- only the first buffer length.
```

# (Q)TRIGGER THE LOOPER

```
1 d1 $ qtrigger 1 $ s "rlooper"
2      <| n "< 0 1 2 3 >"
3      # linput 14
```

# USING THE MACRO SEQC

```
1 -- [Optional] adjusted seqP macro
2 -- specialised for the looper
3 seqC x y pt = (p x . ( |< orbit (x-1) ))
4                                     $ qtrigger x
5                                     $ seqP [(0, y, pt)]
6
7 seqC 1 2 $ s "looper" # n "<0 1>"
```

# RESAMPLE A LOOP

```
1 -- You should change the dr sample
2 d1 $ qtrigger 1 $ off (0.5) (#speed 0.5)
3   $ s "dr/2" # legato 1
4
5 seqC 3 2 $ slow 2 $ s "looper"
6   <| n "0 1 2 3 4 5 6 7" # linput 4
7
8 d1 $ s "loop"
9   <| n "[0*4, 5*8, <7 5 2>, 2 8 6 2, 8 7 4 2]"
10
11 seqC 2 2 $ slow 2 $ s "looper"
12   <| n "0 1 2 3 4 5 6 7"
13   # linput 4
14   # lname "v2"
```

# THANK YOU FOR LISTENING

#TIDALLOOPER is a tool  
which provides live  
sampling with TidalCycles  
and is available as a  
SuperCollider Quark.

Have fun! 😊



thgrund



mrreason89



MrReason



mrreason.org

# SOURCES

## Project pages

- Tidal Looper: <https://github.com/thgrund/tidal-looper>
- CycSeq: <https://github.com/thgrund/cycseq>
- TidalCycles: <https://github.com/tidalcycles/Tidal>
- SuperDirt: <https://github.com/musikinformatik/SuperDirt>
- Jack audio server: <https://jackaudio.org>
- Blackhole: <https://existential.audio/blackhole/>

# SOURCES

## Tidal looper performance examples

- Voicing Code - Alex McLean+Eimear O'Donovan:  
<https://www.youtube.com/watch?v=jPCxkZPYX90>
- Eimear + Alex live code vocal experiment #2:  
<https://www.youtube.com/watch?v=K0cArssGCPI>
- Metal edition (or Industrial?) - MrReason:  
<https://www.youtube.com/watch?v=GWUWhOCO-x0>
- Musical Memories - MrReason:  
<https://www.youtube.com/watch?v=OwOAOLRbnKQ>