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

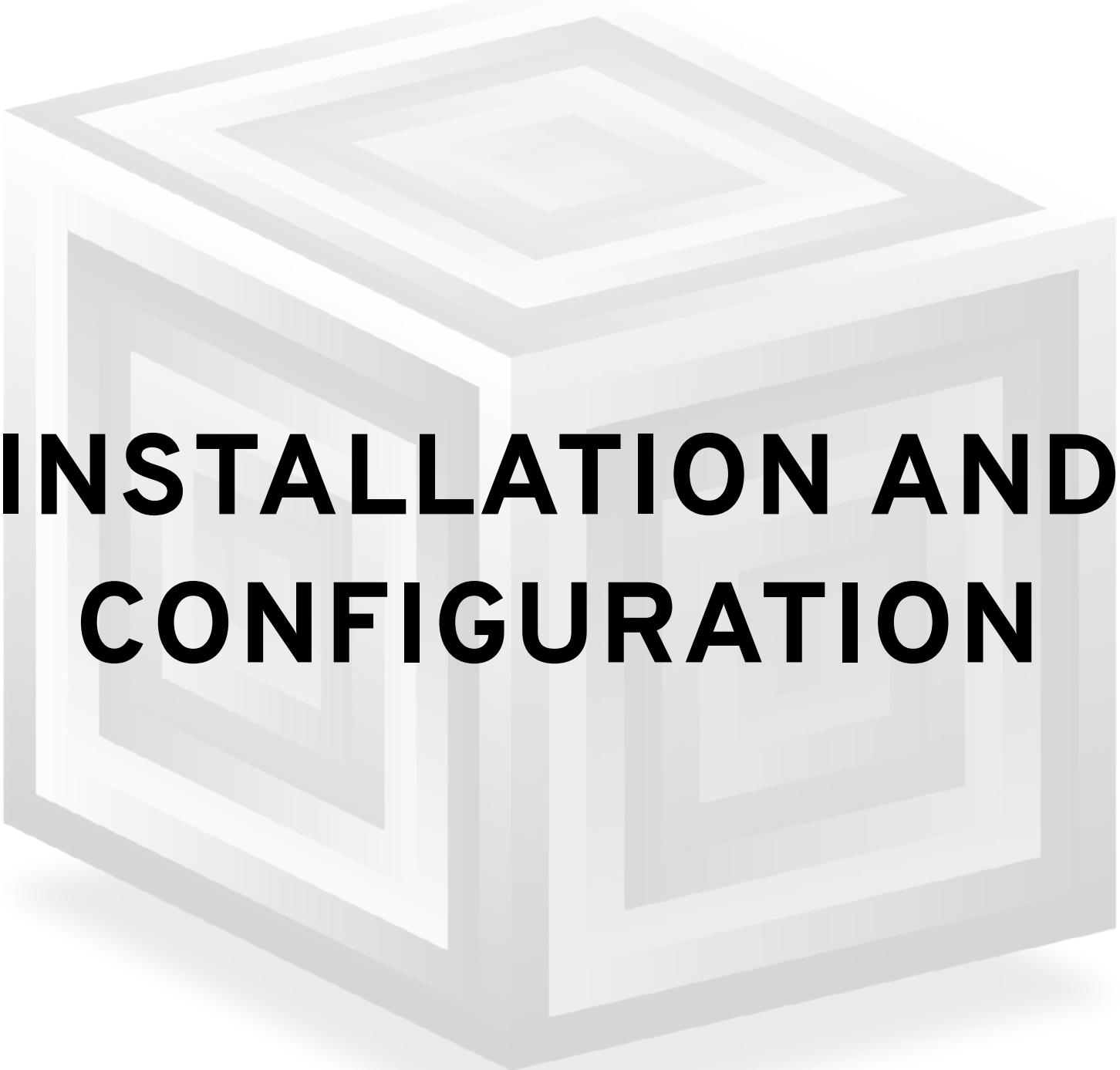

```
1 d1 $ s "looper"  # n "<0 1 2 3>"
2
3 d2 $ s "loop"  # n "[0,1,2,3]"
```

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]"
```

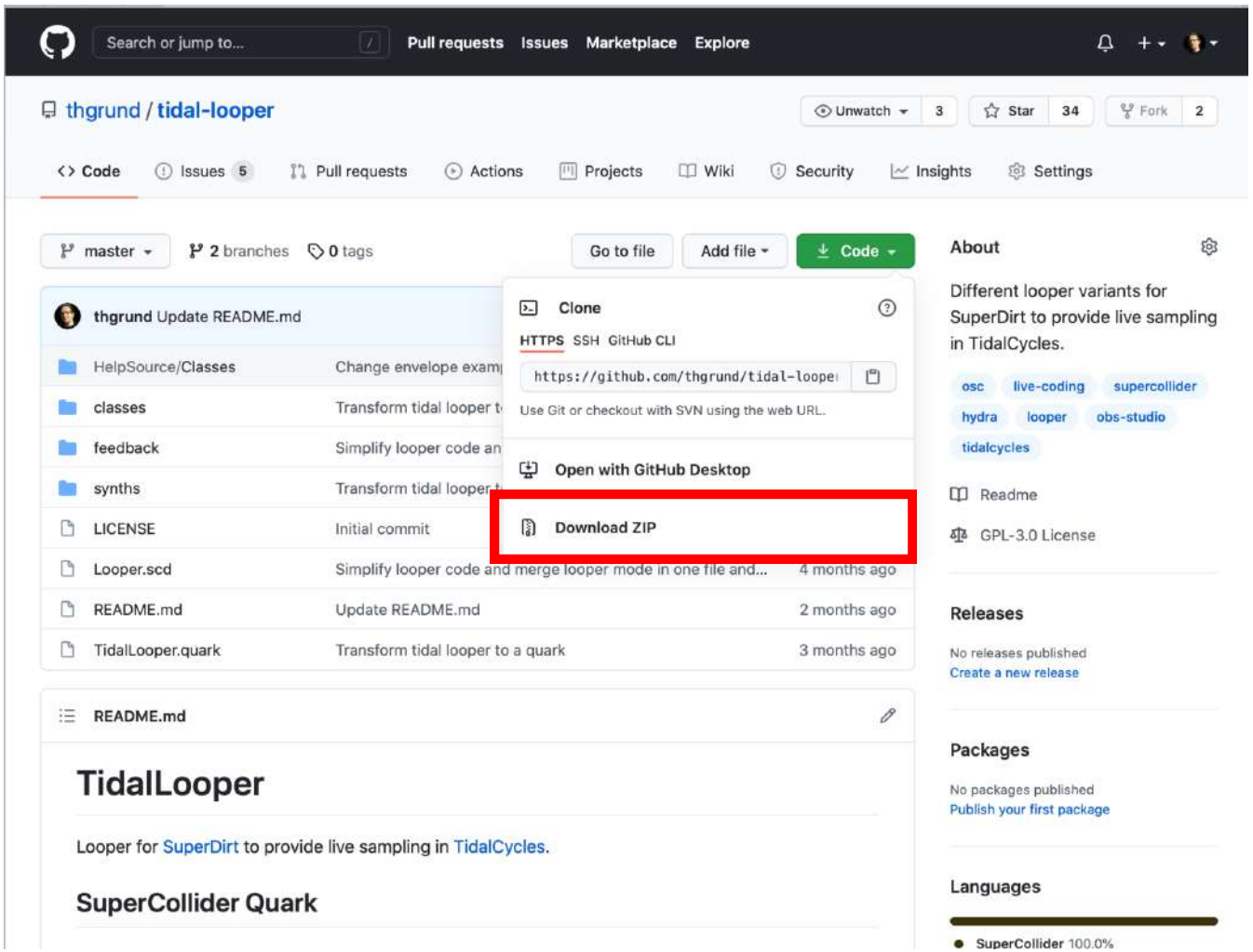
BASIC OVERDUB MODE EXAMPLE

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



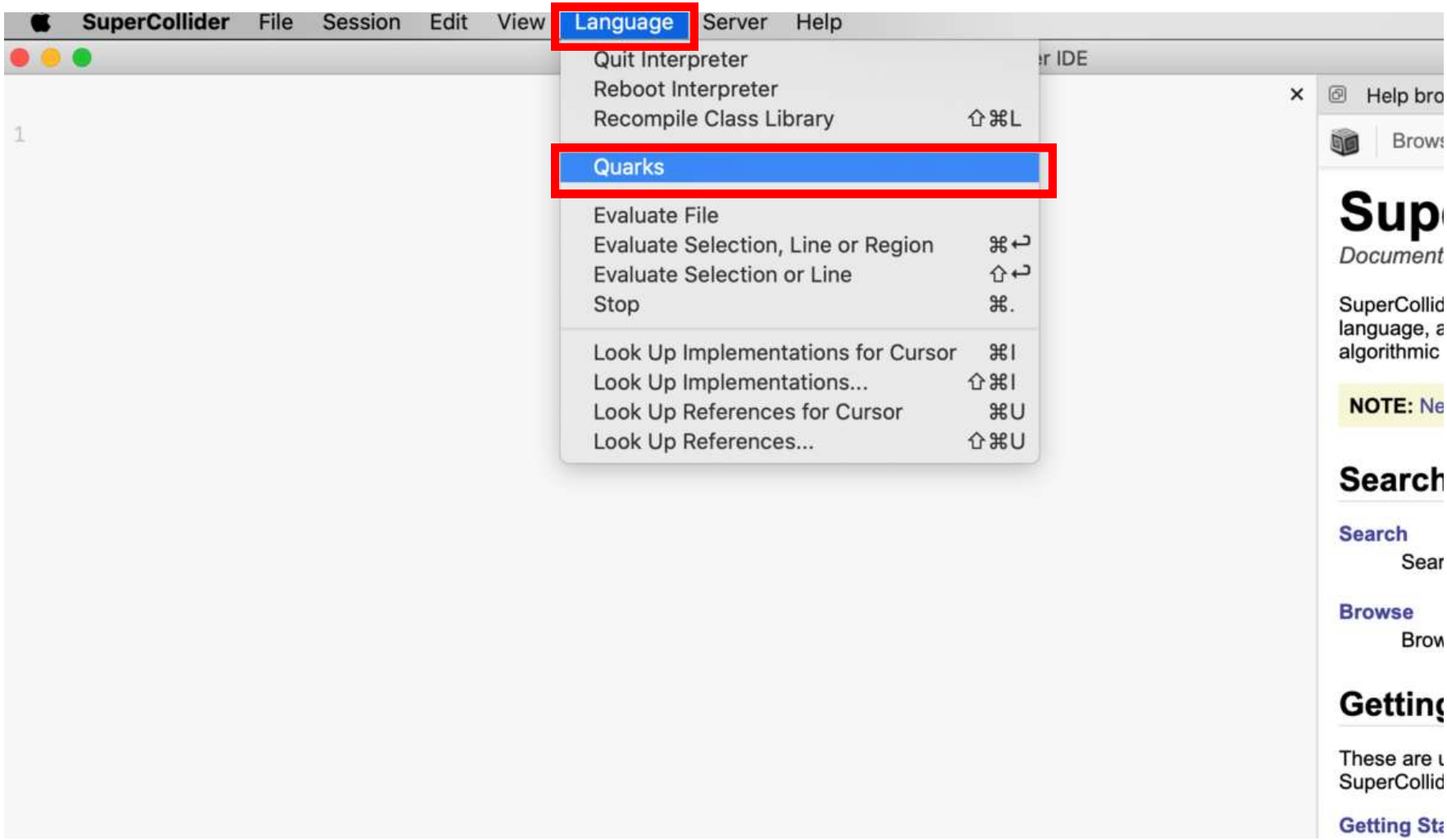
INSTALLATION AND CONFIGURATION

DOWNLOAD FROM GITHUB

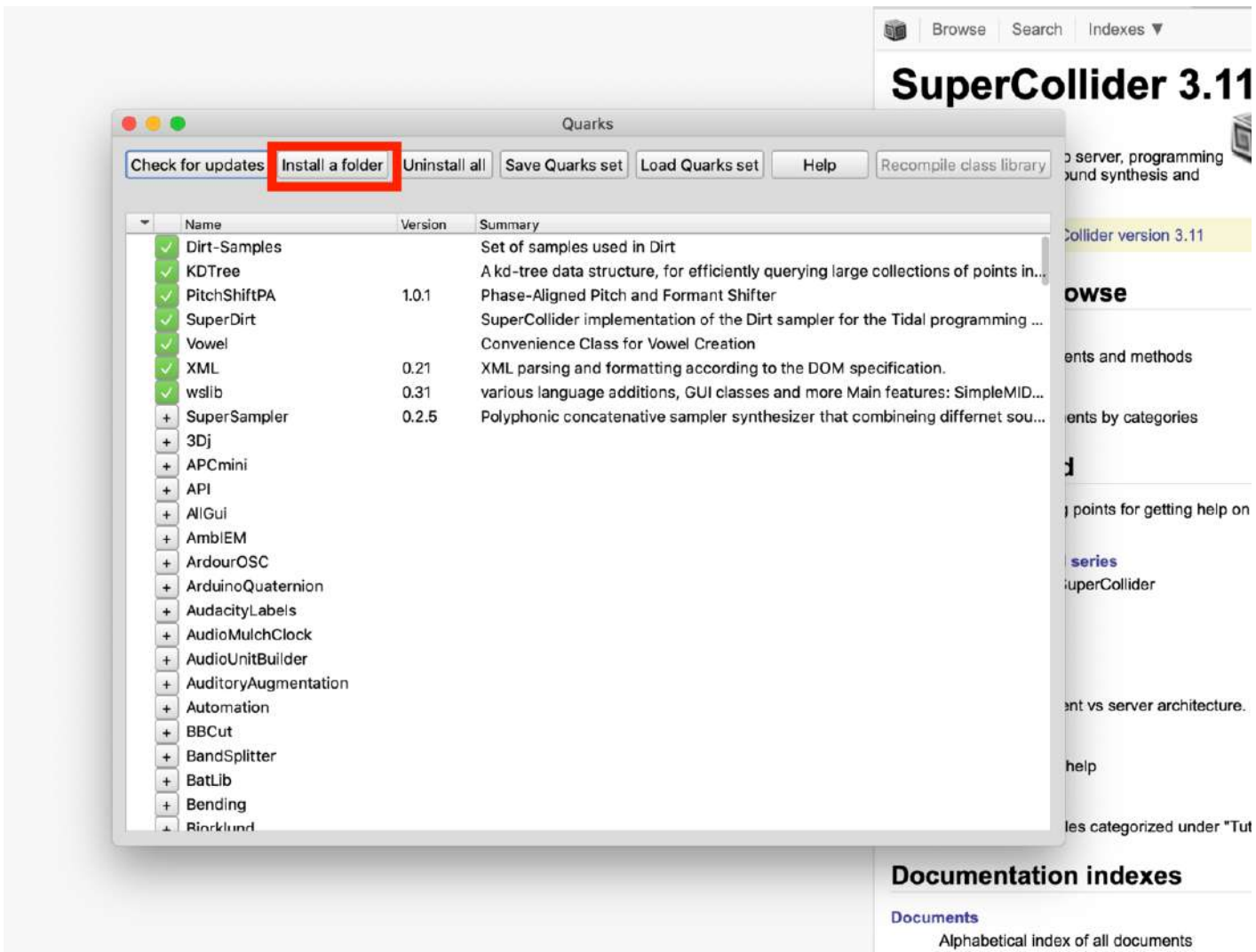


URL: https://github.com/thgrund/tidal-looper

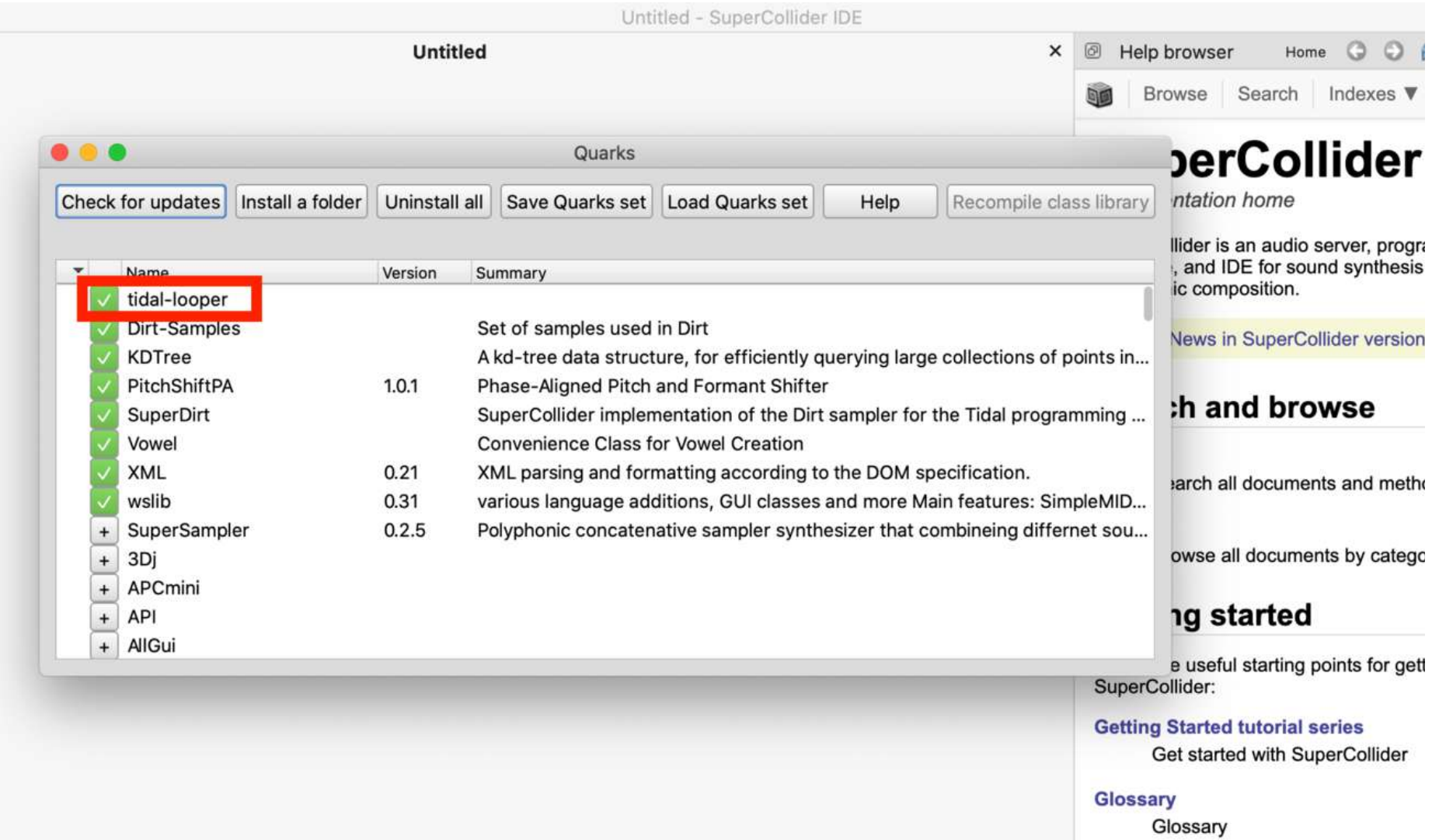
INSTALL THE QUARK I



INSTALL THE QUARK II



INSTALL THE QUARK III



SUPERCOLLIDER STARTUP SCRIPT

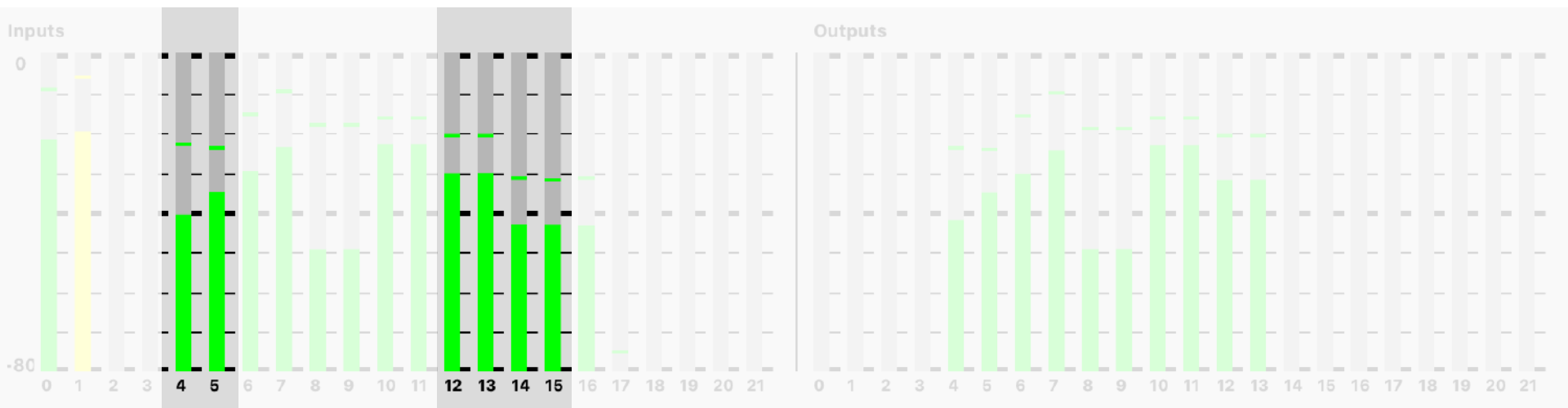
```
(
  s.reboot {
      s.options.numBuffers = 1024 * 256;
      s.options.memSize = 8192 * 32;
      s.options.numWireBufs = 512;
      s.options.maxNodes = 1024 * 32;
      s.options.numOutputBusChannels = 22;
      s.options.numInputBusChannels = 22;

      s.waitForBoot {
          ~dirt = SuperDirt(2, s);
          ~dirt.loadSoundFiles;

          ~dirt.start(57120, [4, 6, 8, 10, 12, 14, 16, 18, 20]);

          // optional, needed for convenient access from sclang:
          (
              ~d1 = ~dirt.orbits[0]; ~d2 = ~dirt.orbits[1]; ~d3 = ~dirt.orbits[2];
              ~d4 = ~dirt.orbits[3]; ~d5 = ~dirt.orbits[4]; ~d6 = ~dirt.orbits[5];
              ~d7 = ~dirt.orbits[6]; ~d8 = ~dirt.orbits[7]; ~d9 = ~dirt.orbits[8];
              ~d10 = ~dirt.orbits[9]; ~d11 = ~dirt.orbits[10]; ~d12 = ~dirt.orbits[11];
          );
      };

      s.latency = 0.2;

      ~looper = TidalLooper(~dirt);
  }
)
```

SUPERCOLLIDER STARTUP SCRIPT

```
(
  s.reboot {
      s.options.numBuffers = 1024 * 256;
      s.options.memSize = 8192 * 32;
      s.options.numWireBufs = 512;
      s.options.maxNodes = 1024 * 32;
      s.options.numOutputBusChannels = 22;
      s.options.numInputBusChannels = 22;

      s.waitForBoot {
          ~dirt = SuperDirt(2, s);
          ~dirt.loadSoundFiles;

          ~dirt.start(57120, [4, 6, 8, 10, 12, 14, 16, 18, 20]);

          // optional, needed for convenient access from sclang:
          (
              ~d1 = ~dirt.orbits[0]; ~d2 = ~dirt.orbits[1]; ~d3 = ~dirt.orbits[2];
              ~d4 = ~dirt.orbits[3]; ~d5 = ~dirt.orbits[4]; ~d6 = ~dirt.orbits[5];
              ~d7 = ~dirt.orbits[6]; ~d8 = ~dirt.orbits[7]; ~d9 = ~dirt.orbits[8];
              ~d10 = ~dirt.orbits[9]; ~d11 = ~dirt.orbits[10]; ~d12 = ~dirt.orbits[11];
          );
      };

      s.latency = 0.2;

      ~looper = TidalLooper(~dirt);
  }
)
```

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

CONFIGURE THE DEFAULT VALUES

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

CONFIGURE THE DEFAULT VALUES

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

CONFIGURE THE DEFAULT VALUES

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

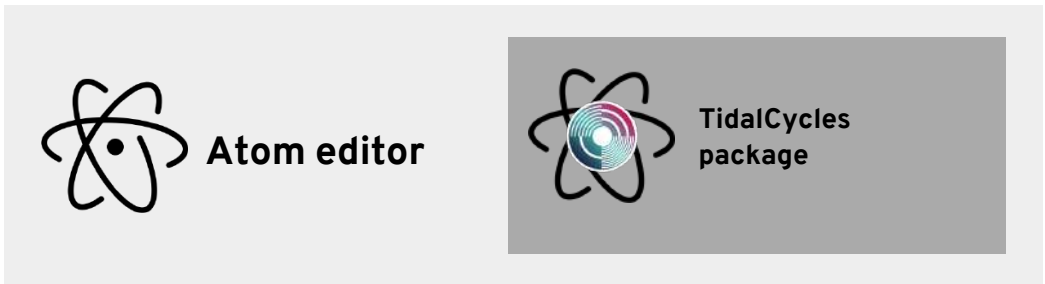
```
1 -- You could add this in BootTidal.hs
2
3 -- Change input bus
4 linput = pI "linput"
5
6 -- Change the buffer name
7 lname = pS "lname"
```

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"
```

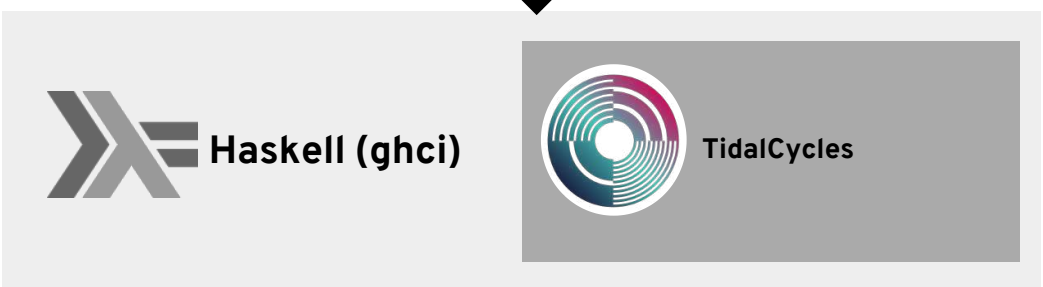
TECHNICAL SETUP

CONTROL THE RECORDING



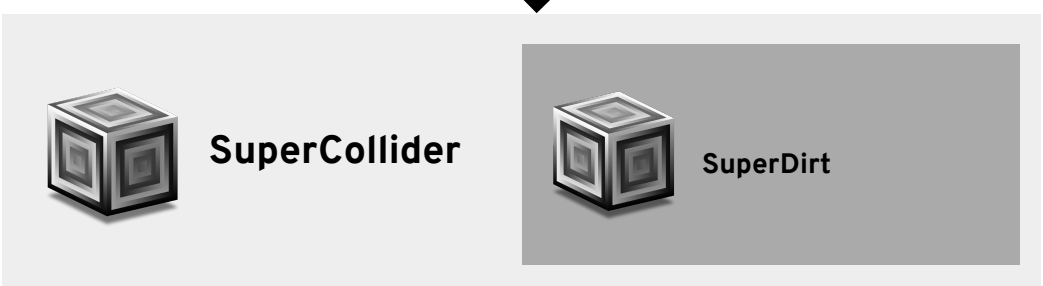

```
d1 $ s "looper"
   # n "<0 1 2 3>"
```

Code




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

OSC messages
(Pattern Events)




```
/*
Includes the looper and
buffer memory.

Processes the requests.
*/
```

RECORD AN INPUT SIGNAL

Input source

Input signal

Audio Routing with BlackHole or Jack

Input signal

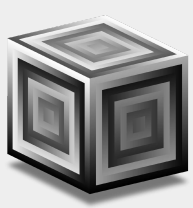
Modified Input signal

DAW (Ableton)

Audio Effects

(i.e. Amp Simulation)

(Modified) input signal



Record the audio signal and make it accessible

RECORD A VST SYNTHESIZER

MIDI messages

DAW (Ableton)

Software synthesizer

as VST plugin

Synthesizer signal

Audio Routing with

BlackHole or Jack

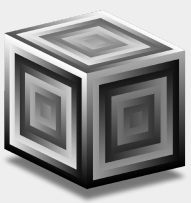
Synthesizer signal

Record the audio signal and make it accessible

RECORD A TIDAL PATTERN

Send TidalCycles patterns as OSC events

OSC message

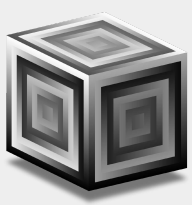


Create audio result SuperDirt

Output signal

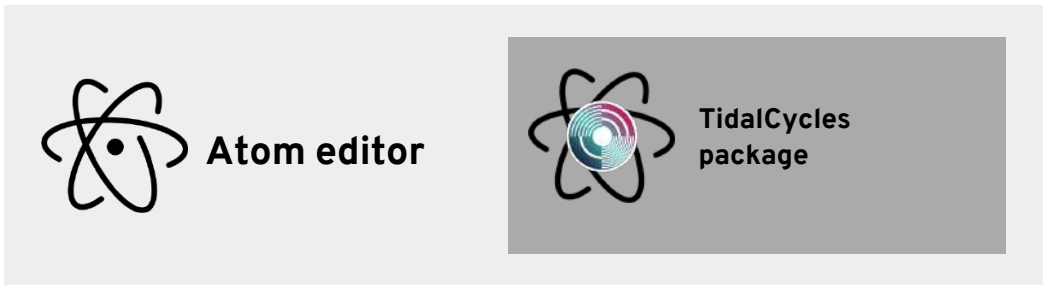
Audio Routing with BlackHole or Jack

Input signal



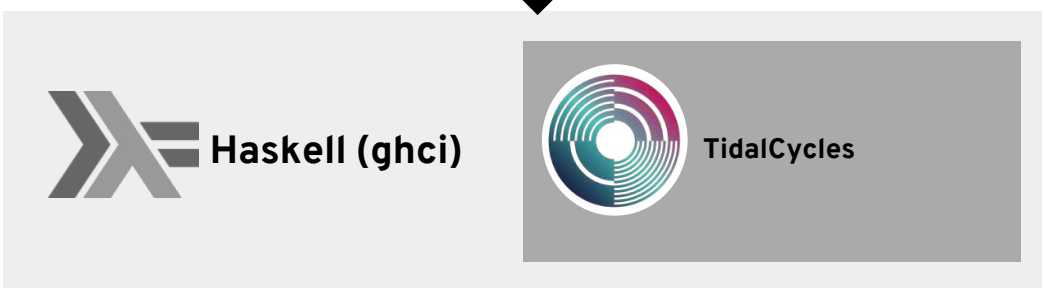
Record the audio signal and make it accessible

CONTROL THE PLAYBACK



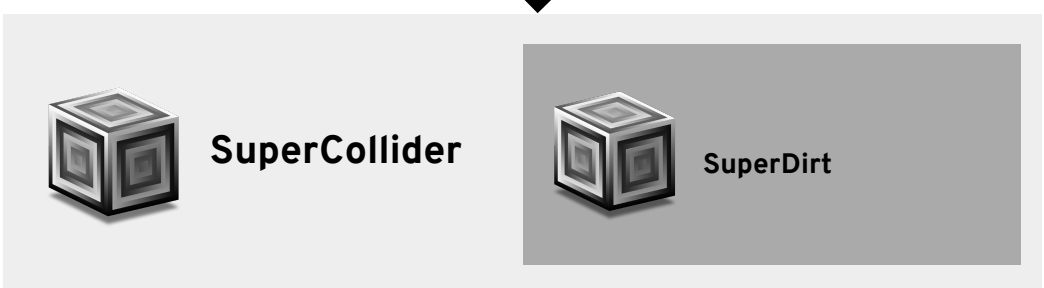

```
d1 $ s "loop"
   # n "[0,1,2,3]"
```

Code



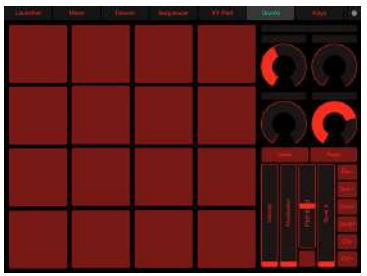

```
[ /play2, cps, 0.5625,
  cycle, 23.0, orbit, 1,
  delta, 0.59259271,
  n, 3.0, s, loop ]
```

OSC messages
(Pattern Events)




```
/*
Includes the looper and
buffer memory.

Processes the requests.
*/
```

TRIGGER THE CODE



Send keyboard shortcuts

Send MIDI

Send OSC

MIDI To OSC

with manual mapping

Send OSC

Trigger code

Single Line

Multi Line

Whole editor

TRIGGER THE CODE

```
(
    s.waitForBoot {
        ~dirt = SuperDirt(2, s);
        // More SuperDirt ...

        ~atomOSC = NetAddr("127.0.0.1", 3333);

        MIDIFunc.cc({arg src, chan, num, val;
            if (num == 60, {
                ~atomOSC.sendMsg("/atom/eval", \type, "line");
            });

            if (num == 61, {
                ~atomOSC.sendMsg("/atom/eval", \type, "multi_line");
            });

            if (num == 62, {
                ~atomOSC.sendMsg("/atom/eval", "whole_editor")
            });
        });
    }
)
```

TRIGGER THE CODE

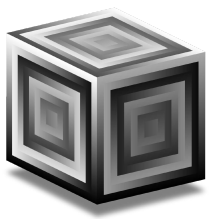
```
(
    s.waitForBoot {
        ~dirt = SuperDirt(2, s);
        // More SuperDirt ...

        ~atomOSC = NetAddr("127.0.0.1", 3333);

        MIDIFunc.cc({arg src, chan, num, val;
            if (num == 60, {
                ~atomOSC.sendMsg("/atom/eval", \type, "line");
            });

            if (num == 61, {
                ~atomOSC.sendMsg("/atom/eval", \type, "multi_line");
            });

            if (num == 62, {
                ~atomOSC.sendMsg("/atom/eval", "whole_editor")
            });
        });
    }
)
```

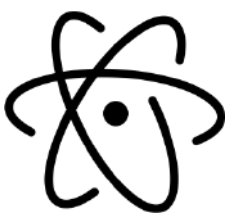
TRIGGER THE CODE

```
(
    s.waitForBoot {
        ~dirt = SuperDirt(2, s);
        // More SuperDirt ...

        ~atomOSC = NetAddr("127.0.0.1", 3333);

        MIDIFunc.cc({arg src, chan, num, val;
            if (num == 60, {
                ~atomOSC.sendMsg("/atom/eval", \type, "line");
            });

            if (num == 61, {
                ~atomOSC.sendMsg("/atom/eval", \type, "multi_line");
            });

            if (num == 62, {
                ~atomOSC.sendMsg("/atom/eval", "whole_editor")
            });
        });
    }
)
```

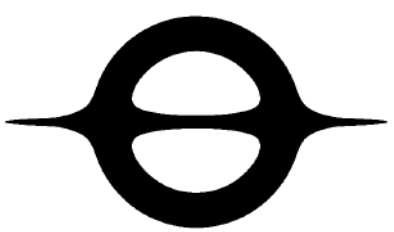
MINIMAL AND EXTENDED SETUP

Minimal setup

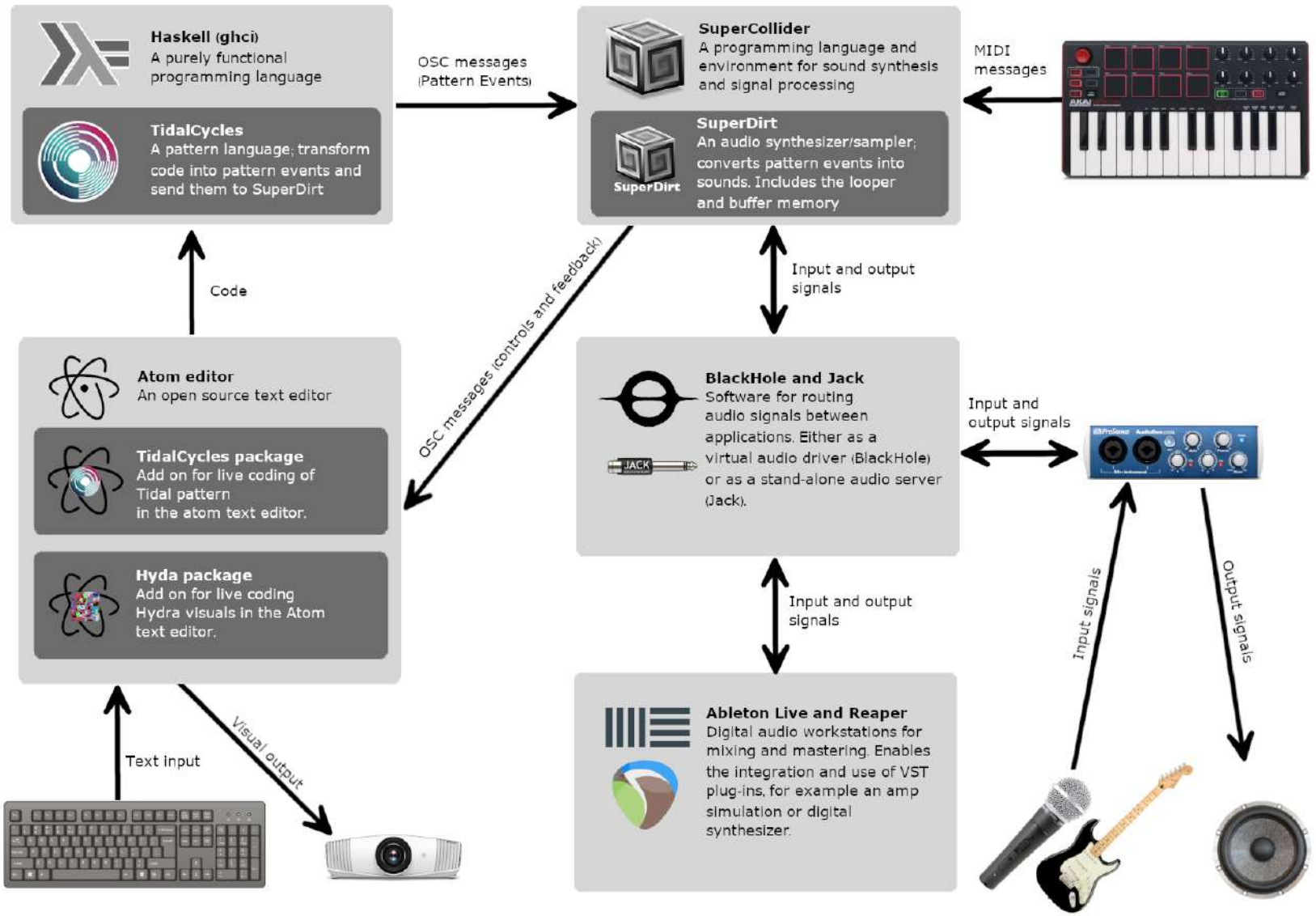




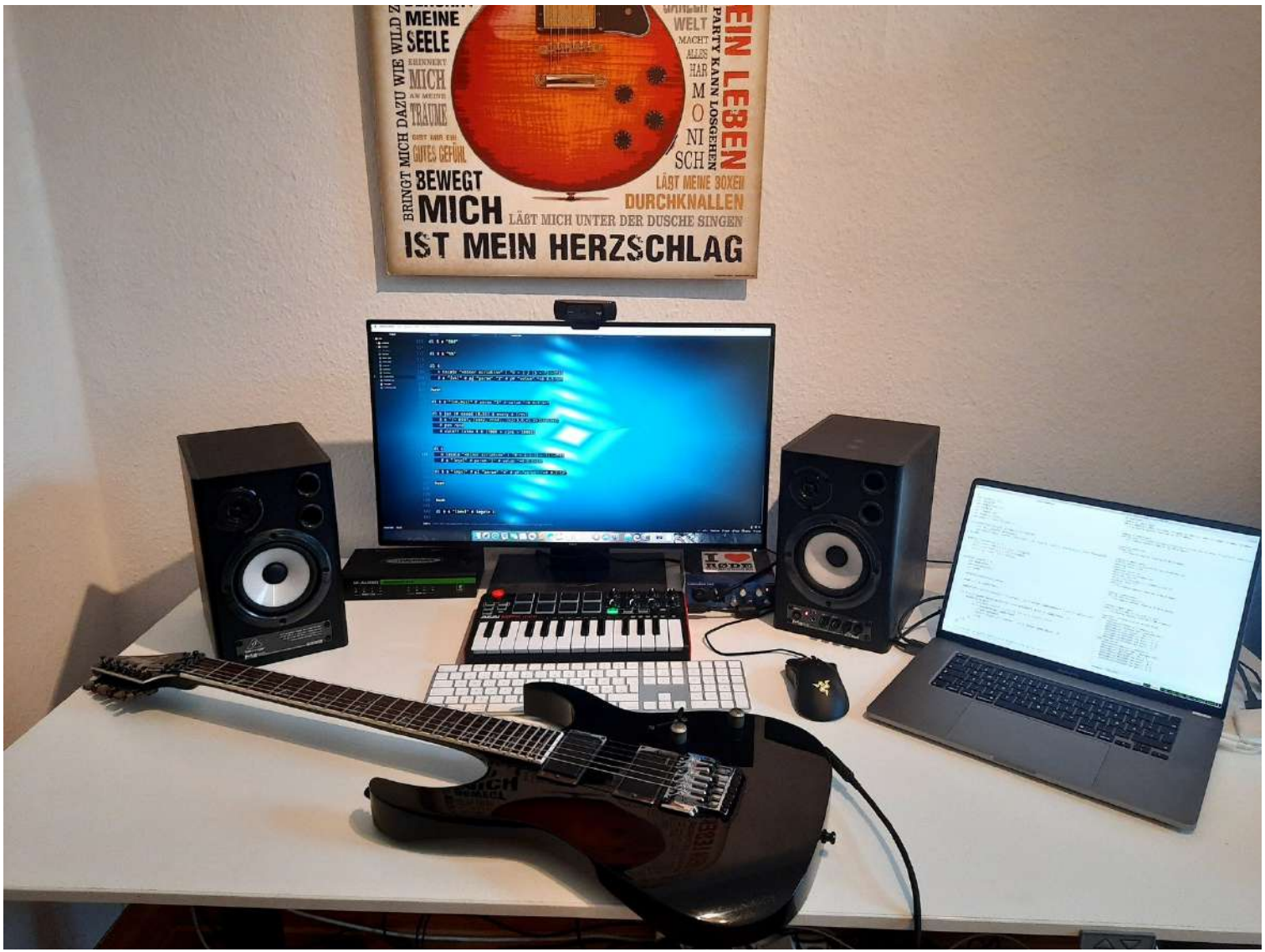
Extended setup



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