Super Collider 3 für Einsteiger

Sommersemester 1

Dr. Chikashi Miyama

16. April 2012

Die Website des Kurses

- http://sc3.chikashi.net
  - Präsentationen (PDF)
  - HA
  - Lösungsschlüssel
  - Soundfiles
  - Links

Kurs

- 14:00 - 16:00

  Grundlage

  Musikalische Anwendungen / Tipps

- 16:00 - 18:00

  Fragen

  Projektvorbereitung

Überblick

Routine

A. *Basis*

B. *Arrays und Routine*

C. *Zufälligkeit und Routine*

D. *Drum Sequencer mit Routine*

E. *Musikalische Anwendungen für Live-Elektronik Stücke*

F. *Demo Stück „DSP of Fury“ von Andrew Babcock mit Quellcode*

SC3 kann auf Englisch sprechen!

```
"I am studying SC3".speak
```

Man kann zwei Strings (Sätze) mit „+“ verbinden

```
("I am studying SC3 now" + "but you are watching TV").speak
```

SC3 wählt ein Subjekt aus

```
(["she", "Michael", "Chikashi"].choose + "is eating a
bread.").speak
```

SC3 wählt ein Subjekt, Verb und Objekt aus

```
(  ["she", "chikashi", "Michael"].choose +
   "is"+
   ["teaching", "singing", "driving", "eating", "doing"].choose +
   ["music", "a song", "a car", "an egg","homework"].choose ).speak
```

A. *Basis*

Routine A.1

```
Routine({
   "I am".postln;
   1.0.wait;
   "eating".postln;
   2.0.wait;
   "an egg".postln;
}).play;
```

1 Sekunde später

2 Sekunde später

mit „speak“

```
Routine({
   "I am".speak;
   1.0.wait;
   "eating".speak;
   2.0.wait;
   "an egg".speak;
}).play;
```

1 Sekunde später

2 Sekunde später

Routine A.2

Mit Synth

```
SynthDef("mySynth",{
   arg freq, amp;
   e = Env.new([0, 1, 1 ,0], [0.01, 0.1, 0.01]);
   a = Saw.ar(freq, amp) * EnvGen.ar(e, doneAction:2);
   Out.ar(0, a );
}).load;


Routine({
   Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
   1.0.wait;
   Synth("mySynth",["freq", 64.midicps, "amp", 0.3]);
   1.0.wait;
   Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
}).play;
```

Routine A.3

- Infinite Loop (Endlosschleife)

```
Routine({
   loop{
      Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 72.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 68.midicps, "amp", 0.3]);
      0.1.wait;
   };
}).play;
```

Routine A.4

- Wie kann man die Endlosschreife stoppen?

Man muss zuerst die Routine benennen

```
r = Routine({
   loop{
      Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 72.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 68.midicps, "amp", 0.3]);
      0.1.wait;
   };
}).play;
```

```
r.stop;
```

stopp!

Routine A.5

```
r = Routine({
   loop{
      Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 72.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 68.midicps, "amp", 0.3]);
      0.1.wait;
   };
});
```

`r.play;` ← Wiedergeben

`r.stop;` ← Stoppen

`r.reset;` ← Zurückspulen

Routine A.6

- Nur 3 mal

```
Routine({
   3.do{
      Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 72.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
      0.1.wait;
      Synth("mySynth",["freq", 68.midicps, "amp", 0.3]);
      0.1.wait;
   };
}).play;
```

Routine A.7

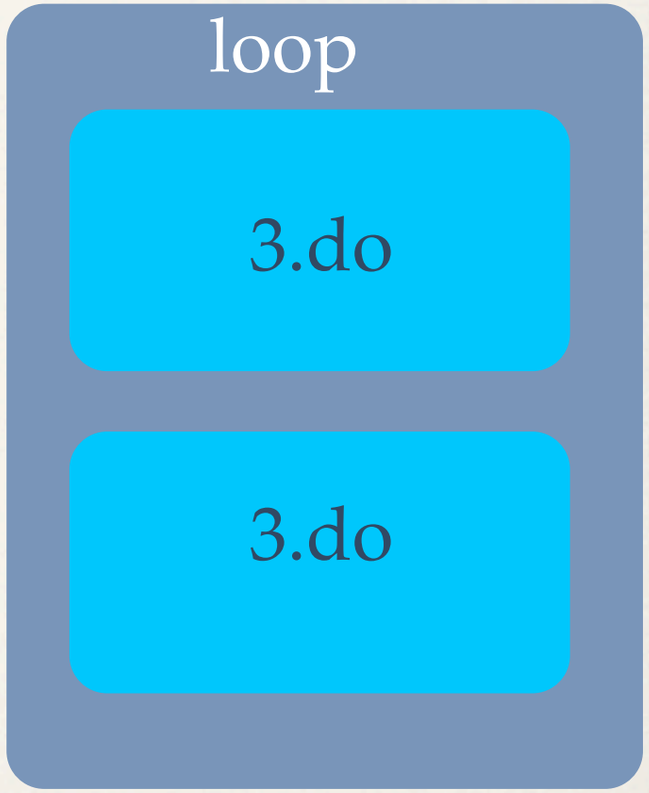
- Zwei Loops

```
Routine({
    3.do{
        Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
        0.1.wait;
        Synth("mySynth",["freq", 72.midicps, "amp", 0.3]);
        0.1.wait;
        Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
        0.1.wait;
        Synth("mySynth",["freq", 68.midicps, "amp", 0.3]);
        0.1.wait;
    };
    3.do{
        Synth("mySynth",["freq", 61.midicps, "amp", 0.3]);
        0.1.wait;
        Synth("mySynth",["freq", 72.midicps, "amp", 0.3]);
        0.1.wait;
        Synth("mySynth",["freq", 68.midicps, "amp", 0.3]);
        0.1.wait;
        Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
        0.1.wait;
    };
}).play;
```

Routine A.8

- Loop in Loop (Nested Loop)

```
Routine({
    loop{
        3.do{
            Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
            0.1.wait;
            Synth("mySynth",["freq", 72.midicps, "amp", 0.3]);
            0.1.wait;
            Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
            0.1.wait;
            Synth("mySynth",["freq", 68.midicps, "amp", 0.3]);
            0.1.wait;
        };
        3.do{
            Synth("mySynth",["freq", 61.midicps, "amp", 0.3]);
            0.1.wait;
            Synth("mySynth",["freq", 72.midicps, "amp", 0.3]);
            0.1.wait;
            Synth("mySynth",["freq", 68.midicps, "amp", 0.3]);
            0.1.wait;
            Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
            0.1.wait;
        };
    };
}).play;
```



Routine A.9

- Man kann zwei *Routines* gleichzeitig verwenden.

```
(
   Routine({
      loop{
         Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
         0.1.wait;
         Synth("mySynth",["freq", 72.midicps, "amp", 0.3]);
         0.1.wait;
         Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
         0.1.wait;
         Synth("mySynth",["freq", 68.midicps, "amp", 0.3]);
         0.1.wait;
      };
   }).play;

   Routine({
      loop{
         Synth("mySynth",["freq", 67.midicps, "amp", 0.3]);
         0.2.wait;
         Synth("mySynth",["freq", 63.midicps, "amp", 0.3]);
         0.2.wait;
      };
   }).play;
)
```

B. *Arrays und Routine*

Routine B.1

- Mit ein Array kann man viele Zahlen zusammenstellen.

```
[3, 11, -2, 4, 9];
```

Ein Array mit 5 Zahlen

- Man kann ein Array benennen

```
a = [3, 11, -2, 4, 9];
```

Das Array heißt *a*

- Jedes Element des Arrays hat *index number*

```
a = [3, 11, -2, 4, 9];
```

0 1 2 3 4 ⟵ index number

- Man kann ein Element des Arrays mit einer *index number* abholen.

```
a = [3, 11, -2, 4, 9];
```

```
a[3] ... 4
```

```
a.at(2) ... -2
```

Routine B.2

- Wiederholung
- *do*

```
5.do{
   "chikashi".postln;
}
```

5 mal wiederholen

- *arg index;*

```
5.do{
   arg index;
   index.postln;
}
```

erste Wiederholung
index = 0
zweite Wiederholung
index = 1
ditte Wiederholung
index = 2
usw...

Routine B.3

- Melodie in einem Array

D D D A B C D C B A

```
a = [74, 74, 74, 69, 71, 73, 74, 73, 71, 69];
Routine({
   10.do{
      arg index;
      Synth("mySynth",["freq", a[index].midicps, "amp", 0.3]);
      0.3.wait;
   };
}).play
```

Ein‘ feste Burg ist unsere SC...

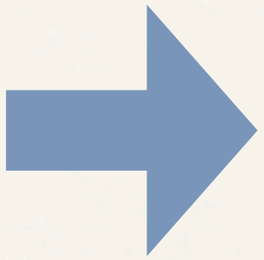
Routine B.4

- Rhythmus in einem Array

```
a = [74, 74, 74, 69, 71, 73, 74, 73, 71, 69];
r = [1, 1, 1, 0.5, 0.5, 1, 0.5, 0.5, 1, 1];
Routine({
   10.do{
      arg index;
      Synth("mySynth",["freq", a[index].midicps, "amp", 0.3]);
      r[index].wait;
   };
}).play
```

Routine B.5

```
[ 2, 4, 3, 5 ] - 1
```



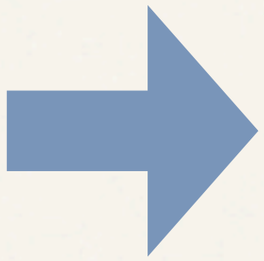
```
[ 1, 3, 2, 4 ]
```

- Transposition

```
a = [74, 74, 74, 69, 71, 73, 74, 73, 71, 69] - 2;
r = [1, 1, 1, 0.5, 0.5, 1, 0.5, 0.5, 1, 1];
Routine({
   10.do{
      arg index;
      Synth("mySynth",["freq", a[index].midicps, "amp", 0.3]);
      r[index].wait;
   };
}).play
```

Routine B.6

```
[ 2, 4, 3, 5 ] * 0.5
```



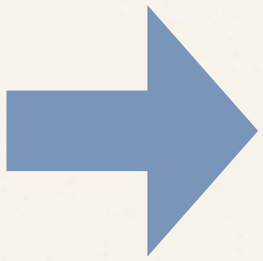
```
[ 1, 2, 1.5, 2.5 ]
```

- Schneller (oder langsamer)

```
a = [74, 74, 74, 69, 71, 73, 74, 73, 71, 69];
r = [1, 1, 1, 0.5, 0.5, 1, 0.5, 0.5, 1, 1] * 0.2;
Routine({
   10.do{
      arg index;
      Synth("mySynth",["freq", a[index].midicps, "amp", 0.3]);
      r[index].wait;
   };
}).play
```

Routine B.5

```
[ 2, 4, 3, 5 ].replace(4, 10)
```



```
[ 2, 10, 3, 5 ]
```

Ersetze H mit B

- D-Moll

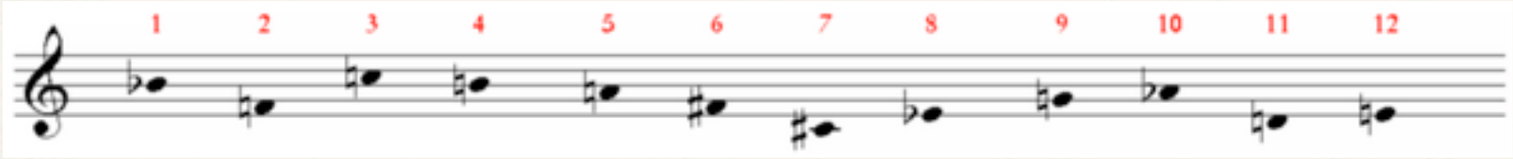
70 = B, 71 = H

```
a = [74, 74, 74, 69, 71, 73, 74, 73, 71, 69].replace(71, 70);
r = [1, 1, 1, 0.5, 0.5, 1, 0.5, 0.5, 1, 1];
Routine({
   10.do{
      arg index;
      Synth("mySynth",["freq", a[index].midicps, "amp", 0.3]);
      r[index].wait;
   };
}).play
```

Routine B.6

- Zwölftontechnik

Schönberg Op. 33 Klavierstück



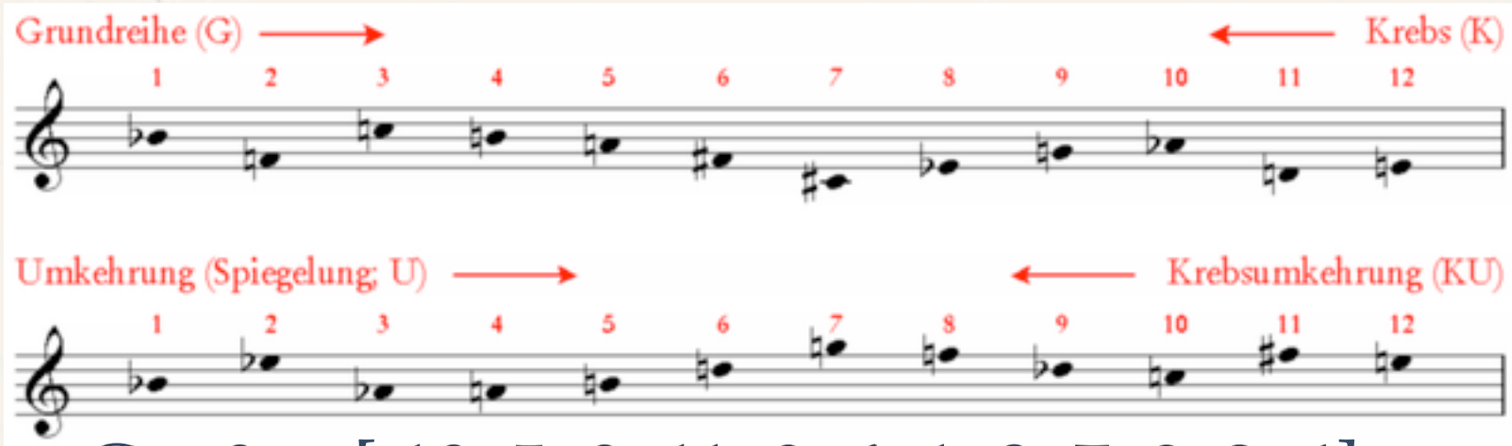
C = 0 ... [10, 5, 0, 11, 9, 6, 1, 3, 7, 8, 2, 4]

```
a = [10, 5, 0, 11, 9, 6, 1, 3, 7, 8, 2, 4];
Routine({
   12.do({
      arg index;
      Synth("mySynth",["freq", (a[index] + 72).midicps, "amp", 0.3]);
      0.2.wait;
   });
}).play
```

Routine B.7

`[ 2, 4, 3, 5 ].reverse` → `[ 5, 3, 4 ,2 ]`

- Krebs

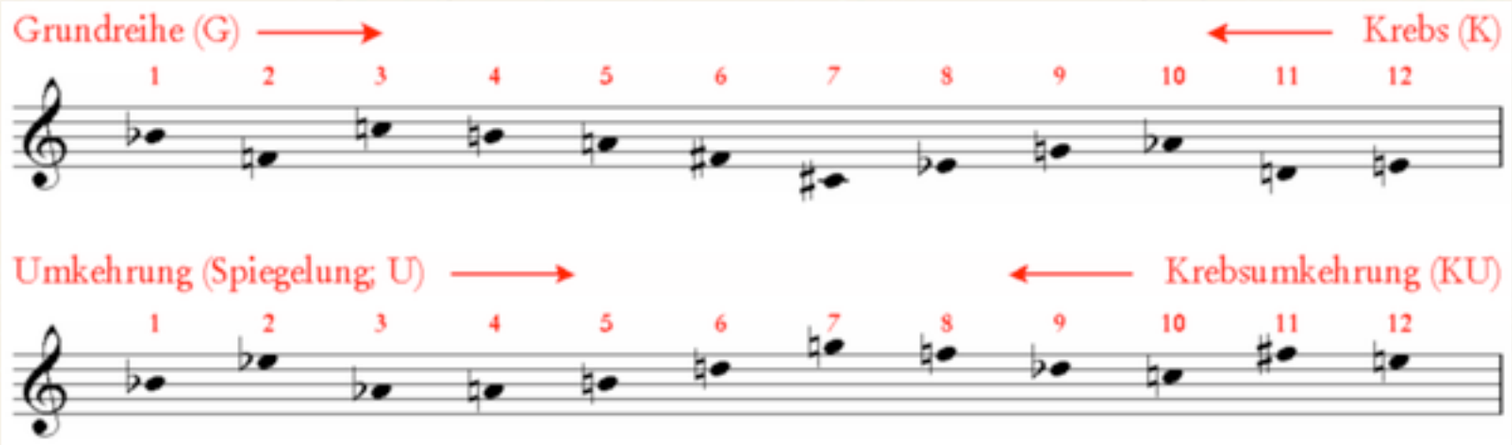


C = 0 ... [10, 5, 0, 11, 9, 6, 1, 3, 7, 8, 2, 4]

```
a = [10, 5, 0, 11, 9, 6, 1, 3, 7, 8, 2, 4].reverse;
Routine({
   12.do({
      arg index;
      Synth("mySynth",["freq", (a[index] + 72).midicps, "amp", 0.3]);
      0.2.wait;
   });
}).play()
```

Routine B.8

- Umkehrung



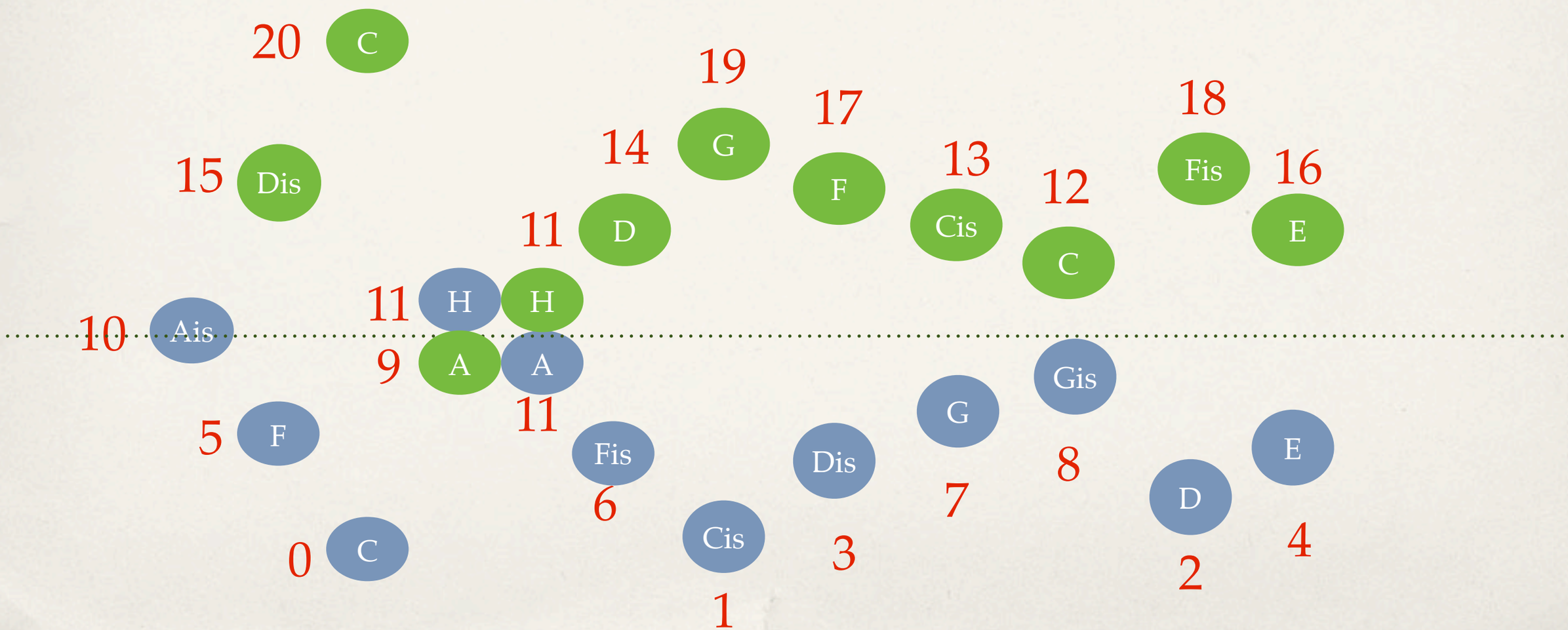
C = 0 ... [10, 5, 0, 11, 9, 6, 1, 3, 7, 8, 2, 4]

```
a = 20-[10, 5, 0, 11, 9, 6, 1, 3, 7, 8, 2, 4];
Routine({
   12.do({
      arg index;
      Synth("mySynth",["freq", (a[index] + 72).midicps, "amp", 0.3]);
      0.2.wait;
   });
}).play
```

Routine B.9

- Umkehrung

```
20 - [10, 5, 0, 11, 9, 6, 1, 3, 7, 8, 2, 4]
...[ 10, 15, 20, 9, 11, 14, 19, 17, 13, 12, 18, 16 ]
```



Routine B.10

```
a = [10, 5, 0, 11, 9, 6, 1, 3, 7, 8, 2, 4];
b = a.reverse;
c = 20-a;
d = c.reverse;

Routine({
    a.do{
      arg item;
        Synth("mySynth",["freq", (item + 72).midicps, "amp", 0.3]);
        0.2.wait;
    };
    b.do{
      arg item;
        Synth("mySynth",["freq", (item + 72).midicps, "amp", 0.3]);
        0.2.wait;
    };
    c.do{
      arg item;
        Synth("mySynth",["freq", (item + 72).midicps, "amp", 0.3]);
        0.2.wait;
    };
    d.do{
      arg item;
        Synth("mySynth",["freq", (item + 72).midicps, "amp", 0.3]);
        0.2.wait;
    };
}).play
```

Grundreihe

Krebs

Umkehrung

Krebsumkehrung

C. *Zufälligkeit und Routine*

Routine C.1

- Zufälligkeit /Random

```
10.rand;
```

generiert eine Zahl zwischen 0 und 9

```
100.rand;
```

generiert eine Zahl zwischen 0 und 99

```
10.rand2;
```

generiert eine Zahl zwischen -10 und 10

```
10.rrand(20);
```

generiert eine Zahl zwischen 10 und 20

```
rrand(10, 20);
```

Man kann auch so schreiben

Routine C.2

```
Routine({
   loop{
      Synth("mySynth",["freq", 127.rand.midicps, "amp", 0.3]);
      0.1.wait;
   };
}).play;
```

Zwischen C4 und C5

```
Routine({
   loop{
      Synth("mySynth",["freq", 60.rrand(72).midicps, "amp", 0.3]);
      0.1.wait;
   };
}).play;
```

Routine C.3

- Beschränkte Zufälligkeit

```
[60, 61, 64, 67, 69].choose
```

wählt eine Zahl aus

```
Routine({

   loop{
      Synth("mySynth",["freq", [60, 61, 64, 67, 69].choose.midicps, "amp", 0.3]);
      0.1.wait;
   };

}).play;
```

Routine C.4

- Wahrscheinlichkeit

```
[60, 64].choose
```

50 % 50 %

```
[60, 64, 69].choose
```

33 % 33 % 33 %

```
[60, 64].wchoose([0.1, 0.9])
```

10 % 90 %

```
[60, 64, 69].wchoose([0.1, 0.8, 0.1])
```

10 % 80 % 10 %

Routine C.5

```
Routine({
   loop{
      Synth("mySynth",["freq", [60, 64, 67].choose.midicps, "amp", 0.3]);
      0.1.wait;
   };
}).play;
```

33% 33% 33%

```
Routine({
   loop{
      Synth("mySynth",["freq", [60, 64, 67].wchoose([0.1, 0.1, 0.8]).midicps, "amp", 0.3]);
      0.1.wait;
   };
}).play;
```

10% 10% 80%

Routine C.6

- Rhythmus mit Zufälligkeit

```
Routine({
   loop{
      Synth("mySynth",["freq", [60, 61, 64, 67, 69].choose.midicps, "amp", 0.3]);
      [0.1, 0.2, 0.3, 0.4].choose.wait;
   };
}).play;
```

Routine C.7

- Rit / Accel.

```
Routine({
   x = 1.0;
   20.do{
      Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
      x.wait;
      x = x * 0.85;
   }
}).play;
```

```
Routine({
   x = 0.1;
   20.do{
      Synth("mySynth",["freq", 60.midicps, "amp", 0.3]);
      x.wait;
      x = x * 1.25;
   }
}).play;
```

D. *Drum Sequencer mit Routine*

Routine D.1

- Drum Samples

```
c = Buffer.read(s, "sounds/closed.wav");
k = Buffer.read(s, "sounds/kick.wav");
d = Buffer.read(s, "sounds/snare.wav");

SynthDef("sample",{
   arg bufnum;
   a = PlayBuf.ar(1, bufnum, doneAction:2 );
   Out.ar(0, a);
}).load;

Synth("sample", ["bufnum", c.bufnum]);
Synth("sample", ["bufnum", k.bufnum]);
Synth("sample", ["bufnum", d.bufnum]);
```

Routine D.2

- 8 beat

```
(
   Routine({
      loop({
            Synth("sample", ["bufnum", c.bufnum]);
            0.25.wait
      });
   }).play;

   Routine({
            0.5.wait;
      loop({
            Synth("sample", ["bufnum", d.bufnum]);
            1.0.wait;
      });
   }).play;

   Routine({
      loop({
            Synth("sample", ["bufnum", k.bufnum]);
            1.0.wait;
      });
   }).play;
)
```

Routine D.3

- Wiederholung mit einer Zahl

```
4.do{
   arg index;
    index.postln;
};
```

erste Wiederholung	zweite Wiederholung	dritte Wiederholung	vierte Wiederholung
index = 0	index = 1	index = 2	index = 3

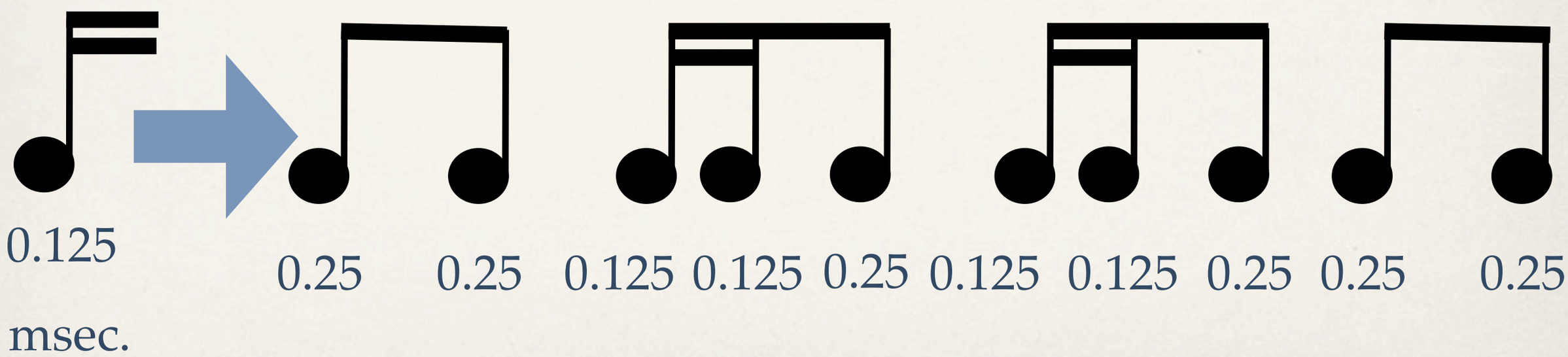
- Wiederholung mit einem Array

```
[60, 61, 64, 65].do{
   arg item;
    item.postln;
};
```

erste Wiederholung	zweite Wiederholung	dritte Wiederholung	vierte Wiederholung
item = 60	item = 61	item = 64	item = 65

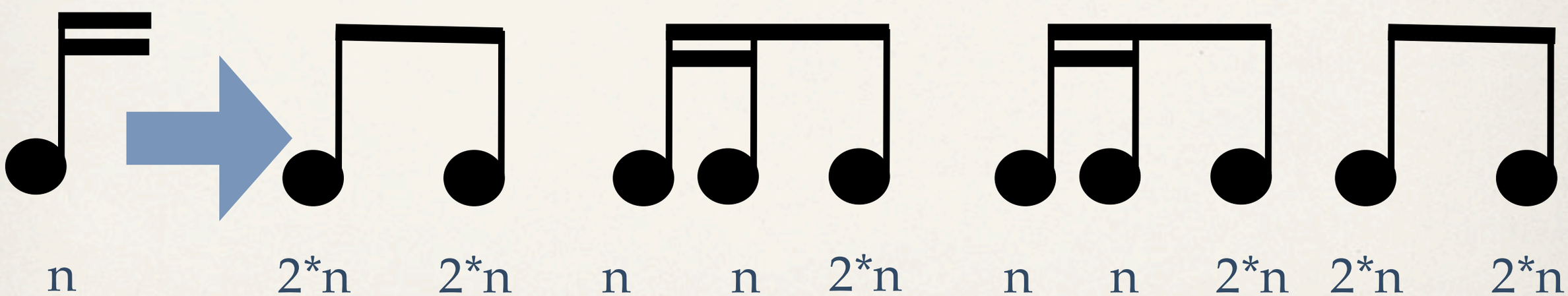
Routine D.4

```
Routine({
   loop{
      [0.25, 0.25, 0.125, 0.125, 0.25, 0.125, 0.125, 0.25, 0.25, 0.25].do{
         arg item;
         Synth("sample", ["bufnum", c]);
         item.wait
      }
   };
}).play;
```



Routine D.5

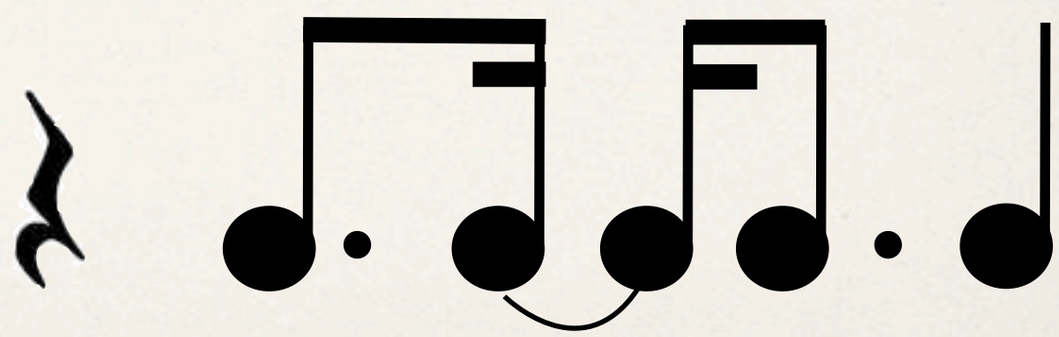
```
Routine({
   n = 0.125;
   loop{
      ([ 2, 2, 1, 1, 2, 1, 1, 2, 2, 2] *n).do{
         arg item;
         Synth("sample", ["bufnum", c]);
         item.wait
      }
   };
}).play;
```



= 0.125

Routine D.6

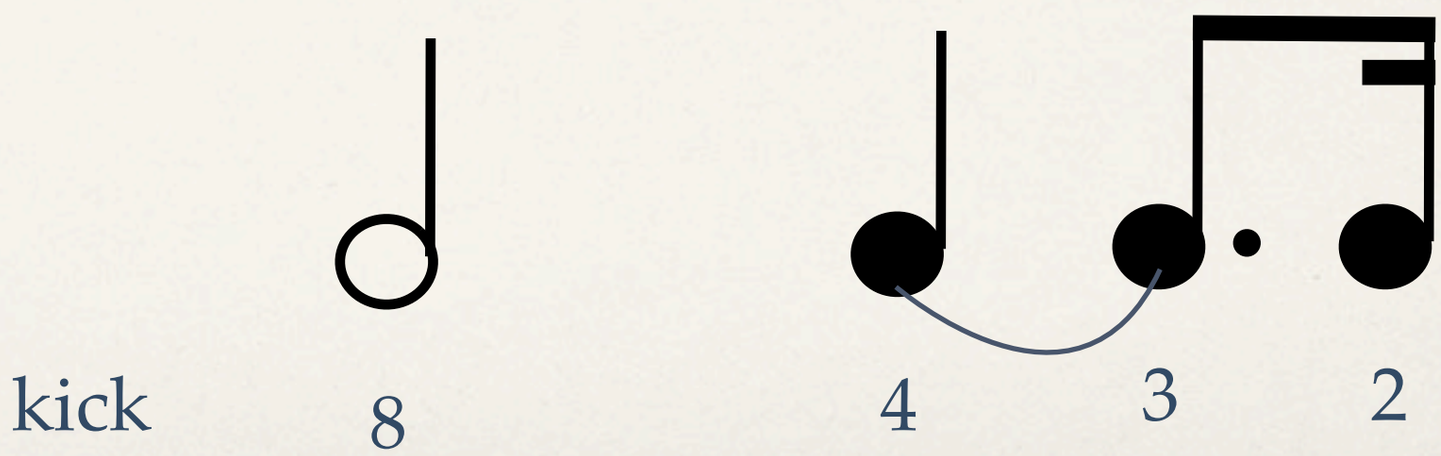
```
Routine({
   n = 0.125;
   loop{
      (4*n).wait;
      ([3, 2, 3, 4]*n).do{
         arg item;
         Synth("sample", ["bufnum", d.bufnum]);
         item.wait
      }
   };
}).play;
```



snare 4 3 2 3 4

Routine D.7

```
Routine({
   n = 0.125;
   loop{
      ([8, 7, 1]*n).do{
         arg item;
         Synth("sample", ["bufnum", k.bufnum]);
         item.wait
      }
   };
}).play;
```



Routine D.8

drum sequence

```
n = 0.155;
Routine({
    loop{
        ([2, 2, 1, 1, 2, 1, 1, 2, 2, 2]*n).do{
            arg item;
            Synth("sample", ["bufnum", c.bufnum]);
            item.wait
        }
    };
}).play;

Routine({
    loop{
        (4*n).wait;
        ([3, 2, 3, 4]*n).do{
            arg item;
            Synth("sample", ["bufnum", d.bufnum]);
            item.wait
        }
    };
}).play;

Routine({
    loop{
        ([8, 7, 1]*n).do{
            arg item;
            Synth("sample", ["bufnum", k.bufnum]);
            item.wait
        }
    };
}).play;
```

Routine D.9

- Eine kleine Technomusik

```
(
    n = 0.1;
    Routine({
        loop{
            ([2, 2, 1, 1, 2, 1, 1, 2, 2, 2]*n * 0.5 ).do{
                arg item;
                Synth("sample", ["bufnum", c.bufnum]);
                item.wait
            }
        };
    }).play;
    Routine({
        loop{
            (4*n).wait;
            ([3, 2, 3, 4]*n ).do{
                arg item;
                Synth("sample", ["bufnum", d.bufnum]);
                item.wait
            }
        };
    }).play;
    Routine({
        loop{
            ([8, 7, 1]*n*0.5).do{
                arg item;
                Synth("sample", ["bufnum", k.bufnum]);
                item.wait
            }
        };
    }).play;
```

```
    Routine({
        loop{
            t = [0, 5, 7].choose;
            4.do{
                Synth("mySynth",["freq", (60+t).midicps, "amp", 0.3]);
                n.wait;
                Synth("mySynth",["freq", (72+t).midicps, "amp", 0.3]);
                n.wait;
                Synth("mySynth",["freq", (67+t).midicps, "amp", 0.3]);
                n.wait;
                Synth("mySynth",["freq", (68+t).midicps, "amp", 0.3]);
                n.wait;
            }
        };
    }).play;
    Routine({
        loop{
            2.do{
                Synth("mySynth",["freq", (36+t).midicps, "amp", 0.3]);
                (n*3).wait;
                Synth("mySynth",["freq", (39+t).midicps, "amp", 0.3]);
                (n*3).wait;
                Synth("mySynth",["freq", (36+t).midicps, "amp", 0.3]);
                (n*2).wait;
            }
        };
    }).play;
)
```

Routine D.10

- Arrays in Arrays

```
a = [3, [3, 4, 5]]
```

→

a[0] ... 3
a[1] ... [3, 4, 5]
a[1][2] ... 5

```
a = [[2, [3, 5]], [3, 4, 5]]
```

→ a[0][1][1] ... 5

- SC3 wählt ein rhythmisches Muster aus.

```
a = [[2, 2, 1, 1, 2],[1, 1, 2, 1, 1, 2],[1, 1, 1, 1, 2, 2]];
Routine({
   loop{
      (a.choose*0.125).do{
         arg item;
         Synth("sample", ["bufnum", c.bufnum]);
         item.wait
      }
   };
}).play;
```

E. *Routine und Live-Elektronik*

Routine E.1

- Wie kann man Routine echtzeitig verändern?

```
(
   x = 0.1;
   Routine({
      loop({
         Synth("mySynth",["freq", [60, 72, 67, 68 ].choose.midicps, "amp", 0.3]);
         x.wait;
      });
   }).play;
)

x = 0.2;
```

Man kann die Geschwindigkeit später verändern

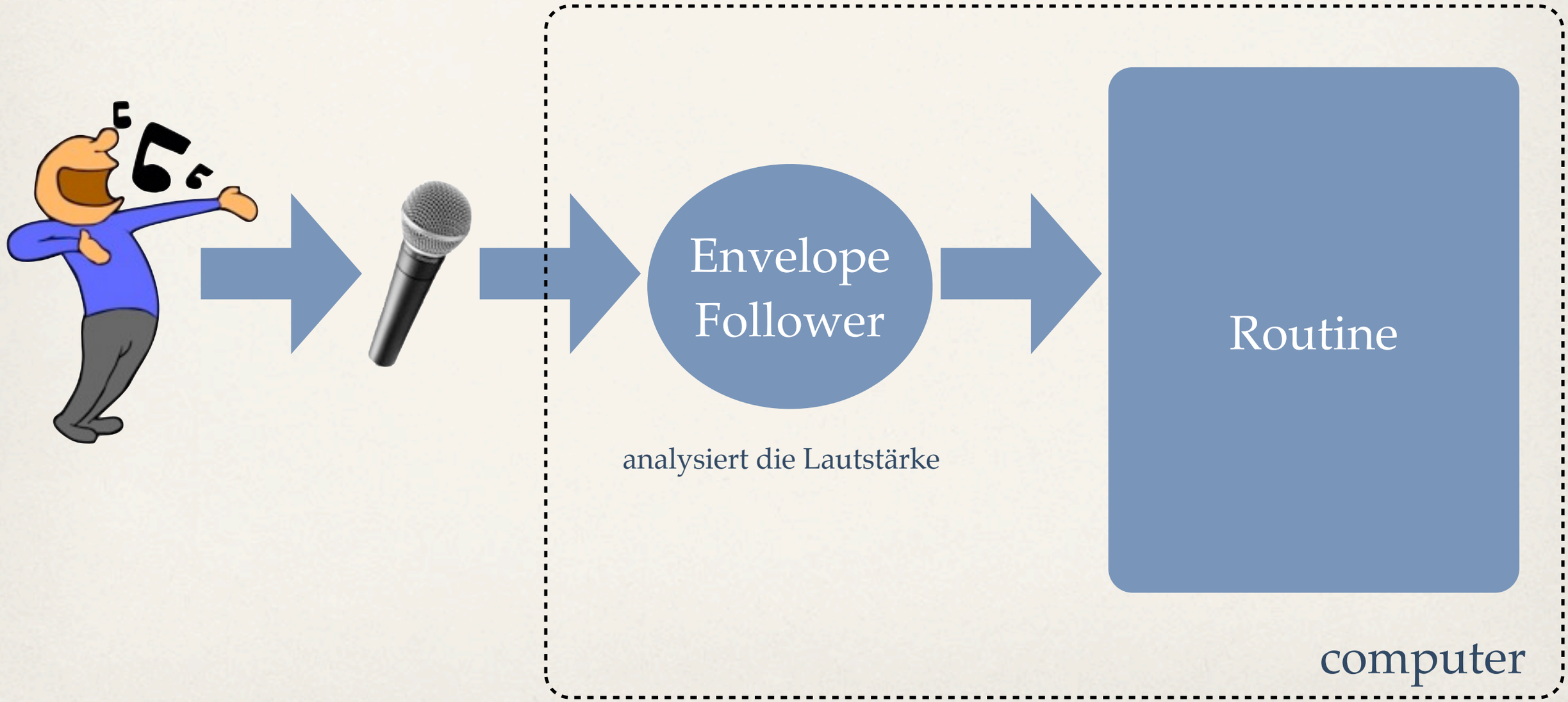
Routine E.2

- mit Array

```
(
   a = [60, 72, 67, 68];
   Routine({
      loop({
         Synth("mySynth",["freq", a.choose.midicps, "amp", 0.3]);
         0.1.wait;
      });
   }).play;
)



a = [61, 71, 69, 62];
a = [60, 72, 68, 69];
```

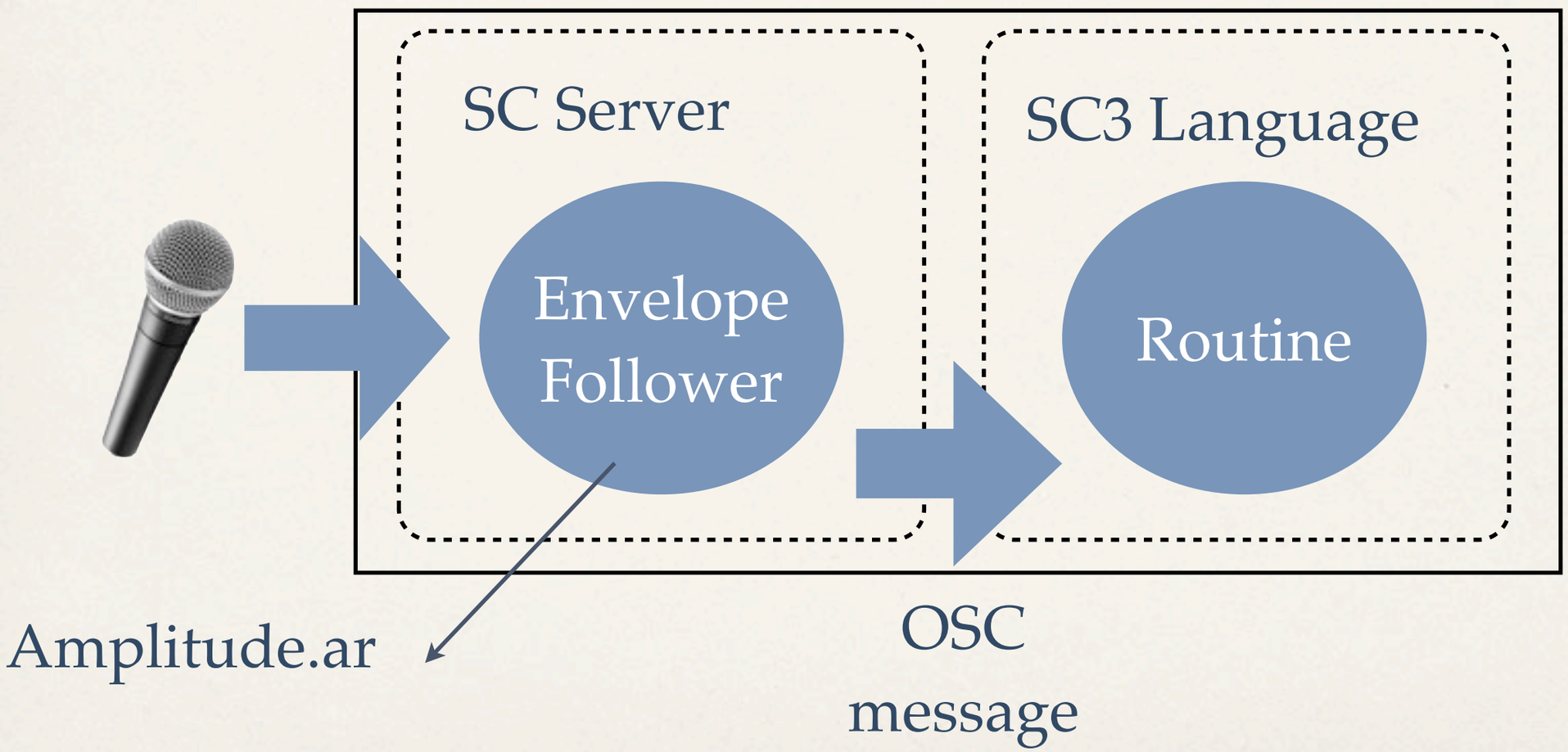
Routine E.3

- Eingegebene Klänge Steuern eine Routine



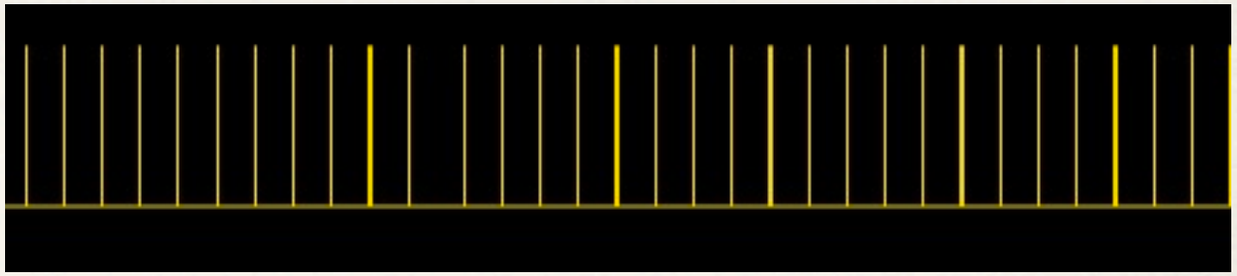
Routine E.4

- SC Server analysiert die Amplitude der eingegebenen Klänge.
- SC Language führt Routines aus



- SC Server muss an SC3 Language OSC messages senden

Routine E.5



- SendTrig

100 mal pro Sekunde

ID

Lautstärke

```
{
   i = SoundIn.ar(0);
   a = Amplitude.ar(i);
   SendTrig.kr(Impulse.kr(100), 0, a);
}.play;

OSCresponder(s.addr, '/tr', { arg time, responder, msg;
   msg.postln;
}).add;
```

Lautstärke

post Fenster

```
[ /tr, 1000, 0, 0.011061498895288 ]
[ /tr, 1000, 0, 0.0090784411877394 ]
[ /tr, 1000, 0, 0.012219598516822 ]
[ /tr, 1000, 0, 0.0077827032655478 ]
[ /tr, 1000, 0, 0.017358642071486 ]
[ /tr, 1000, 0, 0.010703262872994 ]
[ /tr, 1000, 0, 0.015148321166635 ]
[ /tr, 1000, 0, 0.0094004319980741 ]
[ /tr, 1000, 0, 0.0087308213114738 ]
[ /tr, 1000, 0, 0.0079877786338329 ]
```

- OSCresponder

Empfänger der OSC messages.

generiert

Routine E.6

```
{
   i = SoundIn.ar(0);
   a = Amplitude.ar(i);
   SendTrig.kr(Impulse.kr(100), 0, a);
}.play;

OSCresponder(s.addr, '/tr', { arg time, responder, msg;
   msg[3].postln;
}).add;
```

0	1	2	3
[/tr,	1000,	0,	0.011061498895288]

nur Lautsträke

Routine E.7

- Amplitude steuert das Tempo

```
{
   i = SoundIn.ar(0);
   a = Amplitude.ar(i);
   SendTrig.kr(Impulse.kr(100), 0, a);
}.play;

OSCresponder(s.addr, '/tr', { arg time, responder, msg;
   r = (0.5 - msg[3]).clip(0.05, 1.0);
}).add;

Routine({
   loop{
      Synth("mySynth", ["freq", 440, "amp", 0.3]);
      r.wait;
   }
}).play;
```

Routine E.8

- Amplitude steuert den Bereich der Randomisierung

```
{
   i = SoundIn.ar(0);
   a = Amplitude.ar(i);
   SendTrig.kr(Impulse.kr(100), 0, a);
}.play;

OSCresponder(s.addr, '/tr', { arg time, responder, msg;
   b = msg[3] * 400;
}).add;

Routine({
   loop{
      Synth("mySynth", ["freq", 440 + b.rand, "amp", 0.3]);
      0.05.wait;
   }
}).play;
```

Routine E.9

- if

```
{
   i = SoundIn.ar(0);
   a = Amplitude.ar(i);
   SendTrig.kr(Impulse.kr(100), 0, a);
}.play;

OSCresponder(s.addr, '/tr', { arg time, responder, msg;
   var amp;
   amp = msg[3];
   if( amp > 0.3, {
      "laut!".postln;
   });
}).add;
```

Wenn der eingegebene Klang laut ist,
SC3 zeigt „laut!“ auf dem Postfenster an.

Routine E.10

- Logical Operators

<, >	kleiner,großer als
<=, >=	kleiner,großer oder gleich
==, !=	gleich, ungleich

3 > 4	→	false
3 < 4	→	true
3 == 4	→	false
3 != 4	→	true

Routine E.11

- if (*condition* ,{*truefunc*} ,{*falsefunc*});

  if evaluieret die *condition*

  Wenn das Ergebnis **richtig** (true) ist, ***truefunc*** wird ausgeführt werden.

  Wenn das Ergebnis **falsch** (false) ist, ***falsefunc*** wird ausgeführt werden.

```
if(4 > 3, {"richtig".postln},{"falsch".postln});

if(3 > 4, {"richtig".postln},{"falsch".postln});
```

Routine E.12

```
{
   i = SoundIn.ar(0);
   a = Amplitude.ar(i);
   SendTrig.kr(Impulse.kr(100), 0, a);
}.play;

p = "leise";
OSCresponder(s.addr, '/tr', { arg time, responder, msg;
   var amp;
   amp = msg[3];
   if((amp > 0.3) && (p == "leise"), {
      "laut!".postln;
      p = "laut"});

   if((amp <= 0.3) && (p == "laut"),{
      p = "leise"
   });

}).add;
```

Wenn die Lautstärke des eingegebenen Klang laut ist, SC3 zeigt „laut!“ auf dem Postfenster **nur einmal** an.

Routine E.13

- && (Logical AND)

$(4 > 3)$ && $(5 > 2)$ → true

true true

$(4 < 3)$ && $(5 > 2)$ → false

false true

$(4 < 3)$ && $(5 < 2)$ → false

false false

Logical AND generiert *true,* wenn beide *conditions true* sind

Routine E.14

```
{
    i = SoundIn.ar(0);
    a = Amplitude.ar(i);
    SendTrig.kr(Impulse.kr(100), 0, a);
}.play;

p = "leise";
r = [60, 72, 67, 68];

OSCresponder(s.addr, '/tr', { arg time, responder, msg;
    var amp;
    amp =   msg[3];
    if((amp > 0.3) && (p == "leise"), {
        r = [ [60, 72, 67, 68], [61, 71, 69, 62], [60, 72, 68, 69],[60, 61, 62, 63]].choose;
        "change".postln;
        p = "laut"});
    if((amp <= 0.3) && (p == "laut"),{
        p = "leise"
    });
}).add;

Routine({
    loop{
        r.do{
            arg item;
            Synth("mySynth", ["freq", item.midicps, "amp", 0.3]);
            0.1.wait;
        }
    }
}).play;
```

F. *Demostück*

- *DSP of Fury* von Andrew Babcock

Mapping

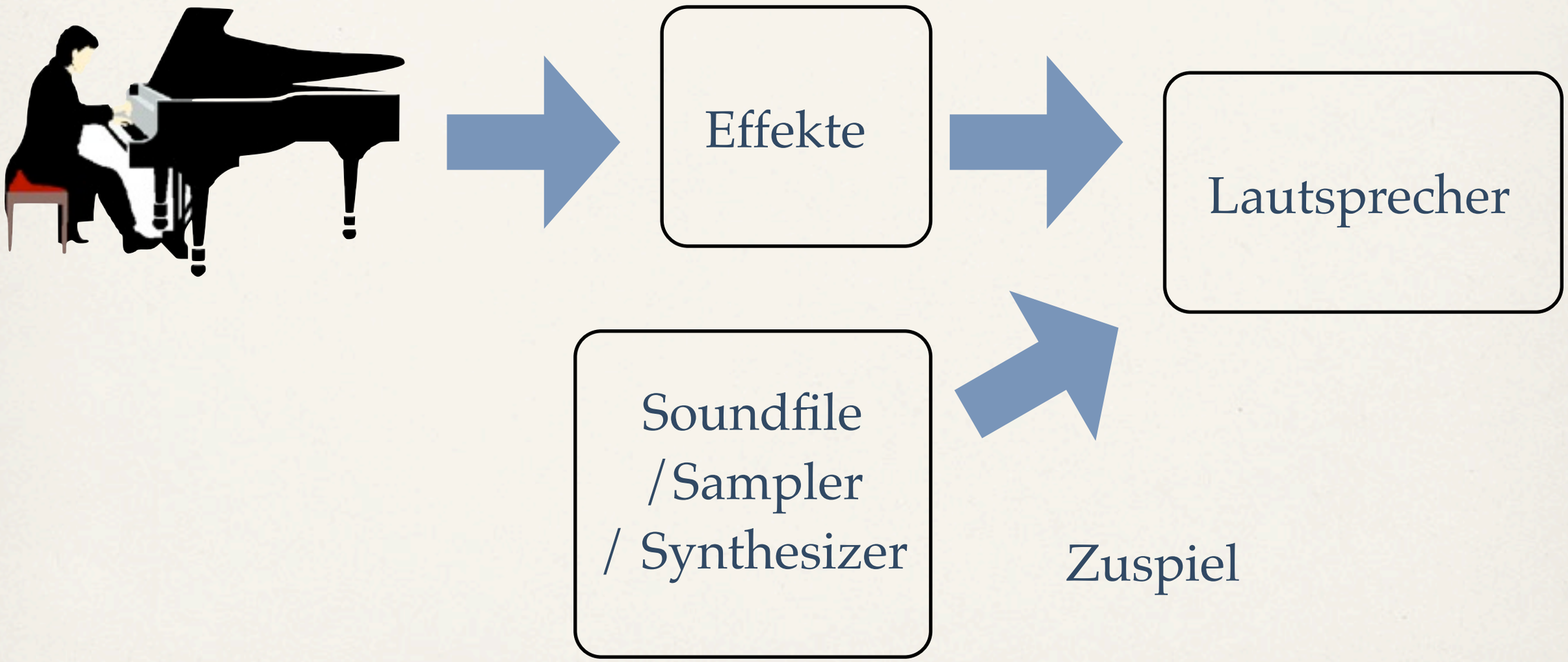
Tipps für Live-Elektronik Kompostion

Live-Elektronik Musik

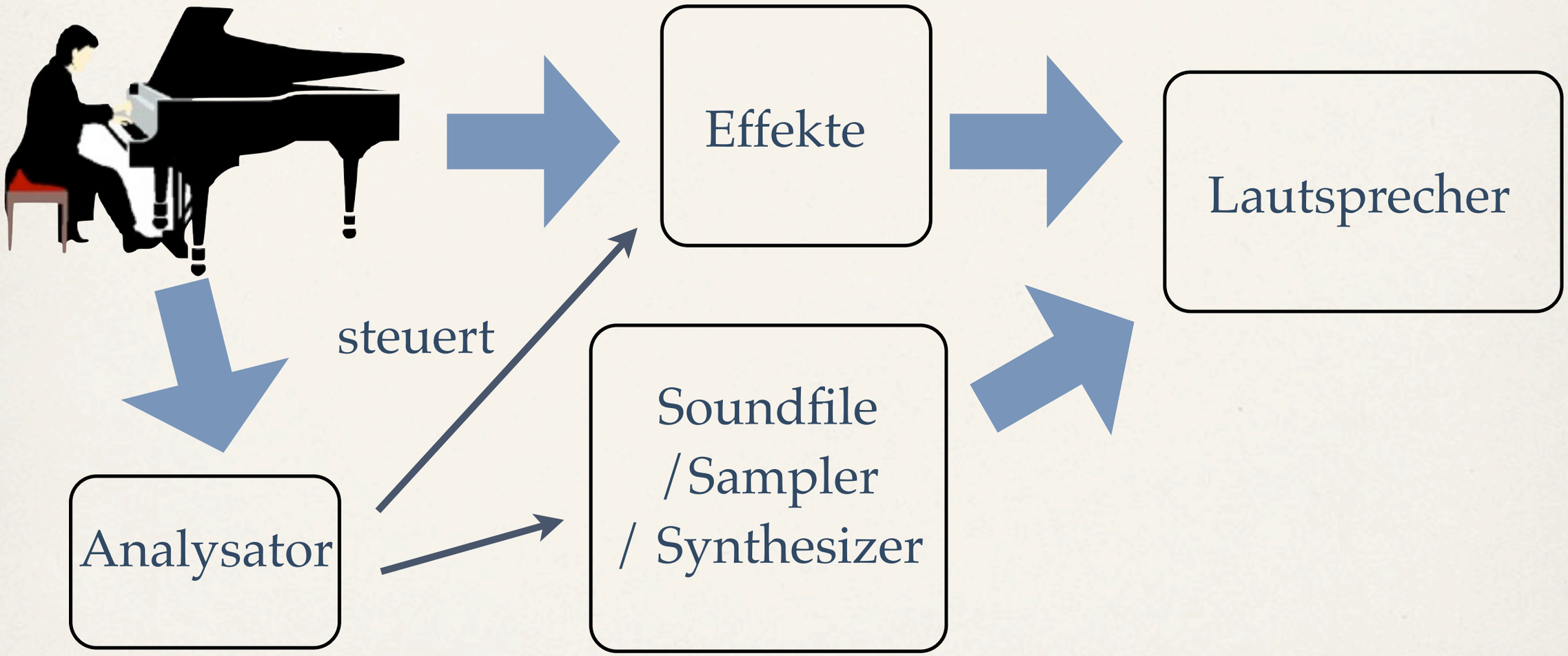
z.B

			Stück		
Cuelist	Delay	Delay + Reverb	Flanger Granular	FFT	Delay

Live-Elektronik Musik

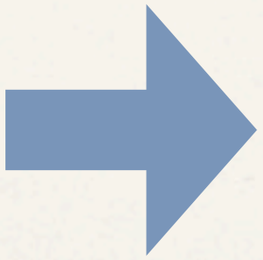


Live-Elektronik Musik



Analysator

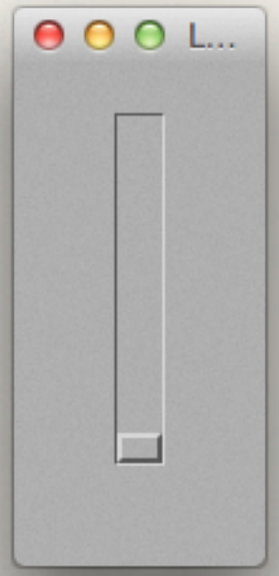
- Wie kann man Lautstärke analysieren?
  - Amplitude.ar
- Wie kann man Tonhöhe analysieren?
  - Pitch.ar



Es gibt viele Analysator Ugens

Quellcode 1

- Level Meter




```
(
   w = Window.new("Loudness", Rect(200, 100, 100, 200));
   v = Slider(w, Rect(40, 20, 20, 140));
   w.front;

   OSCresponder( s.addr, '/tr', {
      arg time, responder, msg;
      {
         v.value_(msg[3]);
      }.defer;
   }).add;
)

{
   i = SoundIn.ar(0);
   a = Amplitude.kr(i);
   SendTrig.kr(Impulse.kr(100), 0, a);
}.play;
```

Quellcode 2

- Pitch Tracker

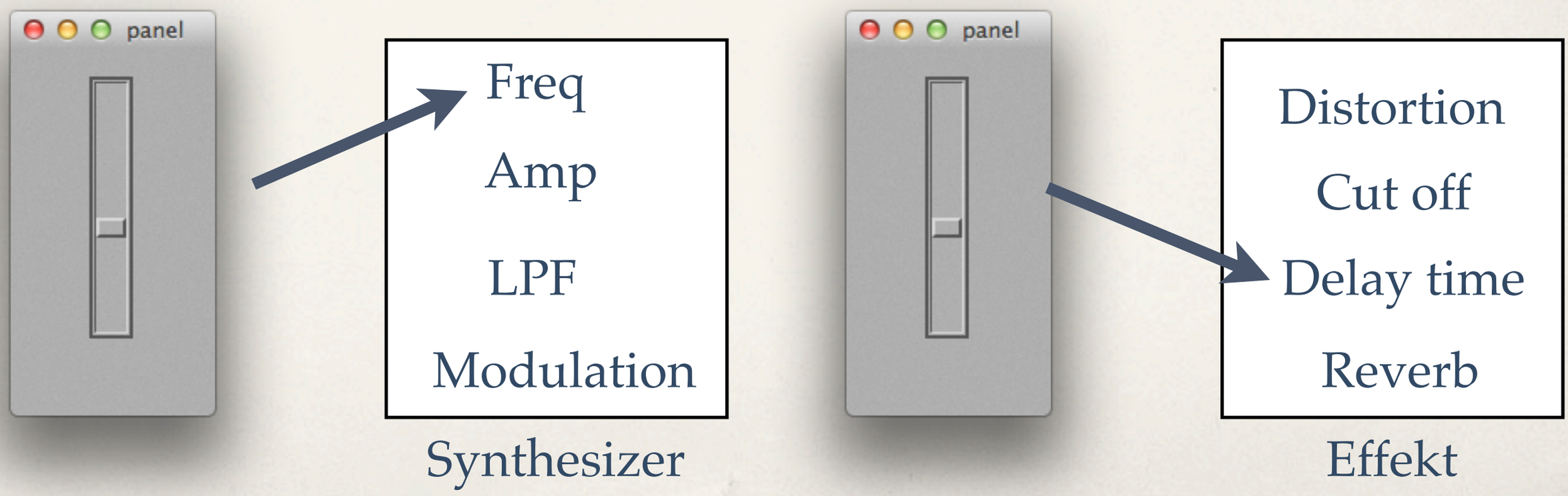
```
(
   w = Window.new("Pitch", Rect(200, 100, 100, 200));
   v = Slider(w, Rect(40, 20, 20, 140));
   w.front;

   OSCresponder( s.addr, '/tr', {
      arg time, responder, msg;
      {
         v.value_(msg[3] / 2000) ;
      }.defer;
   }).add;
)

{
   i = SoundIn.ar(0);
   #a, b = Pitch.kr(i);
   SendTrig.kr(Impulse.kr(100), 0, a * b);
}.play;
```

Schieberegler

Jetzt können wir einen Schieberegler mit den eingegebenen Klänge steuern.

Was können wir mit einem Schieberegler machen?

z.B. einen Parameter eines Synthesizers, Samplers oder Effekts steuern



Wir benötigen viele Schieberegler

Freq

Amp

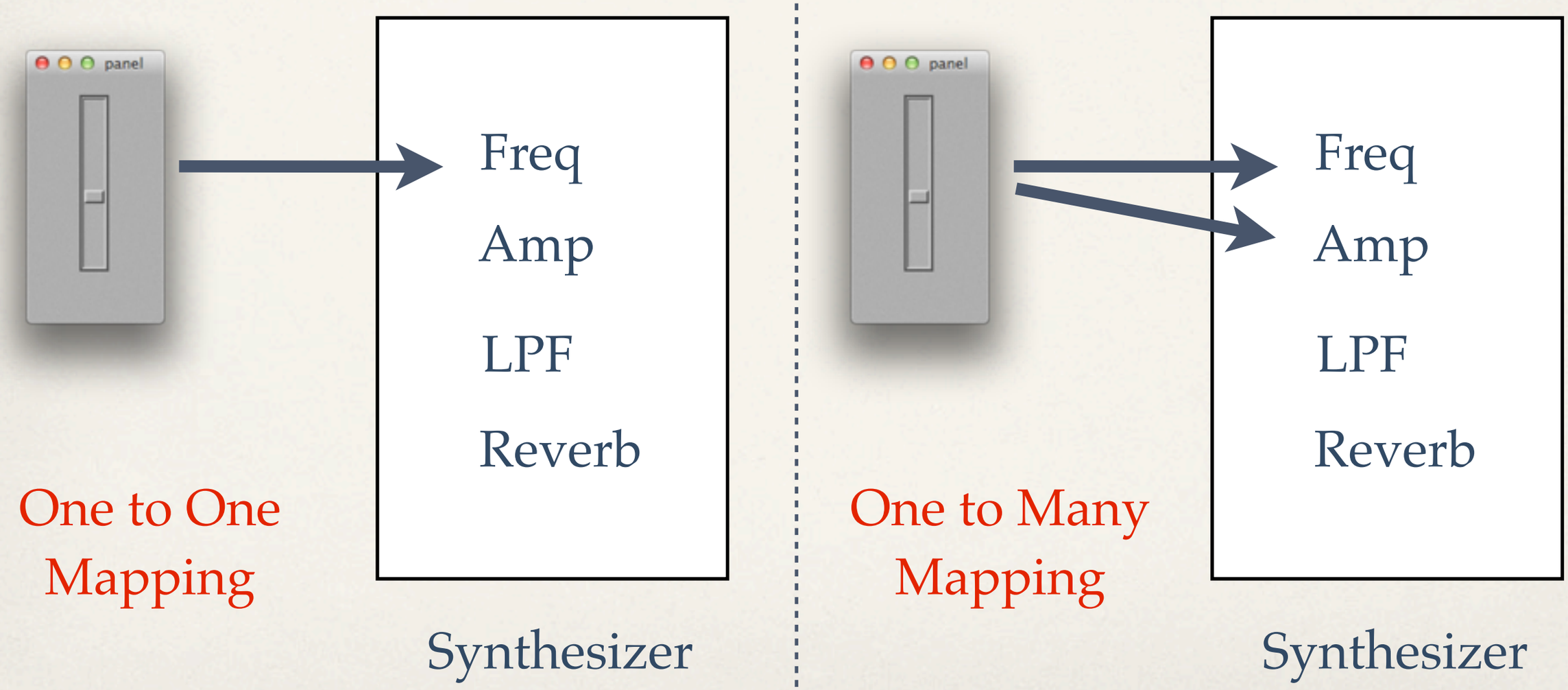
LPF

Reverb

ein typischer Software-Synthesizer
mit viel Schiebereglern und Knöpfen.

One to One / One to Many

mit einem Schiebereglern kann man zwei oder drei Parameter steuern.



Quellcode 3

Die Amplitude kontrolliert die Frequenz des Synthesizers

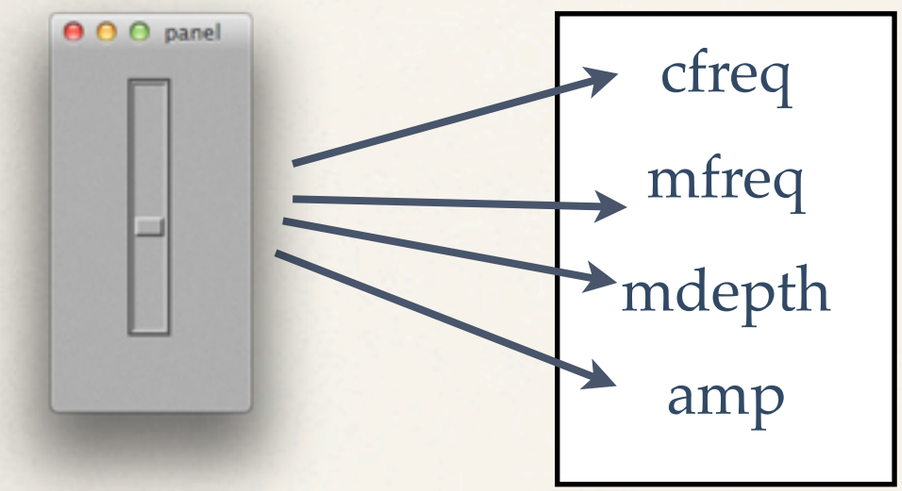
One to One mapping

```
SynthDef("Analyzer",{
   i = SoundIn.ar(0);
   a = Amplitude.ar(i);
   SendTrig.kr(Impulse.kr(100), 0, a);
}).load;
```

```
SynthDef("SawSynth",{
   arg freq, amp;
   r = Saw.ar(freq, 0.3);
   Out.ar(0, r);
}).load;
```

```
(
   w = Window.new("map", Rect(200, 100, 100, 200));
   v = Slider(w, Rect(40, 20, 20, 140));
   w.front;

   Synth("Analyzer");
   k = Synth("SawSynth");
   OSCresponder( s.addr, '/tr', {
      arg time, responder, msg;
      {
         p = msg[3];
         v.value_(p);
         k.set("freq", p * 1500);
      }.defer;
   }).add;
)
```

Quellcode 4



One to Many mapping

Die Amplitude kontrolliert 4 Parameter des FM-Synthesizers

```
SynthDef("FMSynth",{
   arg cfreq, mfreq, mdepth, amp;
   c = SinOsc.ar(mfreq, 0, mdepth) ;
   r = SinOsc.ar(cfreq + c);
   Out.ar(0, r * amp);
}).load;
```

```
(
   w = Window.new("map", Rect(200, 100, 100, 200));
   v = Slider(w, Rect(40, 20, 20, 140));
   w.front;

   Synth("Analyzer");
   k = Synth("FMSynth");

   OSCresponder( s.addr, '/tr', {
      arg time, responder, msg;
      {
         p = msg[3];
         v.value_(p);
         k.set("cfreq", p * 500);
         k.set("mfreq", p * 400);
         k.set("mdepth", p * 600);
         k.set("amp", p);
      }.defer;
   }).add;
)
```

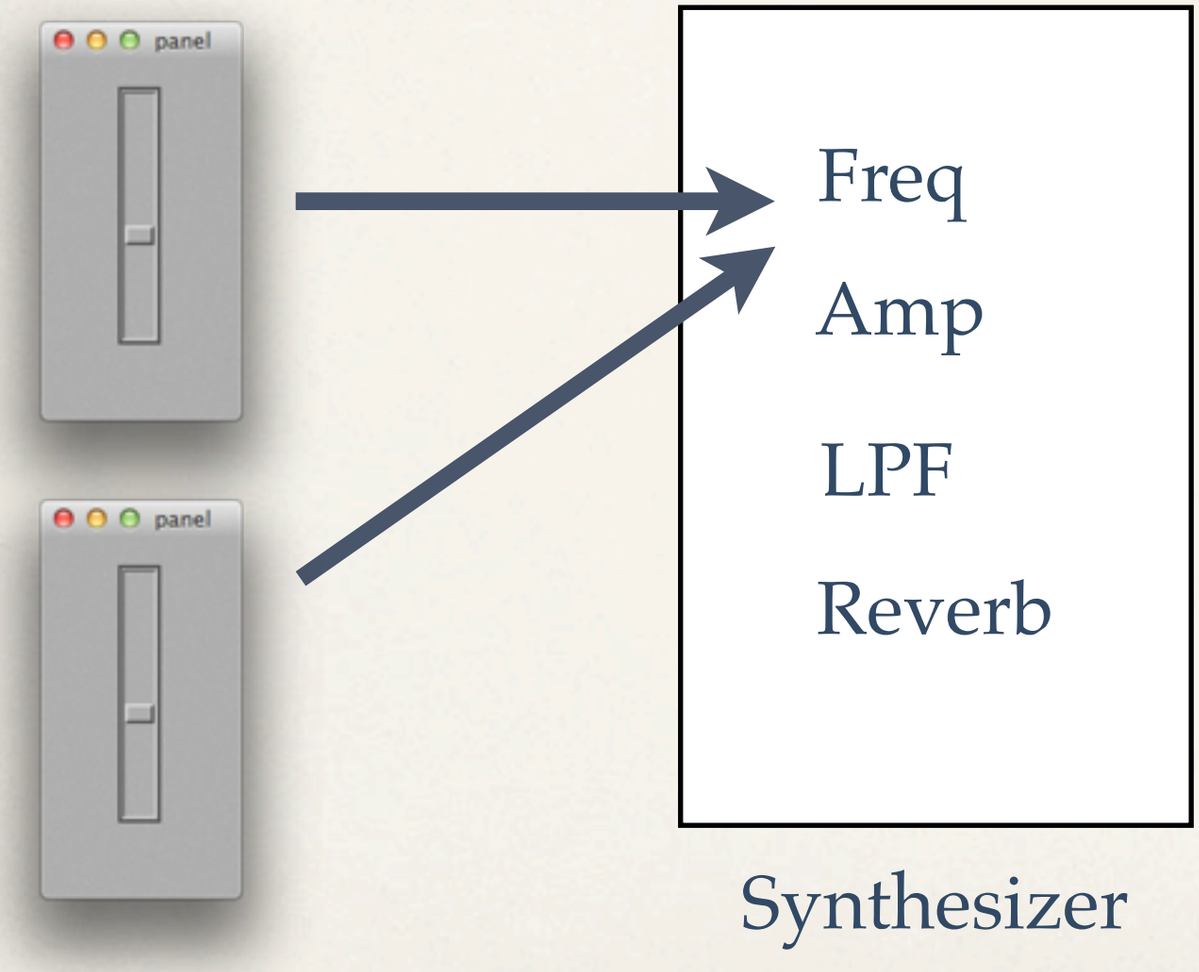
Quellcode 5

- One to Many Mapping mit Granular Sampler

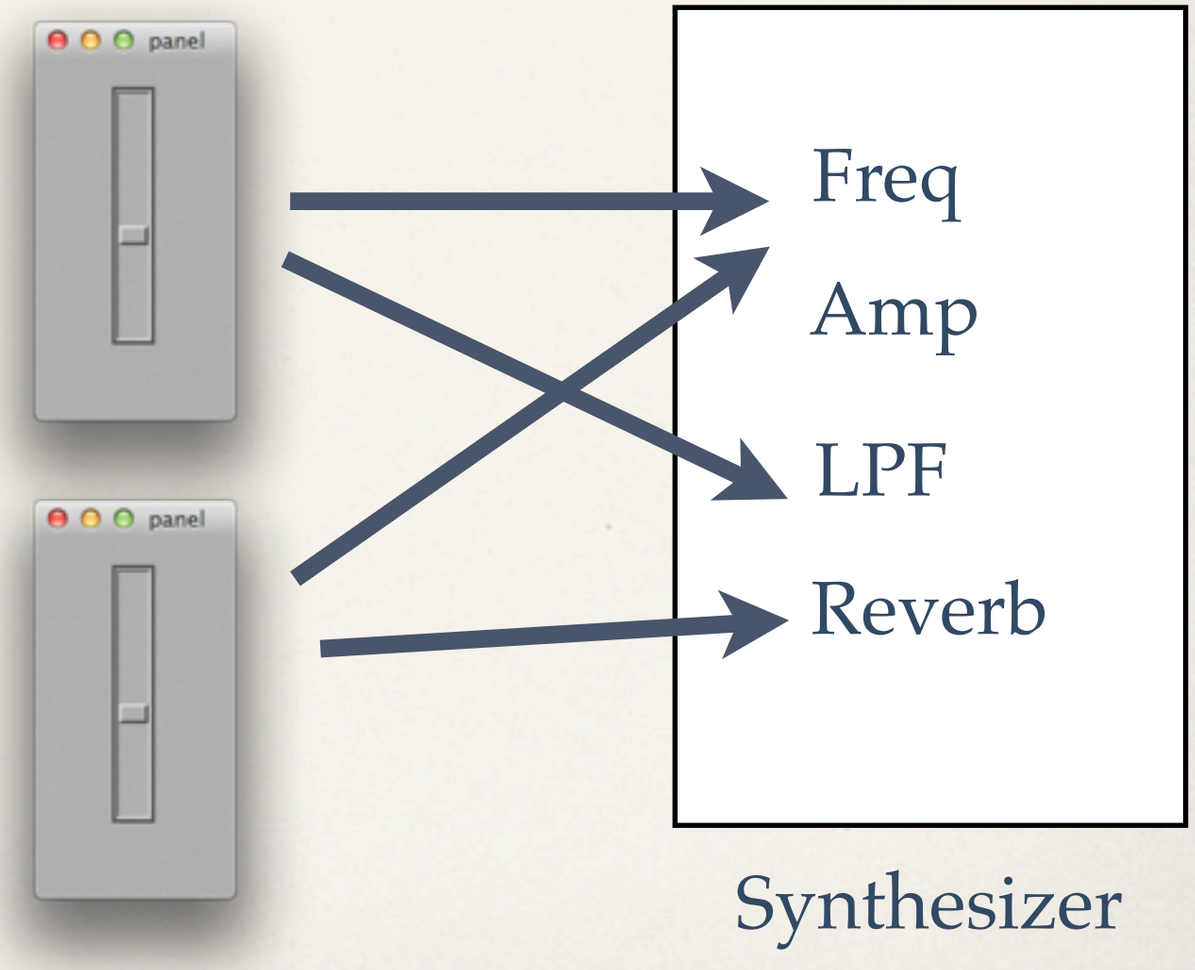
```
b = Buffer.read(s, "sounds/spoken.wav");

SynthDef("Granular",{
    arg buf, freq, rate, pos;
    g = TGrains.ar(2, Impulse.kr(freq), buf, rate, pos);
    Out.ar(0, g);
}).load;


(
    w = Window.new("map", Rect(200, 100, 100, 200));
    v = Slider(w, Rect(40, 20, 20, 140));
    w.front;

    Synth("Analyzer");
    k = Synth("Granular",["buf", b]);

    OSCresponder( s.addr, '/tr', {
        arg time, responder, msg;
        {
            p = msg[3];
            v.value_(p);
            k.set("rate", p*3  + 0.5);
            k.set("pos", p * 4.0);
            k.set("freq", p * 150);
        }.defer;
    }).add;
)
```

Die Amplitude steuert *rate*, *pos* und *freq*.

Many to one / Many to many

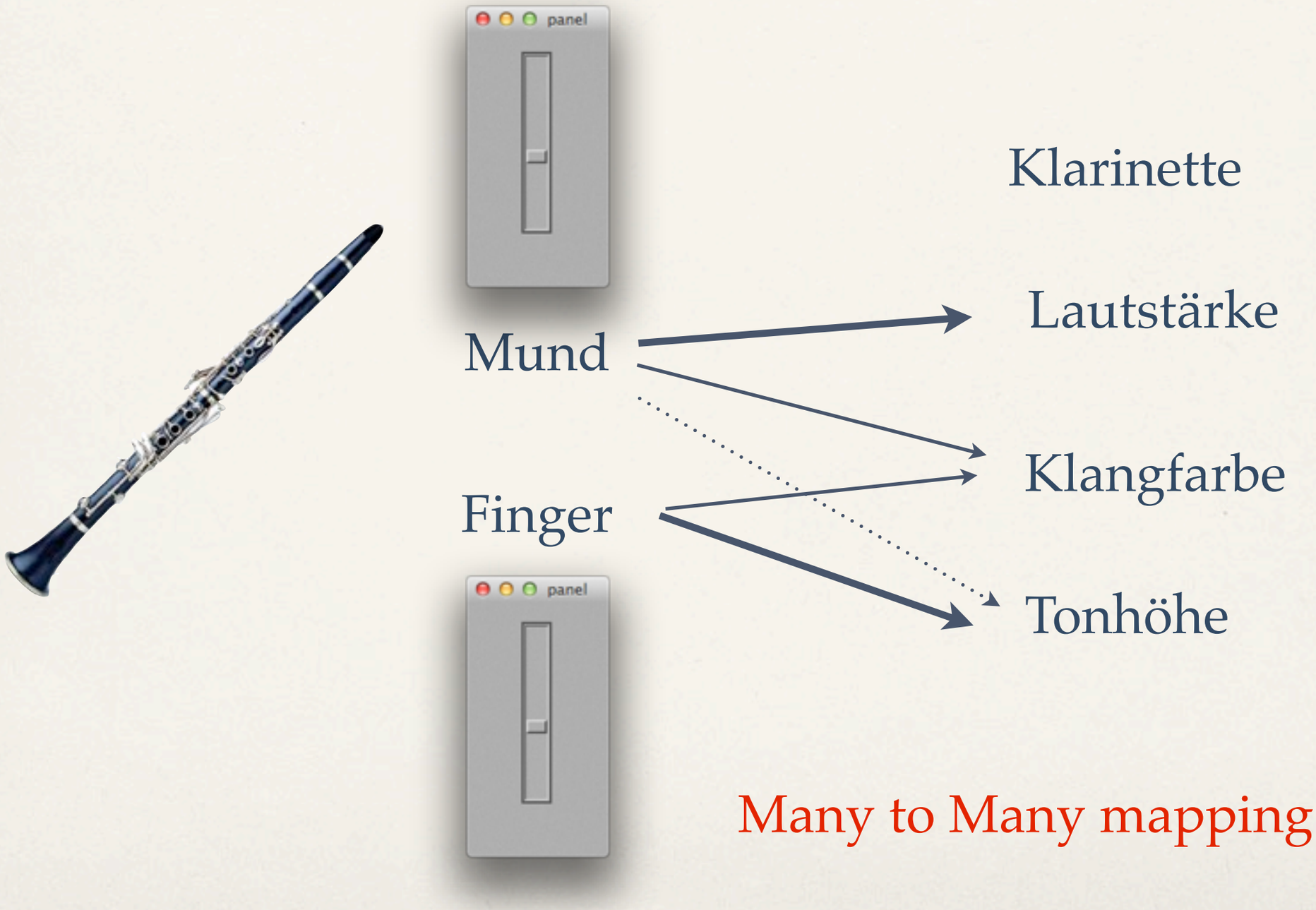


Many to One



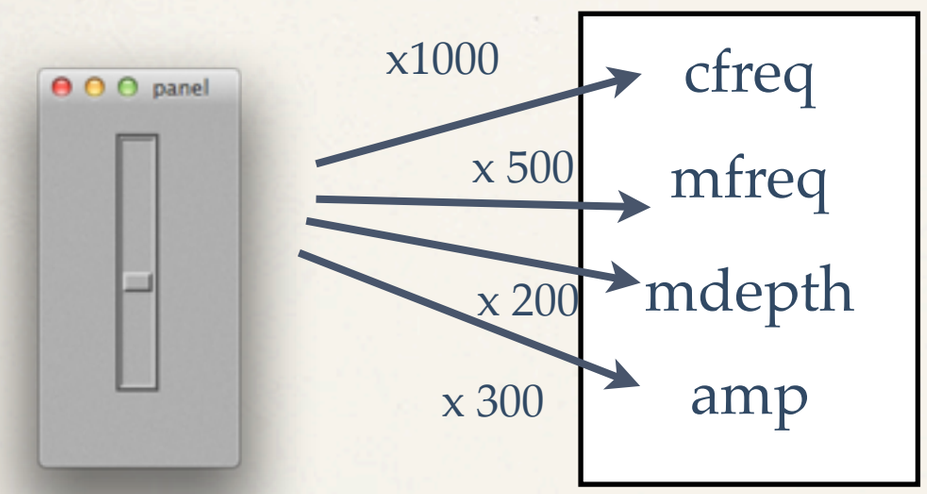
Many to Many

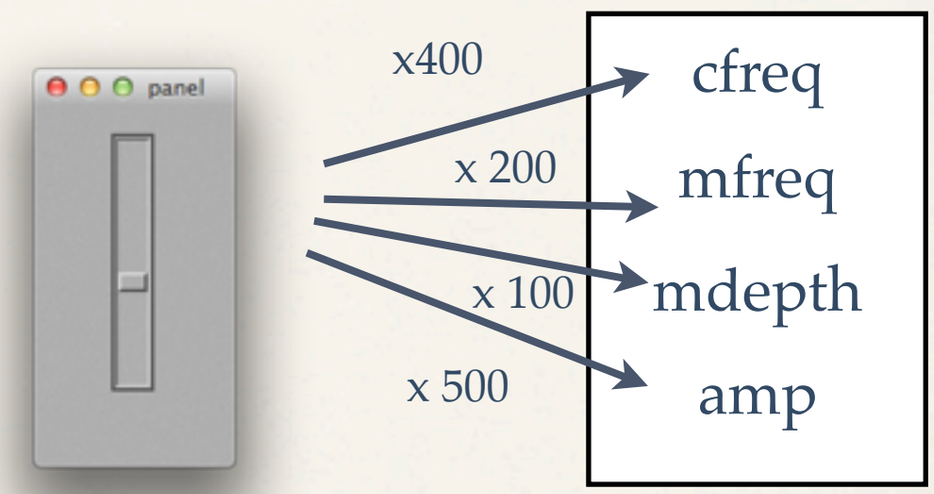
Das Mapping der Akustische Instrumenten



Many to Many mapping

Mapping ist flexibel veränderbar





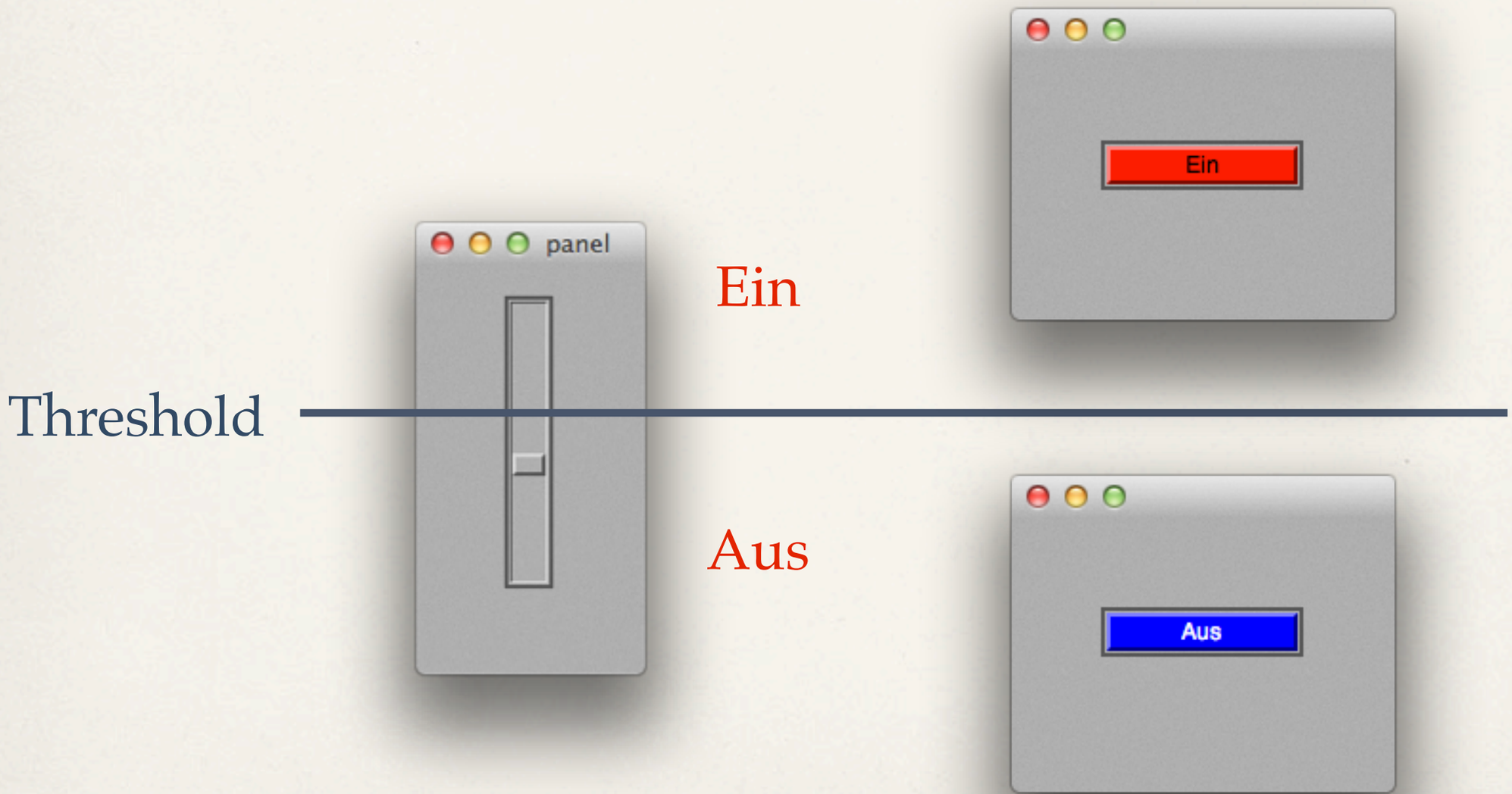
Quellcode 6

- Einstellbares Mapping

```
SynthDef("FMSynth",{
   arg cfreq, mfreq, mdepth, amp;
   c = SinOsc.ar(mfreq, 0, mdepth) ;
   r = SinOsc.ar(cfreq + c);
   Out.ar(0, r * amp);
}).load;
```

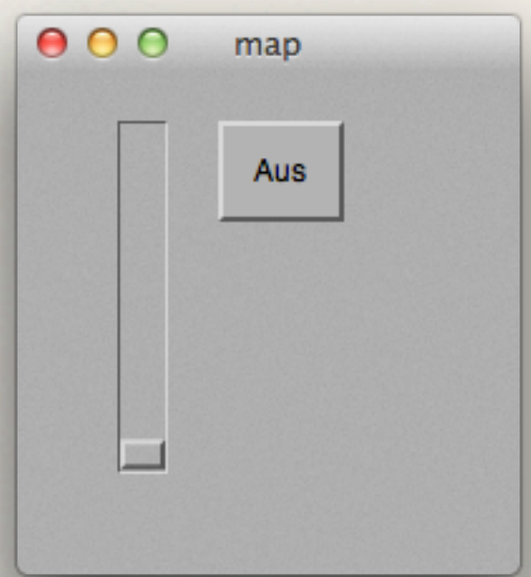
```
(
   w = Window.new("map", Rect(200, 100, 200, 200));
   v = Slider(w, Rect(40, 20, 20, 140));

   Synth("Analyzer");
   k = Synth("FMSynth");

   StaticText(w, Rect(80, 20, 50, 20)).string_("cfreq");
   c = NumberBox(w, Rect(130, 20, 50, 20)).value_(300);
   StaticText(w, Rect(80, 50, 50, 20)).string_("mfreq");
   m = NumberBox(w, Rect(130, 50, 50, 20)).value_(200);
   StaticText(w, Rect(80, 80, 50, 20)).string_("mdepth");
   d = NumberBox(w, Rect(130, 80, 50, 20)).value_(1000);
   StaticText(w, Rect(80, 110, 50, 20)).string_("amp");
   a = NumberBox(w, Rect(130, 110, 50, 20)).value_(0.5);

   w.front;

   OSCresponder( s.addr, '/tr', {
      arg time, responder, msg;
      {
         p = msg[3];
         v.value_(p);
         k.set("cfreq", p * c.value);
         k.set("mfreq", p * m.value);
         k.set("mdepth", p * d.value);
         k.set("amp", p * a.value);
      }.defer;
   }).add;

)
```

Schieberegler als ein Schalter



Quellcode 7

- Schieberegler als Schalter




```
(
    w = Window.new("map", Rect(200, 100, 200, 200));
    v = Slider(w, Rect(40, 20, 20, 140));
    b = Button(w, Rect(80, 20, 50, 40))
            .states_([["Aus"],["Ein", Color.black, Color.red]]);
    w.front;

    Synth("Analyzer");
    OSCresponder( s.addr, '/tr', {
        arg time, responder, msg;
        {
            p = msg[3];
            v.value_(p);
            if( p > 0.3, {
                b.value_(1);
            },{
                b.value_(0);
            });
        }.defer;
    }).add;
)
```

Musikalische Anwendungen

Mapping

- 2 Aktionen
  - Aus - Ein
  - Ein - Aus

- 4 Reaktionen
  - starten
  - stoppen
  - verändern
  - ignorieren

Musikalische Anwendungen

- *Music* for flute von Cort Lippe

Wenn der Flötist Laut spielt, erzeugt der Computer elektronische Klänge

- 2 Aktionen
  - Aus - Ein
  - Ein - Aus

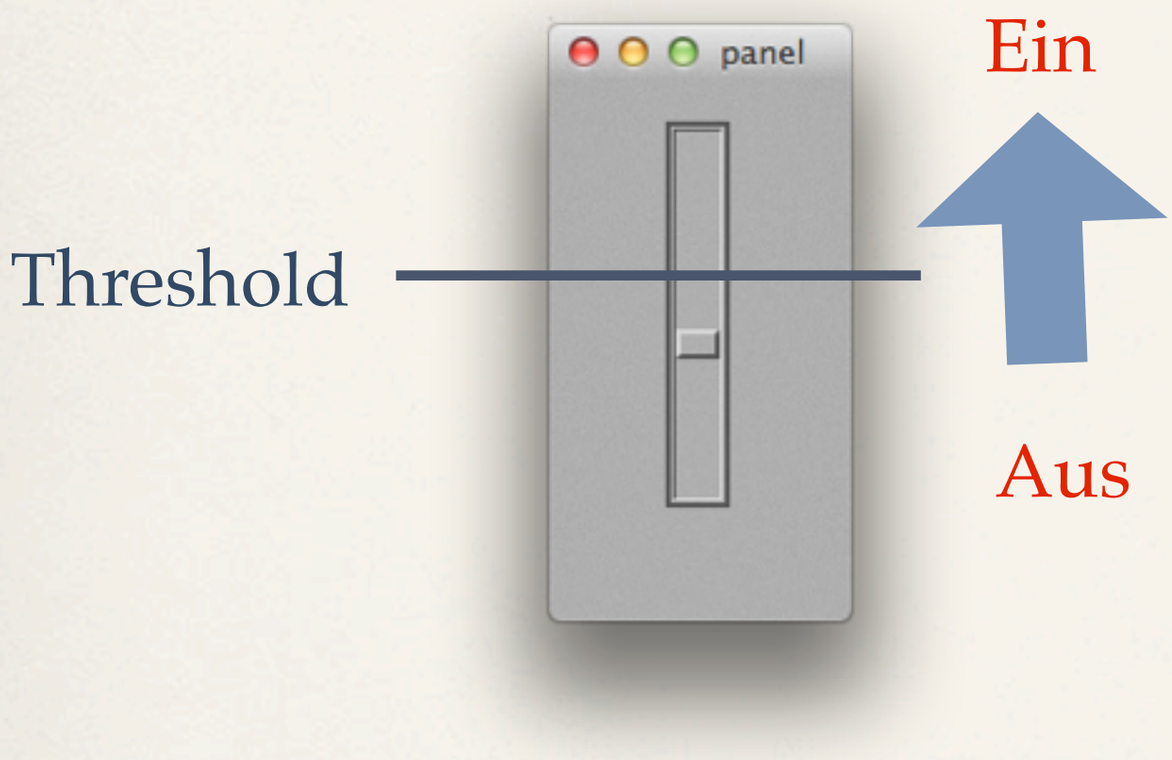
Mapping

- 4 Reaktionen
  - starten
  - stoppen
  - verändern
  - ignorieren

Quellcode 8

```
SynthDef("Granular2",{
    arg buf,freq, pos;
    i = Impulse.kr(freq);
    g = TGrains.ar(2,i , buf, 1.0, pos + TRand.ar(-0.1, 0.1, i));
    Out.ar(0, g);
}).load;

(
    var window, level, switch, onAction, offAction,previous, amplitude, buf, synth;
    buf = Buffer.read(s, "sounds/spoken.wav");
    window = Window.new("talking machine", Rect(200, 100, 200, 200));
    level = Slider(window, Rect(40, 20, 20, 140));
    switch = Button(window, Rect(80, 20, 50, 40))
             .states_([["Aus"],["Ein", Color.black, Color.red]]);
    onAction = PopUpMenu(window, Rect(80, 80 , 100, 25)).items_(["ignore","start","stop","change"]);
    offAction = PopUpMenu(window, Rect(80, 120, 100, 25)).items_(["ignore","start","stop","change"]);
    synth = nil;
    window.front;
    previous = 0;
    Synth("Analyzer");
    OSCresponder( s.addr, '/tr', {
        arg time, responder, msg;
        {
            amplitude = msg[3];
            level.value_(amplitude);
            if( ( amplitude > 0.3) && ( previous <= 0.3 ), {
                switch.value_(1);
                if(onAction.item == "start" && synth == nil,{ synth = Synth("Granular2",["buf", buf, "pos", 1.5, "amp", 1.0, "freq", 50]); });
                if(onAction.item == "stop",{ synth.free; synth = nil });
                if(onAction.item == "change",{ synth.set("pos", 5.0.rand); });
            });

            if( ( amplitude <= 0.3) && ( previous > 0.3 ), {
                switch.value_(0);
                if(offAction.item == "start" && synth == nil,{synth = Synth("Granular2",["buf", buf, "pos", 1.5, "amp", 1.0, "freq", 50]); });
                if(offAction.item == "stop",{ synth.free; synth = nil });
                if(offAction.item == "change",{ synth.set("pos", 5.0.rand); });
            });
            previous = amplitude;
        }.defer;
    }).add;
)
```

Schieberegler als Toggle



Erstes Mal ... Ein

Zweites Mal ... Aus

Drittes Mal ... Ein

Viertes Mal ... Aus

usw...

Quellcode 9

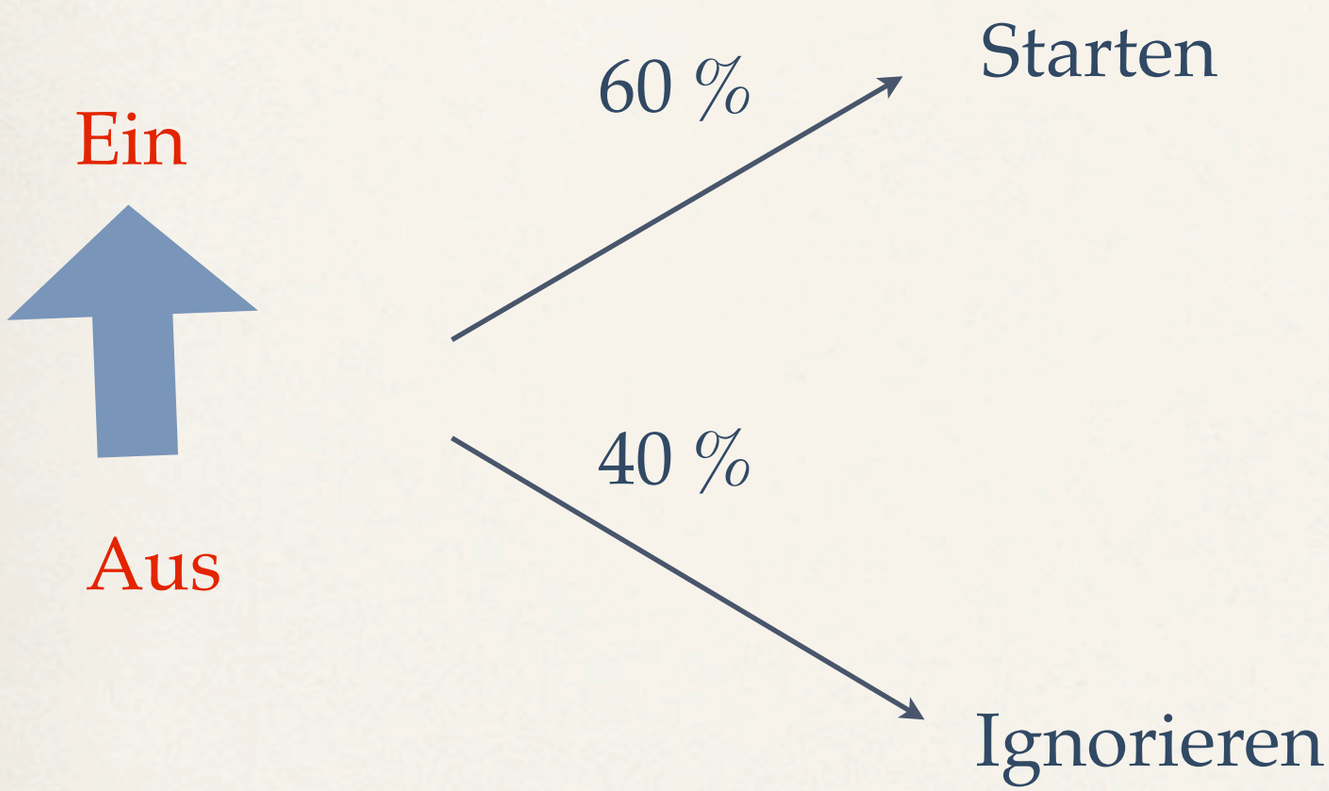
```
(
    var window, switch, amplitude, level, previous;
    window = Window.new("map", Rect(200, 100, 250, 200));
    level = Slider(window, Rect(40, 20, 20, 140));
    switch = Button(window, Rect(80, 20, 50, 40))
           .states_([["Aus"],["Ein", Color.black, Color.red]]);
    window.front;
    previous = 0.0;
    Synth("Analyzer");
    OSCresponder( s.addr, '/tr', {
        arg time, responder, msg;
        {
            amplitude = msg[3];
            level.value_(amplitude);
            if((amplitude >= 0.3) && (previous <= 0.3), {
                if(switch.value == 0,{
                    switch.value_(1);
                },{
                    switch.value_(0);
                });
            });
            previous = amplitude;
        }.defer;
    }).add;
)
```

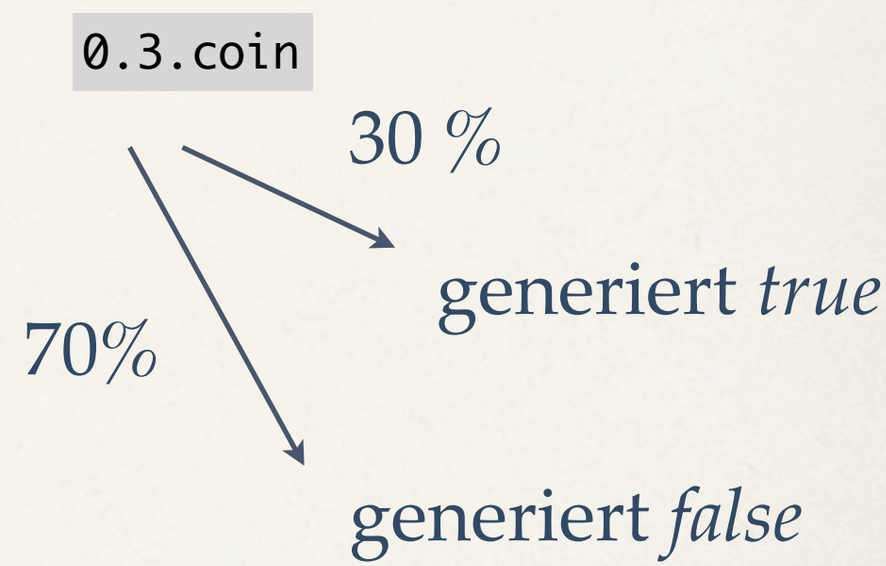
Quellcode 10

```
(
    var window, switch, amplitude, level, previous, buf, synth;
    buf = Buffer.read(s, "sounds/spoken.wav");
    window = Window.new("map", Rect(200, 100, 250, 200));
    level = Slider(window, Rect(40, 20, 20, 140));
    switch = Button(window, Rect(80, 20, 50, 40))
           .states_([["Aus"],["Ein", Color.black, Color.red]]);
    window.front;
    previous = 0.0;
    Synth("Analyzer");
    OSCresponder( s.addr, '/tr', {
        arg time, responder, msg;
        {
            amplitude = msg[3];
            level.value_(amplitude);
            if((amplitude >= 0.3) && (previous < 0.3), {
                if(switch.value == 0,{
                    switch.value_(1);
                    synth = Synth("Granular2", ["buf", buf, "pos", 4.0.rand, "freq", 50.rand + 20]);
                },{
                    switch.value_(0);
                    synth.free;
                });
            });
            previous = amplitude;
        }.defer;
    }).add;
)
```

Wahrscheinlichkeit und Knopf

- Manchmal reagiert mein Knopf nicht!






```
if( 0.3.coin,
{"chikashi".postln},
{"miyama".postln});
```

Quellcode 11

```
(
    var window, switch, amplitude, level, previous, buf, synth;
    buf = Buffer.read(s, "sounds/spoken.wav");
    window = Window.new("map", Rect(200, 100, 250, 200));
    level = Slider(window, Rect(40, 20, 20, 140));
    switch = Button(window, Rect(80, 20, 50, 40))
           .states_([["Aus"],["Ein", Color.black, Color.red]]);
    window.front;
    previous = 0.0;
    Synth("Analyzer");
    OSCresponder( s.addr, '/tr', {
        arg time, responder, msg;
        {
            amplitude = msg[3];
            level.value_(amplitude);
            if((amplitude >= 0.3) && (previous < 0.3), {
                if(switch.value == 0,{
                    if(0.5.coin,{
                        switch.value_(1);
                        synth = Synth("Granular2", ["buf", buf, "pos", 4.0.rand, "freq", 50.rand + 20]);
                    },{"ignored.".postln});
                },{
                    switch.value_(0);
                    synth.free;
                });
            });
            previous = amplitude;
        }.defer;
    }).add;
)
```