# Super Collider 3 für Einsteiger

## Sommersemester 2

Dr. Chikashi Miyama

*30. April 2012*

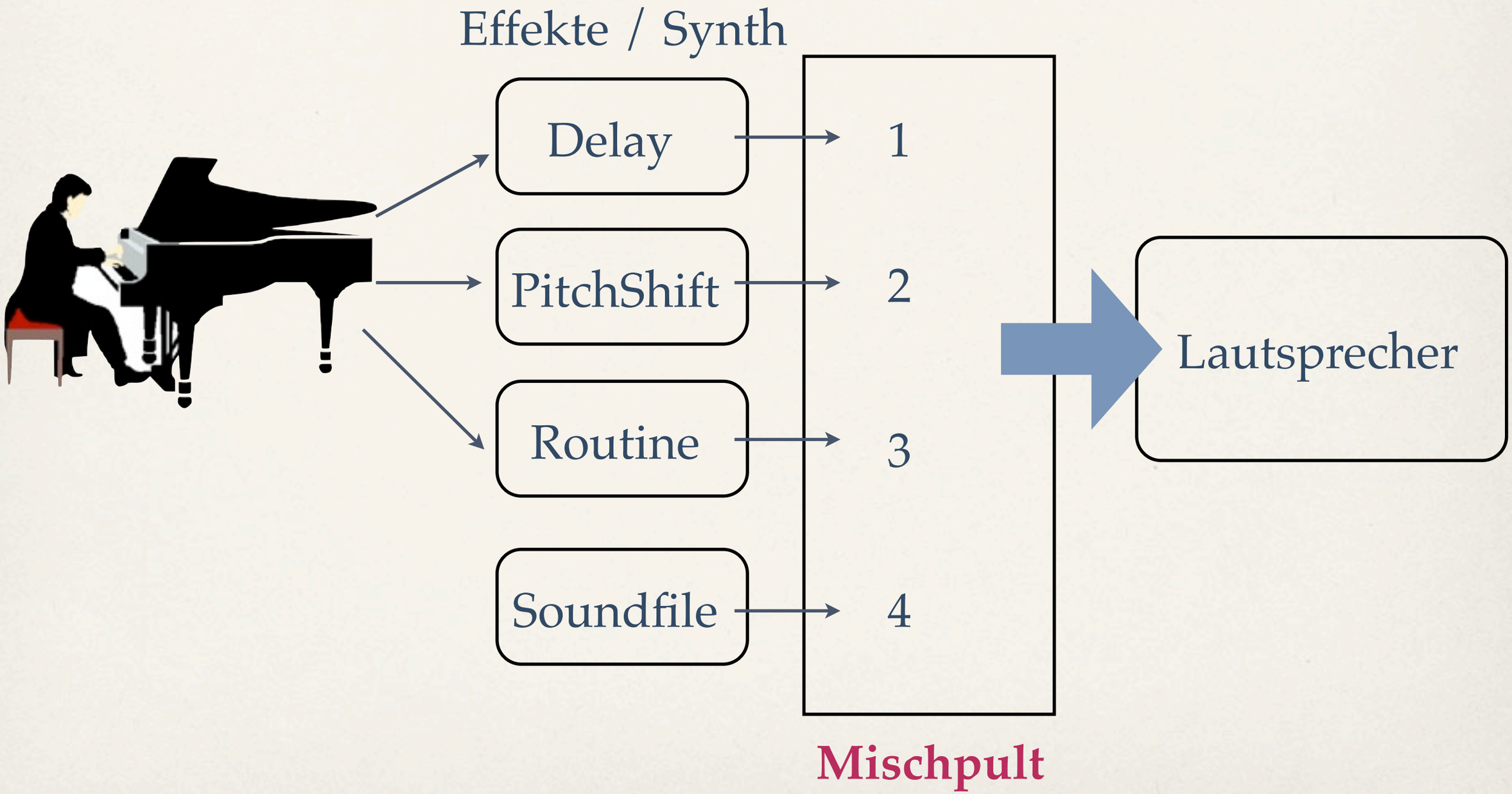
# Überblick

## Live-Elektronik mit SC3

A. *Mischpult (dewdrop_lib)*
B. *Automation*
C. *CueList*
D. *TextField*
E. *Level Meter*
F. *Zuspiel*
G. *Bus*
H. *Routines und Mischpult*
I. *Master Volume*

# A.Mischpult

# A.1 Mischpult



# A.2 Dewdrop_lib

dewdrop_lib... eine SuperCollider Extention

http://www.dewdrop-world.net/sc3/index.php

1. Entpacke **quarks-ddwCommon.tar** und **quarks-ddwMixerChannel.tar**

2. Kopiere die Ordner in

   ~/Library/Application Support/SuperCollider/Extensions

3. Wähle „Lang -> Compile Library“ aus

# A.3 Test Mischpult

Anzahl der Kanäle

Name des Kanals

Input

Output

```
m = MixerChannel("ch1", s, 1, 2);
b = MixingBoard("MixingBoard", nil, m);
```

Name des Fensters

Mixer Skin

MixerChannel

# A.4 Test mit WhiteNoise

```
SynthDef("TestNoise",{
   arg outbus;
   n = WhiteNoise.ar(0.1);
   Out.ar(outbus,n);
}).load;

m = MixerChannel("ch1", s, 1, 2);
b = MixingBoard("MixingBoard", nil, m);
m.play("TestNoise")
```

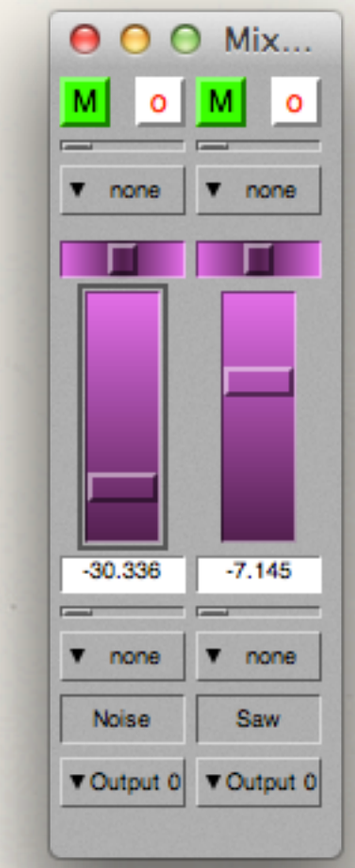
Man kann einen MixerChannel ohne eine MixingBoard verwenden

```
m = MixerChannel("ch1", s, 1, 2);
m.play("TestNoise")
```

# A.5 Zwei Quellen

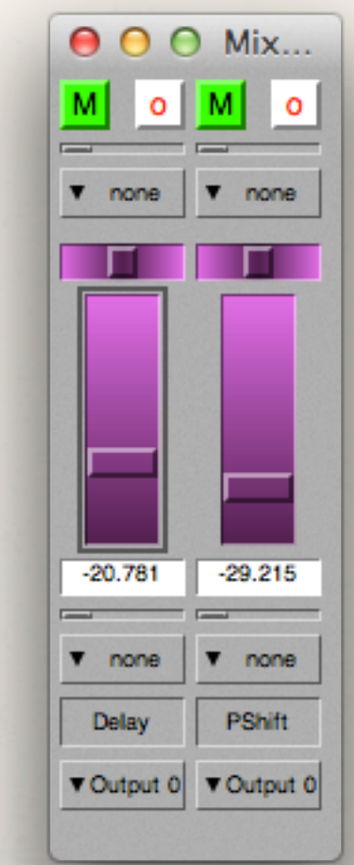
```
SynthDef("TestSaw",{
   arg outbus;
   n = Saw.ar(440, 0.1);
   Out.ar(outbus,n);
}).load;

(
   m = MixerChannel("Noise", s, 1, 2);
   n = MixerChannel("Saw", s, 1, 2);

   a = MixingBoard("MixingBoard", nil, [m,n]);

   m.play("TestNoise");
   n.play("TestSaw");
)
```

Array



# A.6 Zwei Effekte

```
SynthDef("TestDelay",{
    arg outbus;
    i = SoundIn.ar(0);
    d = DelayN.ar(i, 0.7, 0.7);
    Out.ar(outbus,d);
}).load;

SynthDef("TestPS",{
    arg outbus;
    i = SoundIn.ar(0);
    d = PitchShift.ar(i, 0.04, 0.75);
    Out.ar(outbus,d);
}).load;

(
    m = MixerChannel("Delay", s, 1, 2);
    n = MixerChannel("PShift", s, 1, 2);

    a = MixingBoard("MixingBoard", nil, [m,n]);

    m.play("TestDelay");
    n.play("TestPS");
)
```



# B. Automation

# B.1 Automation

Man kann nicht mehrere Schieberegler gleichzeitig per Maus steuern.

**Automation**

# B.2 Automation

- FadeIn / Out

```
(
   m = MixerChannel("Noise", s, 1, 2);
   a = MixingBoard("MixingBoard", nil, m);
   m.play("TestNoise");
)
```

```
m.levelTo(0, 5);
```

← Fade Out

```
m.levelTo(1, 5);
```

← Fade In

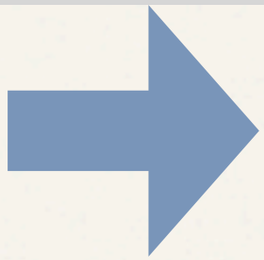
```
m.free
```

# B.3 Automation

- ist die Automation nicht glatt?

```
(
   m = MixerChannel("Noise", s, 1, 2);
   a = MixingBoard("MixingBoard", nil, m);
   m.play("TestNoise");
)
```

```
m.guiUpdateTime = 0.05
```



Achtung: konsumiert CPU

```
m.levelTo(0, 5);
```

Fade Out

```
m.levelTo(1, 5);
```

Fade In

```
m.free
```

# B.4 Automation

- Andere Messages

```
m.levelTo(1, 5);
m.PanTo(1, 5);
```

Ziel Dauer

```
m.levelLine(0.7, 0, 5);
m.PanLine(0.3, 0.5, 5);
```

Start Ziel Dauer

```
m.level = 0.3;
m.Pan = -0.2;
```

Ohne Interpolation

# B.5 StopAuto

```
(
   m = MixerChannel("Noise", s, 1, 2);
   a = MixingBoard("MixingBoard", nil, m);
   m.play("TestNoise");
)
```

```
m.guiUpdateTime = 0.05;
m.levelTo(0, 5);
```

```
m.stopAuto;
```

← unterbrechen

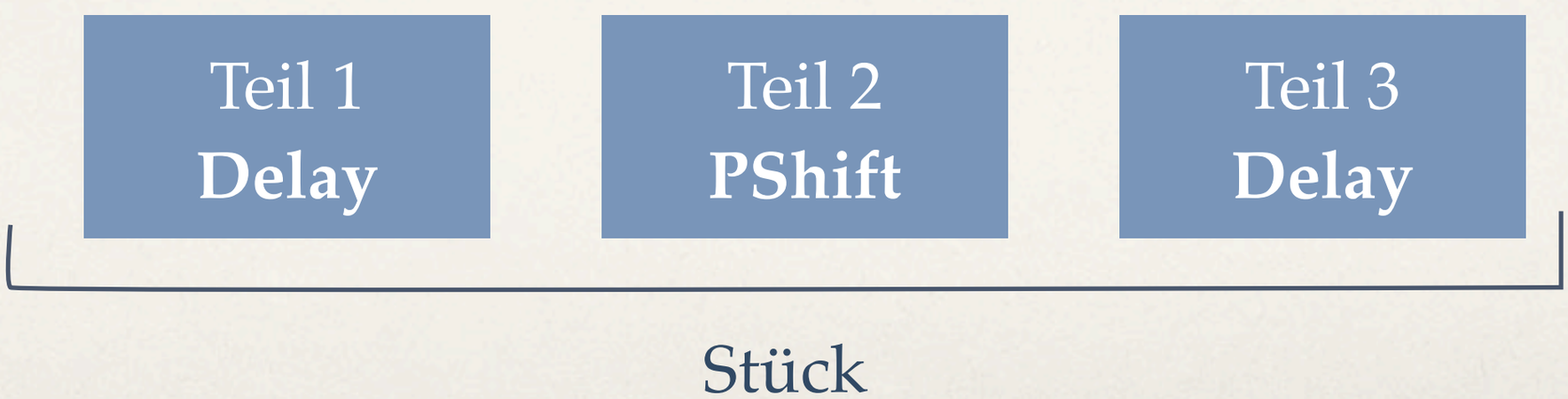
# C. Cue List

# C.1 Cue List

```
(
    m = MixerChannel("Delay", s, 1, 2);
    n = MixerChannel("PShift", s, 1, 2);
    m.guiUpdateTime = 0.05;
    n.guiUpdateTime = 0.05;
    a = MixingBoard("MixingBoard", nil, [m,n]);

    m.play("TestDelay");
    n.play("TestPS");
)
```

## Automation

```
//reset
m.level_(0);
n.level_(0);

//Section 1 - Delay Fade In
m.levelTo(1, 5);

//Section 2 - Delay Fade Out, PShift Fade In
m.levelTo(0, 5);
n.levelTo(1, 5);


//Section 3 - PShift Fade Out, Delay Fade In
n.levelTo(0, 2);
m.levelTo(1, 5);
```



# C.2 Cue List = Routine auf SC3

```
(
   t = Routine({
      "Erste Nachricht".postln;
      0.yield;
      "Zweite Nachricht".postln;
      1.yield;
      "Dritte Nachricht".postln;
      2.yield;
   });
)

t.next;

t.reset;
```

bis nächste „yield“

zurückspulen

# C.3 Mit GUI

```
(
   t = Routine({
      "Erste Nachricht".postln;
      0.yield;
      "Zweite Nachricht".postln;
      1.yield;
      "Dritte Nachricht".postln;
      2.yield;
   });

   w = Window("CueList", Rect(200, 200, 300, 200));
   n = Button(w, Rect(20,20,50,40)).states_([["next"]]);
   n.action = {t.next;};
   r = Button(w, Rect(80,20,50,40)).states_([["rewind"]]);
   r.action = {t.reset;};
   w.front;

)
```

# C.4 Cue List Mit GUI

```
(
    m = MixerChannel("Delay", s, 1, 2);
    n = MixerChannel("PShift", s, 1, 2);
    m.guiUpdateTime = 0.05;
    n.guiUpdateTime = 0.05;
    a = MixingBoard("MixingBoard", nil, [m,n]);

    m.play("TestDelay");
    n.play("TestPS");

)
```

```
(
    t = Routine({
        //reset
        m.level_(0);
        n.level_(0);
        0.yield;

        //Section 1 - Delay Fade In
        m.levelTo(1, 5);
        1.yield;

        //Section 2 - Delay Fade Out, PShift Fade In
        m.levelTo(0, 5);
        n.levelTo(1, 5);
        2.yield;

        //Section 3 - PShift Fade Out, Delay Fade In
        n.levelTo(0, 2);
        m.levelTo(1, 5);
        3.yield;
    });

    w = Window("CueList", Rect(200, 200, 300, 200));
    b = Button(w, Rect(20,20,50,40)).states_([["next"]]);
    b.action = {t.next;};
    c = Button(w, Rect(80,20,50,40)).states_([["rewind"]]);
    c.action = {t.reset;};
    w.front;
)
```

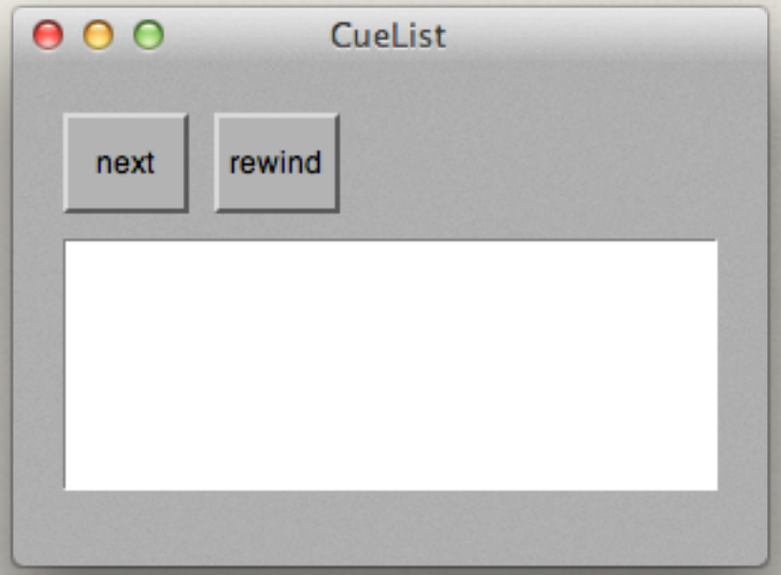
# D. Text Field

# D.1 Mixer, CueList und Automation

```
(
    m = MixerChannel("ch1", s, 1, 2);
    a = MixingBoard("MixingBoard", nil, m);
    m.play("TestNoise");
    m.guiUpdateTime = 0.05;
    m.level = 0;

    c = Routine({
        m.levelTo(0.5, 1);
        0.yield;
        m.levelTo(0, 1);
        1.yield;
        m.levelTo(1, 1);
        2.yield;
    });

    w = Window("CueList", Rect(200, 200, 300, 200));
    n = Button(w, Rect(20,20,50,40)).states_([["next"]]);
    n.action = {c.next;};
    r = Button(w, Rect(80,20,50,40)).states_([["rewind"]]);
    r.action = {c.reset;};
    w.front;
)
```

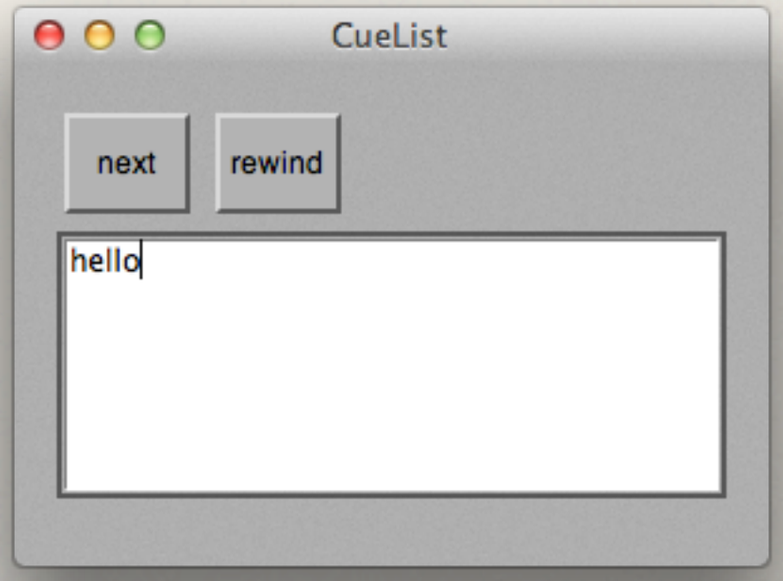
# D.2 Mit ein Textfield

```
(
   w = Window("CueList", Rect(200, 200, 300, 200));
   n = Button(w, Rect(20,20,50,40)).states_([["next"]]);
   n.action = {c.next;};
   r = Button(w, Rect(80,20,50,40)).states_([["rewind"]]);
   r.action = {c.reset;};
   t = TextField(w, Rect(20, 70, 260, 100));
   w.front;
)
```



# D.3 Kommentare

```
t.string =  "Hallo";
```



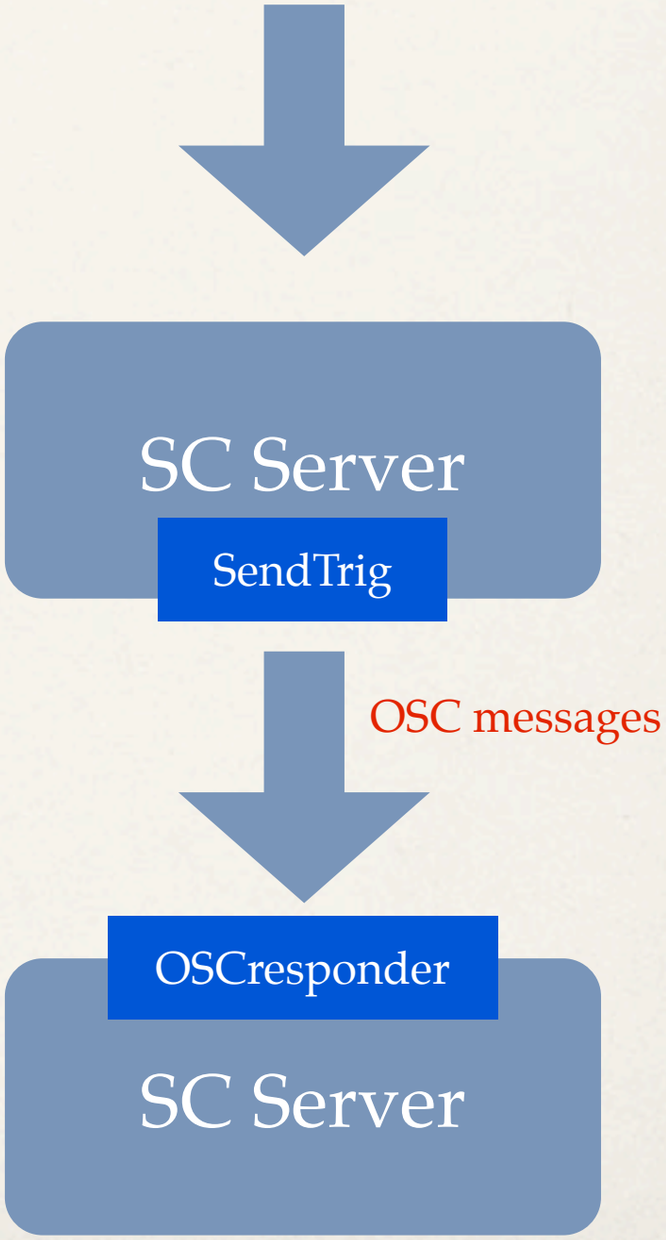
# D.4 Cue-List mit Kommentaren

```
(
    m = MixerChannel("ch1", s, 1, 2);
    a = MixingBoard("MixingBoard", nil, m);
    m.play("TestNoise");
    m.guiUpdateTime = 0.05;
    m.level = 0;

    c = Routine({
        m.levelTo(0.5, 1);
        t.string = "Section1 - Part1";
        0.yield;

        m.levelTo(0, 1);
        t.string = "Section1 - Part2";
        1.yield;

        m.levelTo(1, 1);
        t.string = "Section1 - Part3";
        2.yield;
    });

    w = Window("CueList", Rect(200, 200, 300, 200));
    n = Button(w, Rect(20,20,50,40)).states_([["next"]]);
    n.action = {c.next;};
    r = Button(w, Rect(80,20,50,40)).states_([["rewind"]]);
    r.action = {c.reset;};
    t = TextField(w, Rect(20, 70, 260, 100));
    w.front;
)
```

# E. Level Meter

# E.1 OscResponder

```
{
   i = SoundIn.ar(0);
   a = Amplitude.kr(i);
   t = Impulse.kr(50);
   SendTrig.kr(t, 0, a);
}.play;

(
   o = OSCresponder(s.addr, '/tr',
   {arg time, resp, msg;
      msg[3].postln;
   }).add;
   w.front;
)
```



# E.3 Level Meter mit GUI

```
{
   i = SoundIn.ar(0);
   a = Amplitude.kr(i);
   t = Impulse.kr(50);
   SendTrig.kr(t, 0, a);
}.play;

(
   w = Window("Levels", Rect(40, 180, 40, 180));
   a = SCLevelIndicator(w, Rect(10, 10, 20, 160));
   o = OSCresponder(s.addr, '/tr', {arg time, resp, msg;
      {
         a.value = msg[3];
      }.defer;
   }).add;
   w.front;
)
```

# E.4 Stereo Level Meter

```
{
    i = SoundIn.ar([0, 1]);
    a = Amplitude.kr(i);
    t = Impulse.kr(50);
    SendTrig.kr(t, [0, 1], a);
}.play;

(
    w = Window("Levels", Rect(40, 180, 80, 180));
    l = SCLevelIndicator(w, Rect(10, 10, 20, 160));
    r = SCLevelIndicator(w, Rect(40, 10, 20, 160));

    o = OSCresponder(s.addr, '/tr', {arg time, resp, msg;
        {
                if(msg[2] == 0,{
                    l.value = msg[3];
                },{
                    r.value = msg[3];
                });
        }.defer;
    }).add;
    w.front;
)
```

Linker Kanal / Rechter Kanal

ID Nr. : 0 1

ID

# E.5 Mit Farben

```
{
    i = SoundIn.ar([0,1]);
    a = Amplitude.kr(i);
    t = Impulse.kr(50);
    SendTrig.kr(t, [0,1], a);
}.play;

(
    w = Window("Levels", Rect(40, 180, 80, 180));
    l = SCLevelIndicator(w, Rect(10, 10, 20, 160));
    r = SCLevelIndicator(w, Rect(40, 10, 20, 160));
    l.warning = 0.6;
    r.warning = 0.6;
    l.critical = 0.9;
    r.critical = 0.9;

    o = OSCresponder(s.addr, '/tr', {arg time, resp, msg;
        {
                if(msg[2] == 0,{
                    l.value = msg[3];
                },{
                    r.value = msg[3];
                });
        }.defer;
    }).add;
    w.front;
)
```

# F. Zuspiel

# F.1 DiskIn

```
b = Buffer.cueSoundFile(s, "sounds/chopin.wav", 0, 1);
{ DiskIn.ar(1, b) }.play;
```

Anzahl der Kanäle

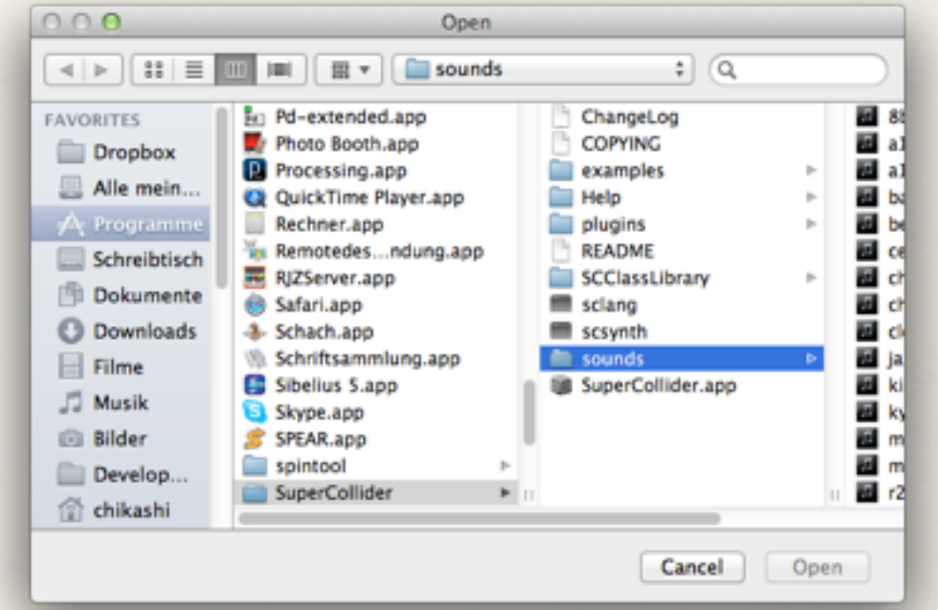
```
b = Buffer.cueSoundFile(s, "sounds/chopin.wav", 0, 1);
{ DiskIn.ar(1, b, 3) }.play;
```

Loop ( 3 mal)

# F.2 OpenPanel

```
Dialog.getPaths({ arg paths;
    paths.postln;
});
```

```
Dialog.getPaths({ arg paths;
    paths[0].postln;
});
```

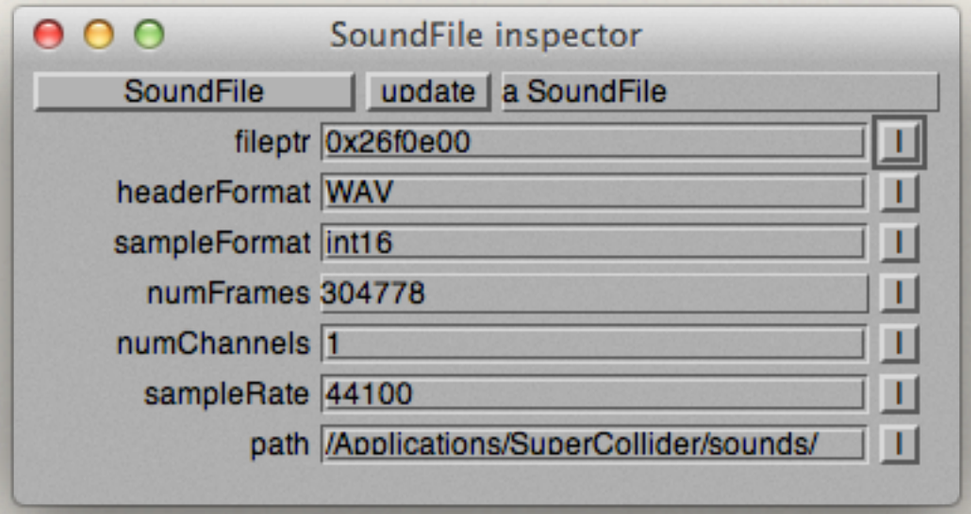



```
Dialog.getPaths({arg path; p = path[0]});

(
  b = Buffer.cueSoundFile(s, p, 0, 1);
  { DiskIn.ar(1, b) }.play;
)
```

# F.3 Informationen eines Soundfiles

- SoundFile

```
Dialog.getPaths({arg path; p = path[0]});

k = SoundFile.openRead(p);
k.inspect;
```



# F.4 Soundfile-Spieler mit GUI

```
SynthDef("SfPlay", {
   arg bufnum;
   d = DiskIn.ar(1, bufnum);
   Out.ar(0, d);
}).load;

(
   w = Window("SFPlayer", Rect(300, 300 ,120, 50));
   Button(w, Rect(10,10,40,20)).states_([["Open"]]).action_({
      Dialog.getPaths({arg path; p = path[0]});
   });
   Button(w, Rect(60,10,40,20)).states_([["Play"],["Stop"]]).action_({
      arg butt;
      if(butt.value == 1, {
         b = Buffer.cueSoundFile(s, p, 0 ,1);
         x = Synth("SfPlay", ["bufnum", b]);
      },{
         x.free;
      });
   });
```

# F.5 Soundfiles und Cue-List

```
(
    b = Buffer.cueSoundFile(s, "sounds/chopin.wav", 0, 1);
    m = MixerChannel("ch1", s, 1, 2);
    a = MixingBoard("MixingBoard", nil, m);
    m.guiUpdateTime = 0.05;
    m.level = 0;

    c = Routine({
        m.levelTo(1, 1);
        t.string = "Fade In";
        m.play("SfPlay", ["bufnum", b]);
        0.yield;

        m.levelTo(0, 1);
        t.string = "Fade Out";
        1.yield;
    });

    w = Window("CueList", Rect(200, 200, 300, 200));
    n = Button(w, Rect(20,20,50,40)).states_([["next"]]);
    n.action = {c.next;};
    r = Button(w, Rect(80,20,50,40)).states_([["rewind"]]);
    r.action = {c.reset;};
    t = TextField(w, Rect(20, 70, 260, 100));
    w.front;
)
```

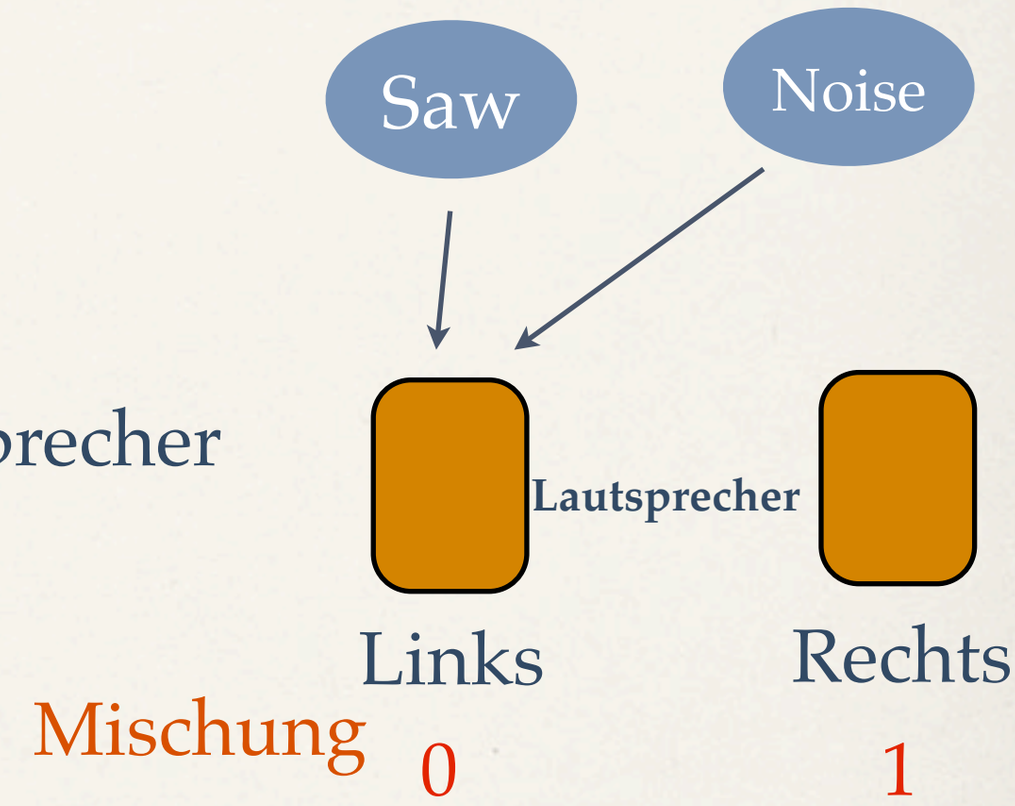
# G. Bus

# G.1 Mischung mit Bus

```
SynthDef("Saw",{
   a = Saw.ar(220, 0.3);
   Out.ar(0, a);
}).load;


SynthDef("Noise",{
   n = WhiteNoise.ar(0.3);
   Out.ar(0, n);
}).load;


(
   Synth("Saw");
   Synth("Noise");
)
```

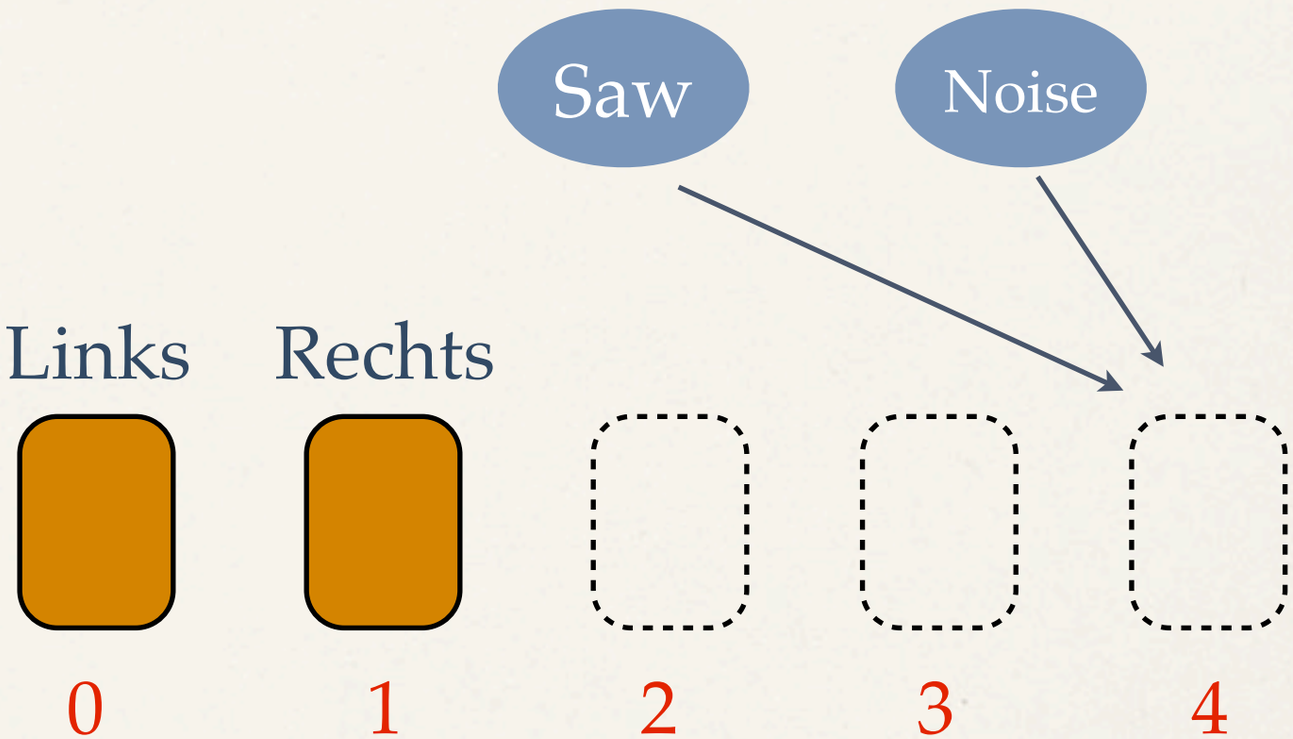
Linker Lautsprecher

Linker Lautsprecher



# G.2 Mischung

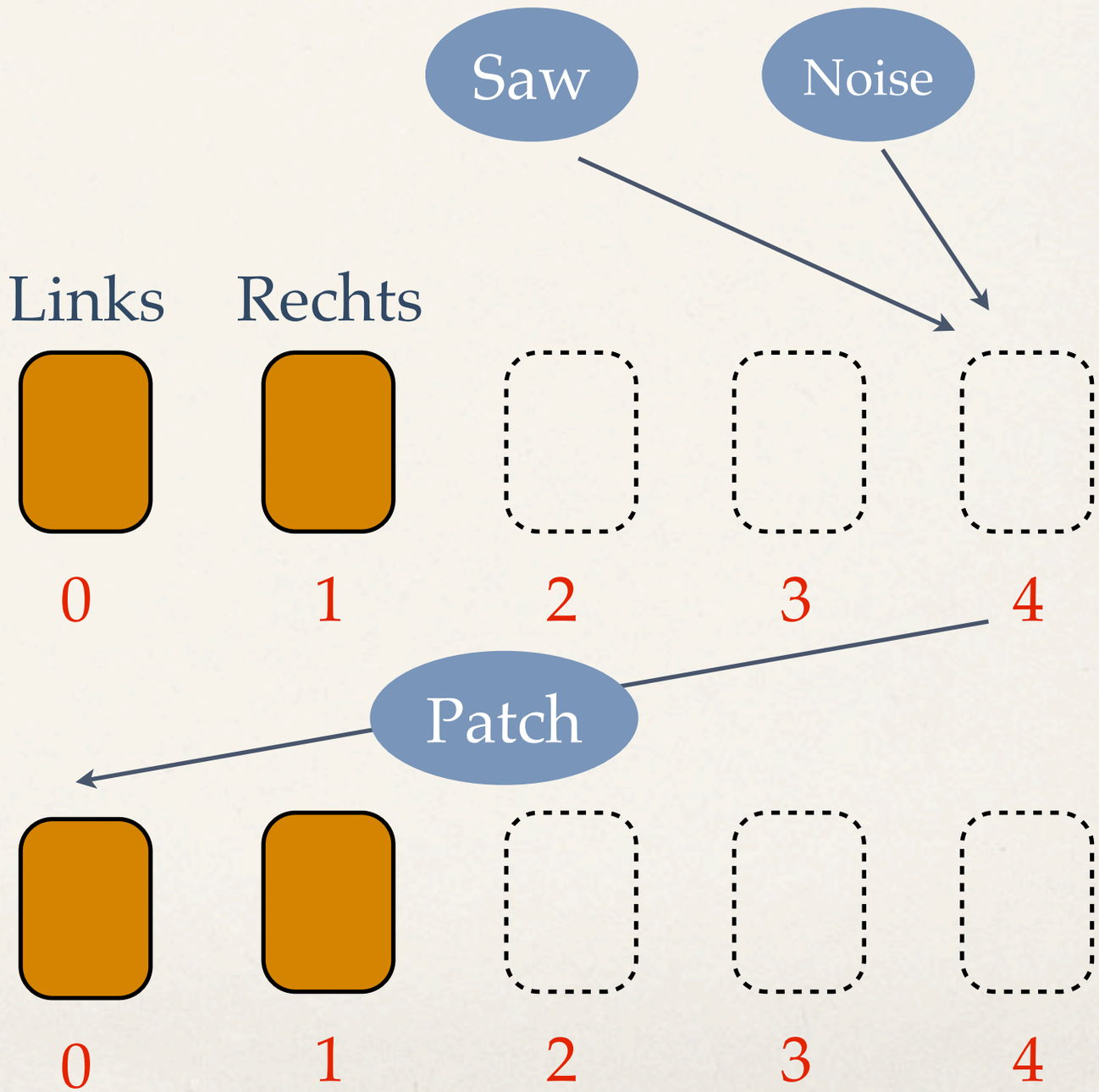
```
SynthDef("Saw",{
   a = Saw.ar(220, 0.3);
   Out.ar(4, a);
}).load;


SynthDef("Noise",{
   n = WhiteNoise.ar(0.3);
   Out.ar(4, n);
}).load;



(
   Synth("Saw");
   Synth("Noise");
)
```



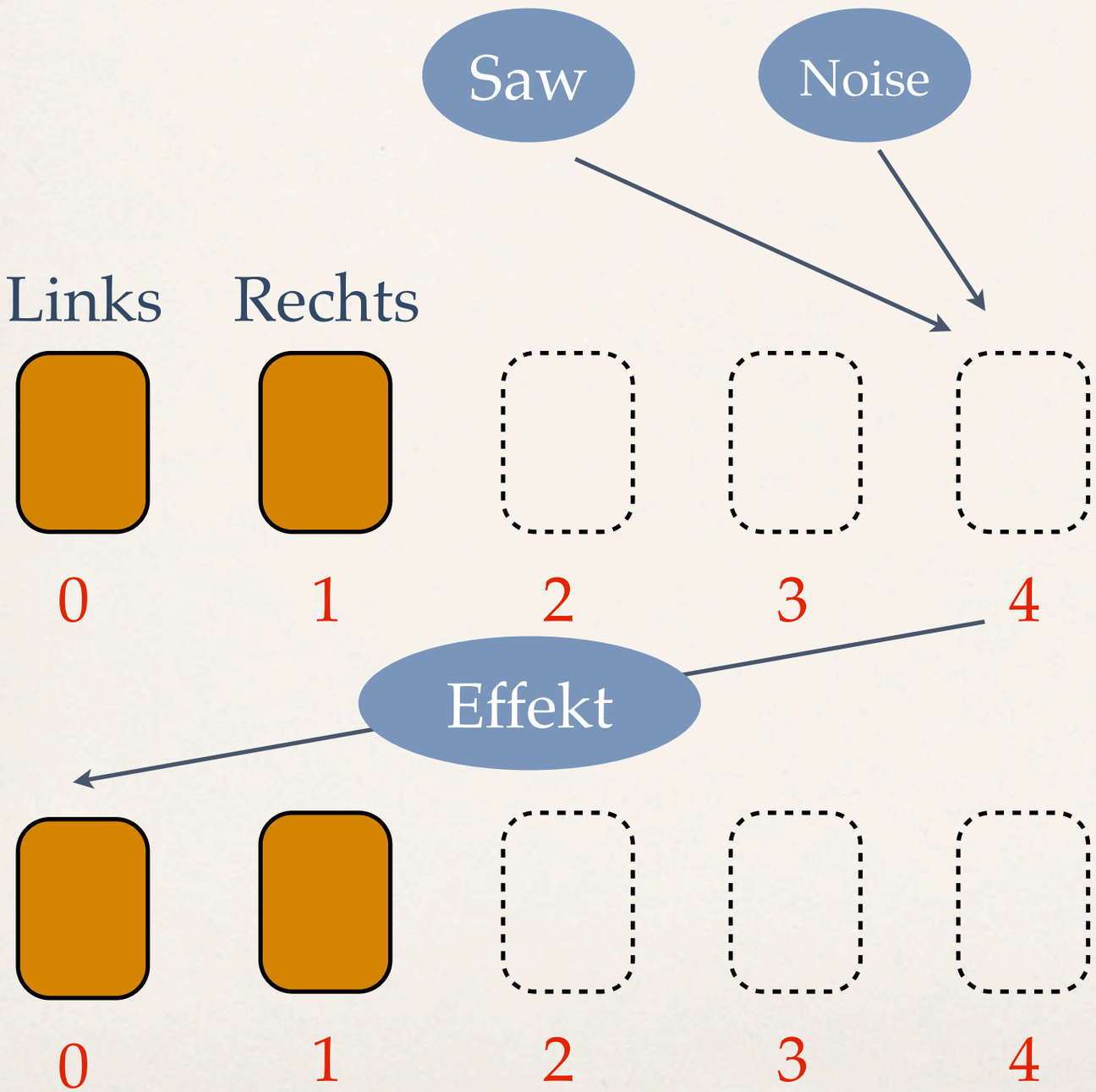

produziert keine Klänge...

# G.3 In.ar

```
SynthDef("Saw",{
   a = Saw.ar(220, 0.3);
   Out.ar(4, a);
}).load;

SynthDef("Noise",{
   n = WhiteNoise.ar(0.3);
   Out.ar(4, n);
}).load;

SynthDef("Patch",{
   i = In.ar(4);
   Out.ar(0, i);
}).load;

(
   Synth("Patch");
   Synth("Saw");
   Synth("Noise");
)
```

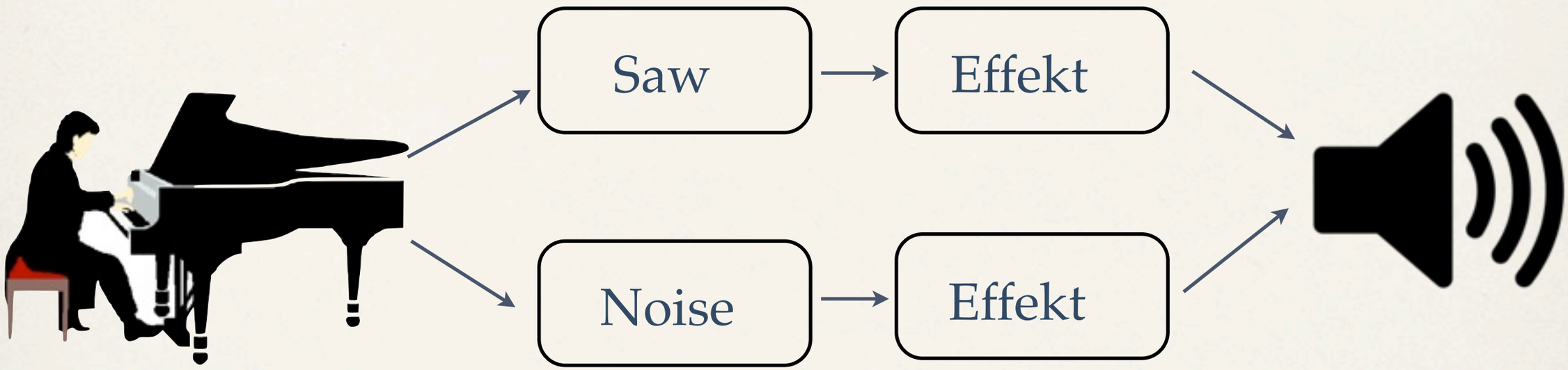


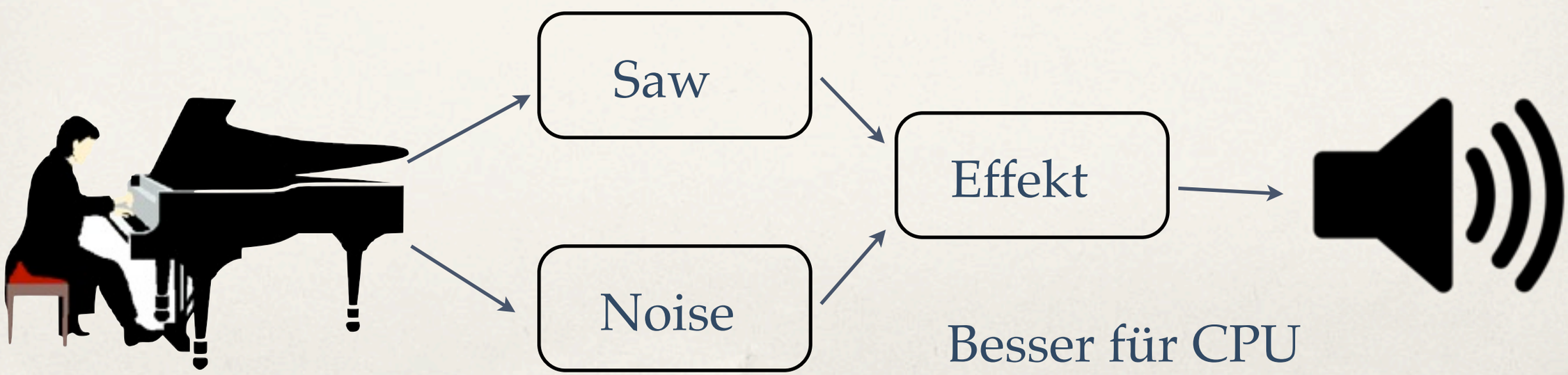
# G.4. Warum brauchen wir ein Bus?




Man kann statt „Patch“ einen Effekt verwenden

# G.5 Der Vorteil

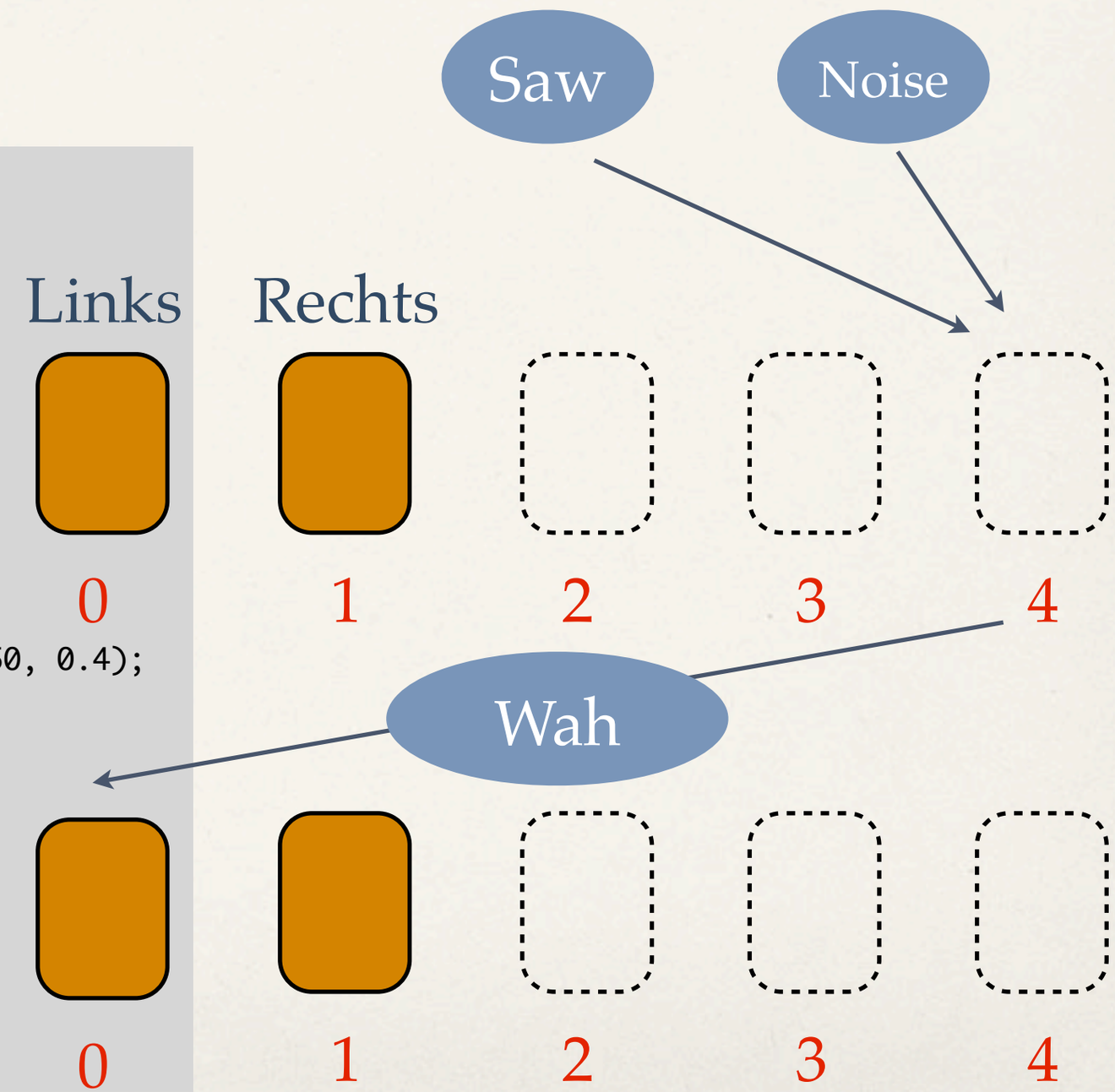



Besser für CPU

# A.6 Wah Wah

```
SynthDef("Saw",{
    a = Saw.ar(220, 0.3);
    Out.ar(4, a);
}).load;

SynthDef("Noise",{
    n = WhiteNoise.ar(0.3);
    Out.ar(4, n);
}).load;

SynthDef("Wah",{
    i = In.ar(4);
    b = BPF.ar(i , SinOsc.ar(10, 0, 200)  + 450, 0.4);
    Out.ar(0, b);
}).load;

(
    Synth("Wah");
    Synth("Saw");
    Synth("Noise");
)
```

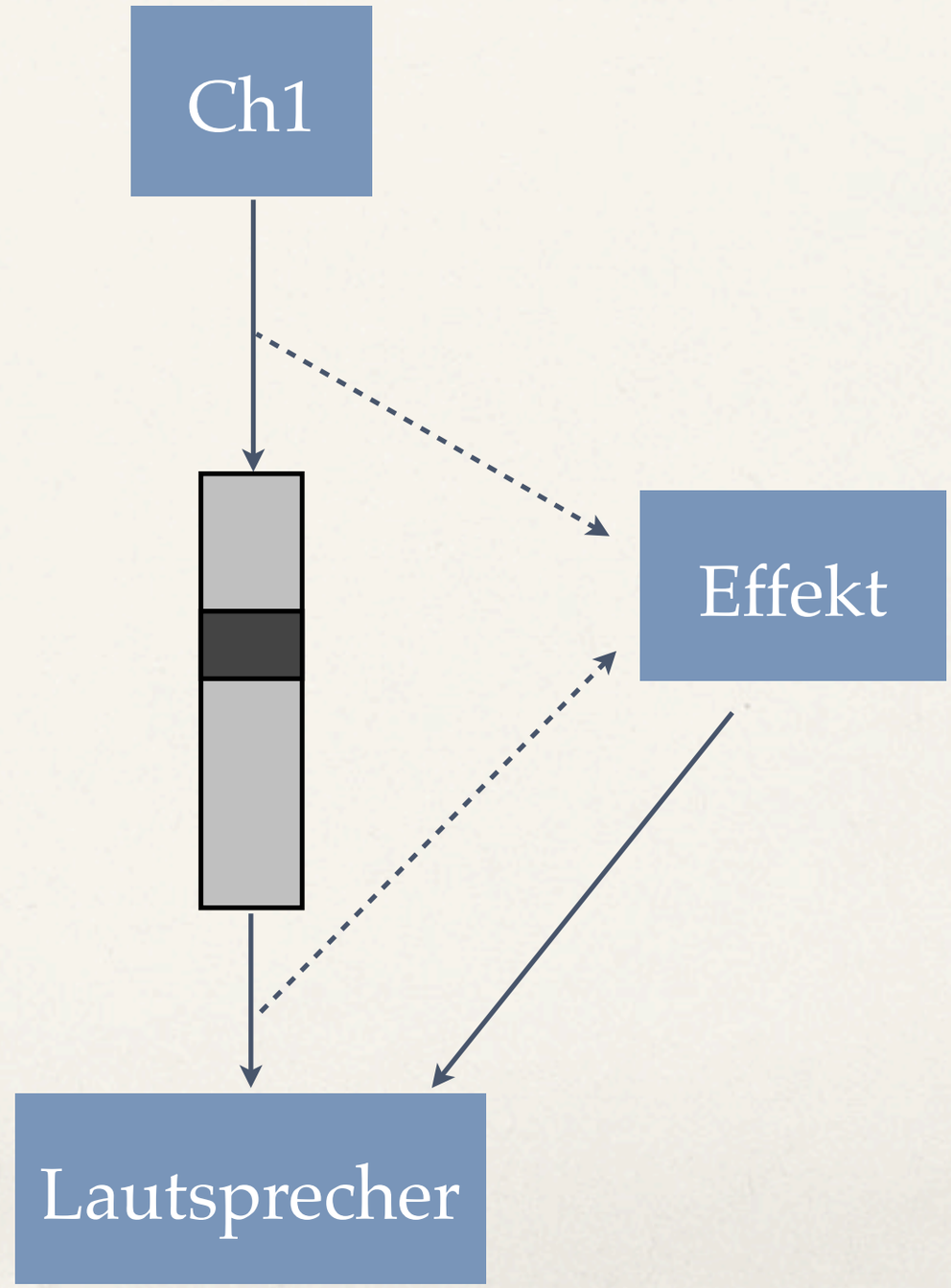


# G.7 Bus mit der Mischpult

- Pre Fader / Post Fader

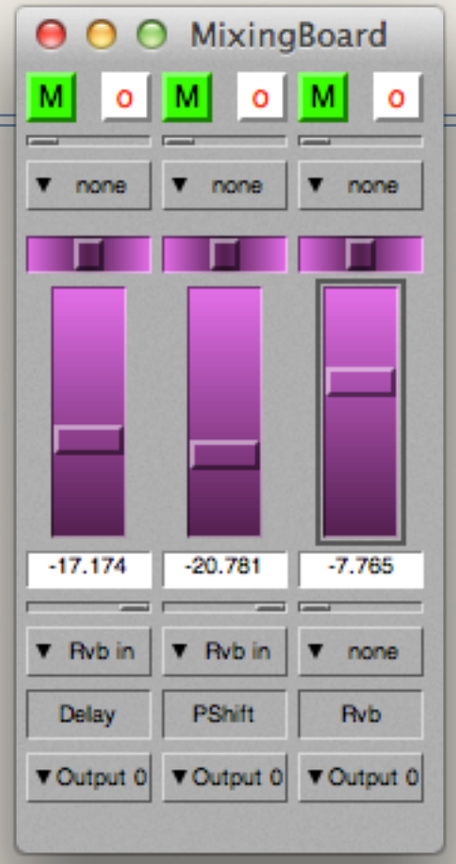
Pre Fader Send

Post Fader Send



# G.8 Die typische Nutzung

- Reverb Send




```
SynthDef("TestReverb",{
    arg outbus;
    a = In.ar(outbus, 2);
    v = FreeVerb2.ar(a[0], a[1], 1, 0.6, 0.5);
    Out.ar(outbus, v);
}).load;

(
    m = MixerChannel("Delay", s, 1, 2);
    n = MixerChannel("PShift", s, 1, 2);
    r = MixerChannel("Rvb", s, 1, 2);
    a = MixingBoard("MixingBoard", nil, [m,n,r]);

    m.play("TestDelay");
    n.play("TestPS");
    r.playfx("TestReverb");
)
```

Delay

PShift

Reverb

Besser für CPU

# H. Routines und Mischpult

# H.1 Zwei Routines

```
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;
```

```
(
    n = 0.1;
    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;
)
```

# H.2 Zwei Routines u. Zwei Kanäle

```
SynthDef("technoSynth",{
    arg outbus, 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(outbus, a );
}).load;
```

```
(
m = MixerChannel("melody", s);
b = MixerChannel("bass", s);
MixingBoard("mix", nil, [m, b]);

n = 0.1;
Routine({
    loop{
        t = [0, 5, 7].choose;
        4.do{
            m.play("technoSynth",["freq", (60+t).midicps, "amp", 0.3]);
            n.wait;
            m.play("technoSynth",["freq", (72+t).midicps, "amp", 0.3]);
            n.wait;
            m.play("technoSynth",["freq", (67+t).midicps, "amp", 0.3]);
            n.wait;
            m.play("technoSynth",["freq", (68+t).midicps, "amp", 0.3]);
            n.wait;
        }
    };
}).play;
Routine({
    loop{
        2.do{
            b.play("technoSynth",["freq", (36+t).midicps, "amp", 0.3]);
            (n*3).wait;
            b.play("technoSynth",["freq", (39+t).midicps, "amp", 0.3]);
            (n*3).wait;
            b.play("technoSynth",["freq", (36+t).midicps, "amp", 0.3]);
            (n*2).wait;
        }
    };
}).play;
)
```

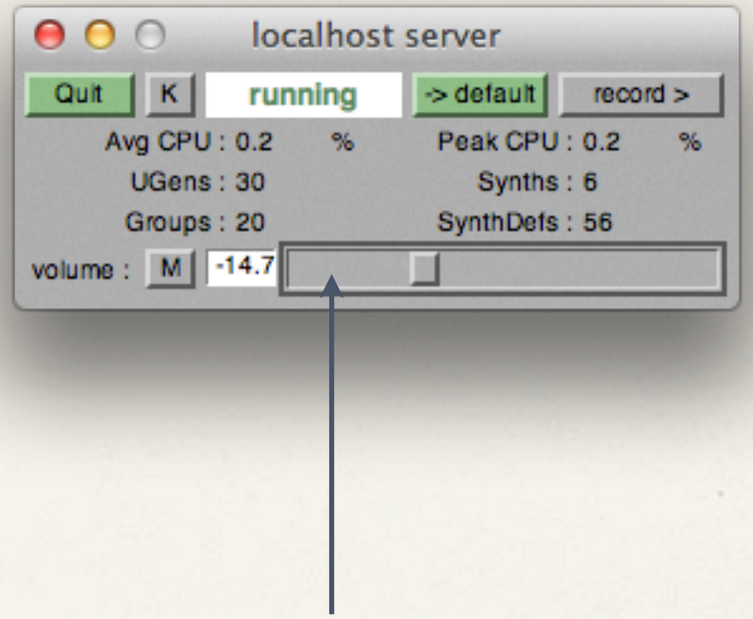
# I.1 Master Volume

- s.volume_(level)

s.volume_(0)

s.volume_(6)

s.volume_(-90)




Master Volume

# Tipps

- Debugging
- SuperCollider und Live Video
- Randomisierung
- Tipps für Live-Elektronik Komposition

# Debugging

# Typische Bugs

- Message -------- not understand

Index 0 1 2

```
a = [3, 4, 5];
a.at(2);
a.at(0);

o.at(2);
```

*a* ist ein Array

5

3

Was ist *o*?

```
ERROR: Message 'at' not understood.
RECEIVER:
   nil
ARGS:
   Integer 2
CALL STACK:
    DoesNotUnderstandError:reportError   096A4410
        arg this = <instance of DoesNotUnderstandError>
    Nil:handleError   096A43B0
        arg this = nil
        arg error = <instance of DoesNotUnderstandError>
    Thread:handleError   096A42F0
        arg this = <instance of Thread>
        arg error = <instance of DoesNotUnderstandError>
    Object:throw   096A4290
        arg this = <instance of DoesNotUnderstandError>
    Object:doesNotUnderstand   096A4230
        arg this = nil
        arg selector = 'at'
        arg args = [*1]
    Interpreter:interpretPrintCmdLine   09778120
        arg this = <instance of Interpreter>
        var res = nil
        var func = <instance of Function>
        var code = "
```

# Typische Bugs

- Parse error

```
(3  - (4 / 2.1)))
```

← zu viel Klammer

```
• ERROR: Parse error
   in file 'selected text'
   line 1 char 17:
  (3  - (4 / 2.1)))•
-----------------------------------
unmatched ')'
```

```
(3  - (4 / 2.1)}
```

← Der Art des Klammeres ist nicht richtig

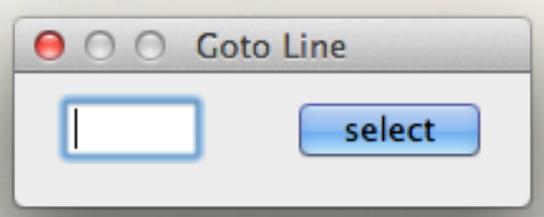
```
x = 4 + 5 * 0.2
y = x / 4;
```

← Kein Strichpunkt (;)

# Goto Line

```
• ERROR: Parse error
   in file 'selected text'
   line 1 char 17:
  (3  - (4 / 2.1)))•
-----------------------------------
unmatched ')'
```

- Command + Komma (,)



# Balance

```
(
  {
  })
  {
    (
    )
  }
)
```

Fehler

Typische SC3 Programme enthalten viele Klammern

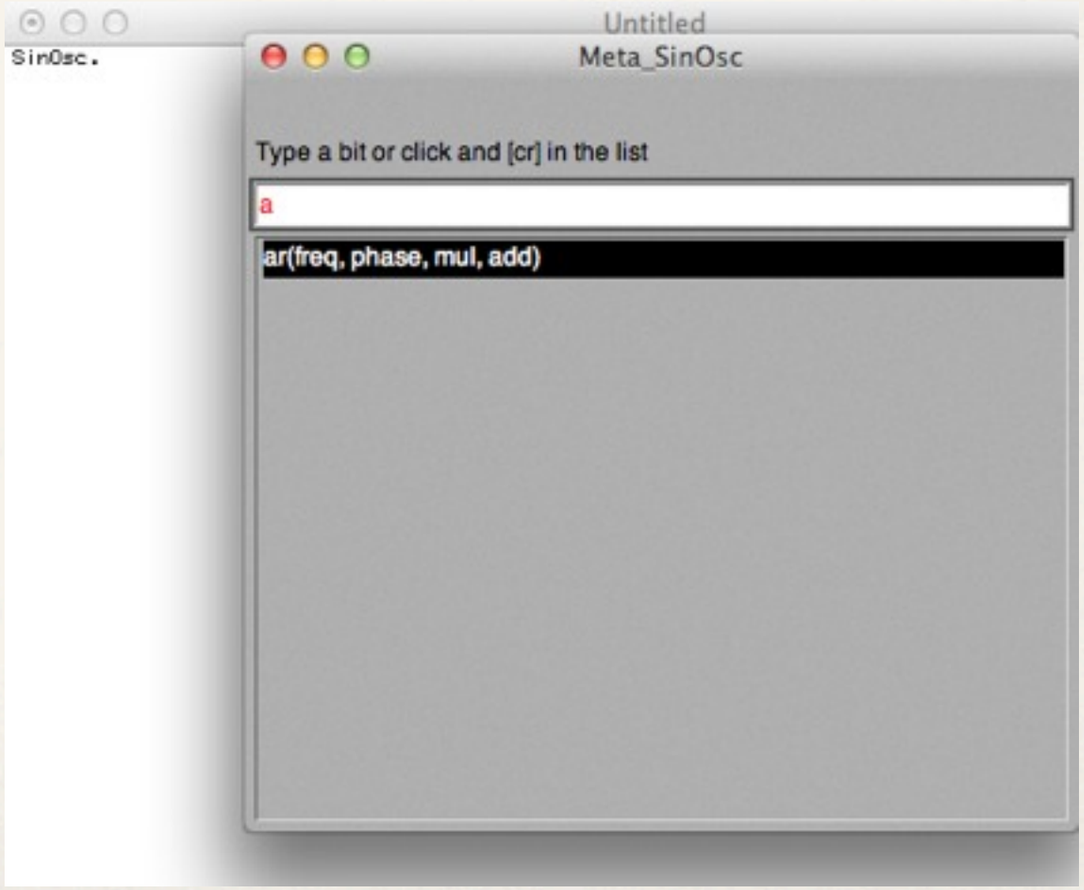
- Command + Shift + B

  hilft Fehler mit Klammern zu finden.

# Wie kann man Tippfehler vermeiden?

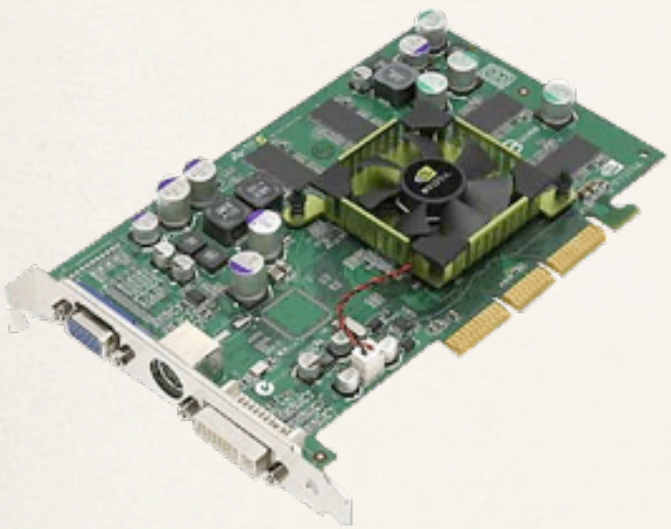
- Auto Completion

```
Document.autoComplete
```



# SuperCollider und Live Video

# Computergraphik



Eine Graphikkarte in einem Computer prozessiert 2D oder 3D Graphik

Zwei Größte Unternehmen

**AMD (früherATI)**

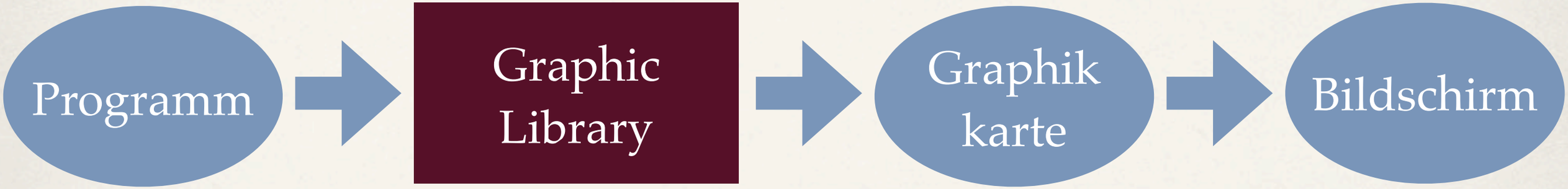
Merken: Radeon, FirePro

**NVidia**

Merken: GeForce, Quadro, Tesla

# Computergraphik

Wie benutzt mein Computer die Grafikkarte ?

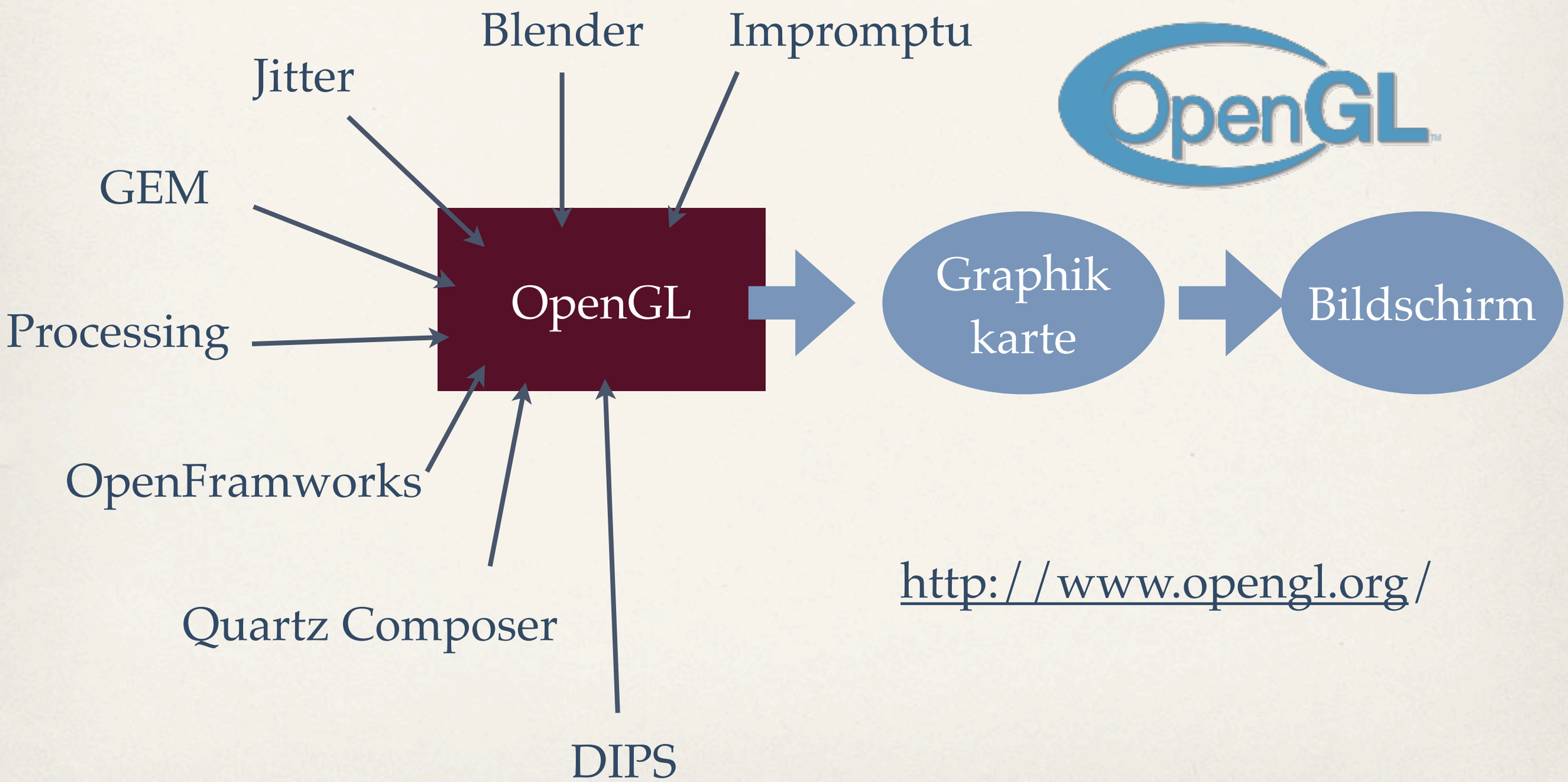



Zwei bekannteste Graphic Libraries

**OpenGL** ... Library auf C-Sprach
Mac, Linux, Windows für CAD (Computer-aided Design)

**Direct X** ... nur für Windows.

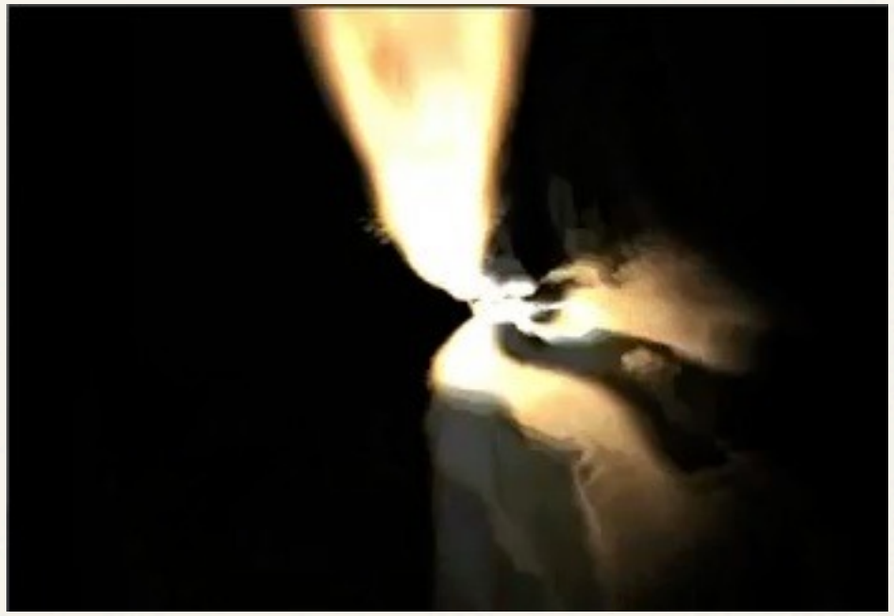
# Fast alle Live-Video-Softwares auf MacOX benutzen OpenGL




http://www.opengl.org/

# Liquid Flame (2010)

Open GL Computergraphik



# Live-Video-Software auf MacOS

| | Form | Sprache | Preis | Schwierigkeit | Vorteil |
|---|---|---|---|---|---|
| **Processing** | Programmierumgebung | C-like | kostenlos | 3 | Bücher auf Deutsch |
| **Quartz Composer** | Dataflow | GUI | kostenlos | 2 | Apple-Produkt |
| **GEM** | Dataflow | GUI | kostenlos | 2 | Michael benutzt |
| **Open Frameworks** | C++ Bibrothek | C ++ | kostenlos | 5 | mac, win, linux kompatibilität |
| **Jitter** | Dataflow | GUI | $250 | 2 | viele Benutzer FANCY! |
| **Blender** | 3D- Modeling, Animation software | GUI | kostenlos | 3 | Object-Modeling |

# Software für Live-Video

- Processing

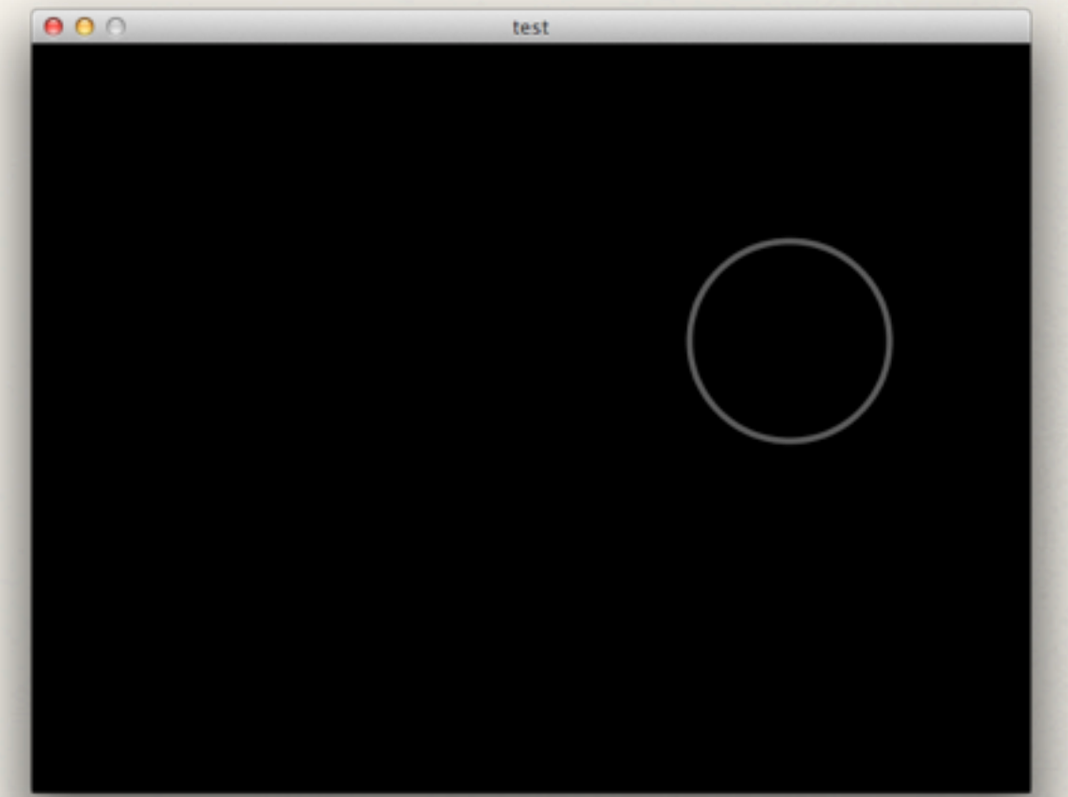
  http://processing.org/

  C-ähnliche Sprache
  Leicht
  Bücher auf Deutsch http://www.amazon.de/dp/3897219972
  Viele Bibliotheken
  Viele Beispiele
  Akzeptiert OSC
  Kostenlos

# OSC message an Processing

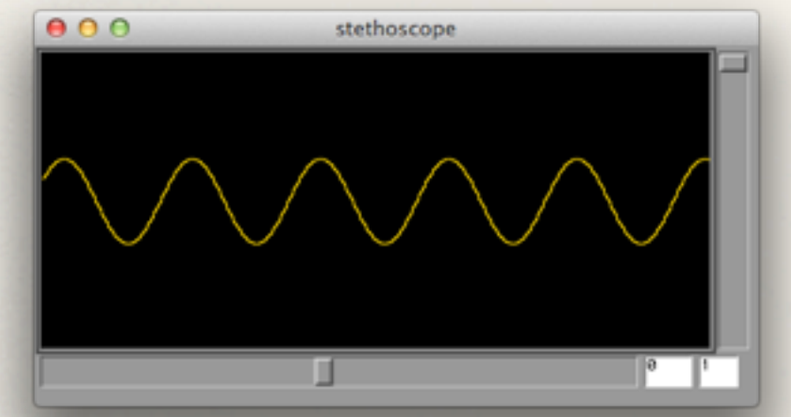
SC3

```
n = NetAddr("127.0.0.1", 12000);
Routine({
    loop{
        n.sendMsg("/test", rrand(40, 600), rrand(100, 300));
        1.5.wait;
    };
}).play
```



# Randomisierung

# Elektronische Klänge sind oft zu statisch




```
{
   SinOsc.ar(440, 0, 0.1);
}.play;
```

Lösungen

Hüllkurve
Filter
Delay
Panning
usw.

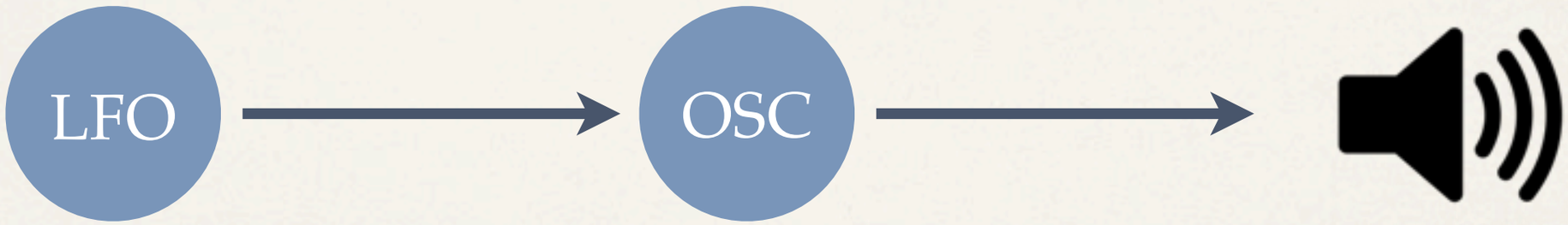
**LFO und Rondomisierung**

# LFO

- LFO = Low Frequency Oscillator

  http://de.wikipedia.org/wiki/Low_Frequency_Oscillator

LFO steuert einen Parameter eines andren Oszillators.



# LFO- Beispiele

```
{
   SinOsc.ar(440, 0, 0.1);
}.play;
```

```
{
   r = SinOsc.ar(5, 0, 10);
   SinOsc.ar(440 + r, 0, 0.1);
}.play;
```

## Mit einer Hüllkurve

```
{
   l = SinOsc.ar(5, 0, 10);
   e = EnvGen.ar(Env([0,1,1,0],[1,1,1]));
   m = l* e;
   SinOsc.ar(440 + m, 0, 0.1);
}.play;
```

# 2x LFO

```
{
   var lfo1, lfo2;
   lfo1 = SinOsc.ar(3, 0, 10);
   lfo2 = SinOsc.ar(2, 0, 10);

   SinOsc.ar(440 + lfo1 + lfo2, 0, 0.3);
}.play
```

```
{
   var lfo1, lfo2;
   lfo1 = SinOsc.ar(4, 0, 10);
   lfo2 = SinOsc.ar(13, 0, 10);

   SinOsc.ar(440 + lfo1 + lfo2, 0, 0.3);
}.play
```

```
{
   var lfo1, lfo2;
   lfo1 = LFSaw.ar(5 ,0, 10);
   lfo2 = LFPulse.ar(3, 0, 0.5, 10);

   SinOsc.ar(440 + lfo1 + lfo2, 0, 0.3);
}.play
```

mit unterschiedlichen Oszillatoren

Viele Software-Synthesizer verwenden diese Technik

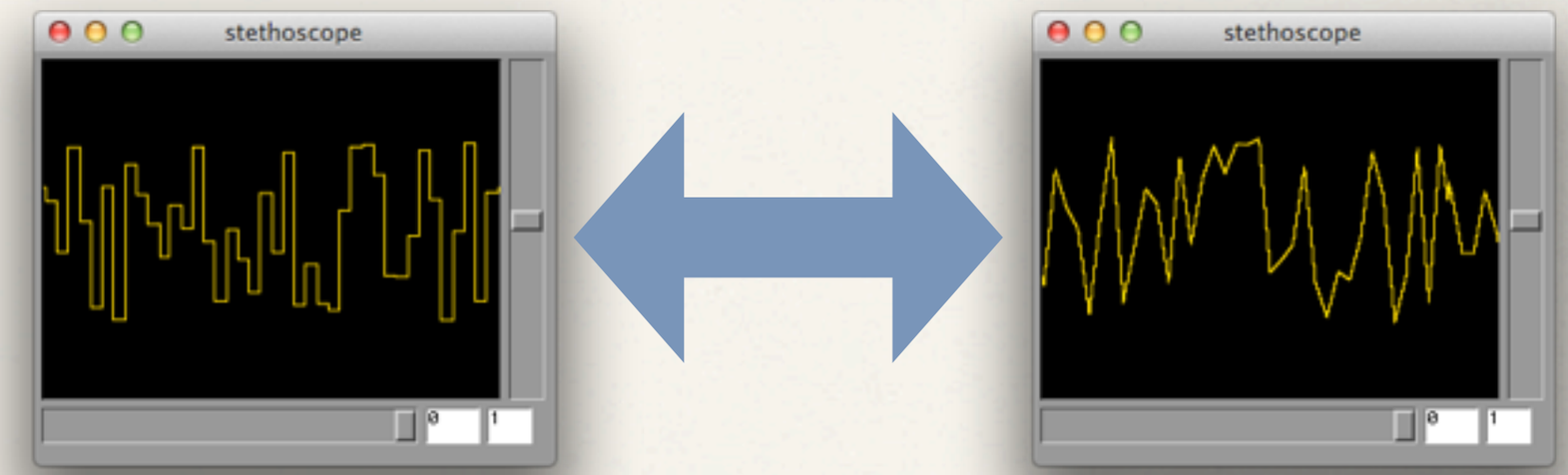
# LFNoise0, LFNoise1, LFNoise2

- LFNoise0.ar
- LFNoise1.ar
- LFNoise2.ar

```
{
    f = LFNoise0.ar(20, 30);
    Saw.ar(440 + f, 0 , 0.3);
}.play

{
    f = LFNoise1.ar(20, 30);
    Saw.ar(440 + f, 0 , 0.3);
}.play

{
    f = LFNoise2.ar(20, 30);
    Saw.ar(440 + f, 0 , 0.3);
}.play
```

# LFNoise0, LFNoise1, LFNoise2

- Interpolation zwischen zwei LFNoise




```
{
   f = LFNoise0.ar(20, 50);
   g = LFNoise1.ar(20, 50);
   x = LinXFade2.ar(f,g, MouseX.kr(-1, 1));
   Saw.ar(440 + x, 0.3);
}.play
```

LinXFade2

... Linear Cross Fade

# TRand

- TRand

```
{
   i = Impulse.ar(10);
   r = TRand.ar(440, 880, i);
   SinOsc.ar(r + 440, 0.0, 0.1);
}.play;
```

- Graduell schneller

```
{
   i = Impulse.ar(Line.kr(10, 50, 10));
   r = TRand.ar(440, 880, i);
   SinOsc.ar(r + 440, 0.0, 0.1);
}.play;
```

# Ramp.ar

```
{
   i = Impulse.ar(10);
   r = TRand.ar(440, 880, i);
   SinOsc.ar(r + 440, 0.0, 0.1);
}.play;
```

Ohne Ramp

```
{
   i = Impulse.ar(10);
   r = TRand.ar(440, 880, i);
   r = Ramp.ar(r, 0.1);
   SinOsc.ar(r + 440, 0.0, 0.1);
}.play;
```

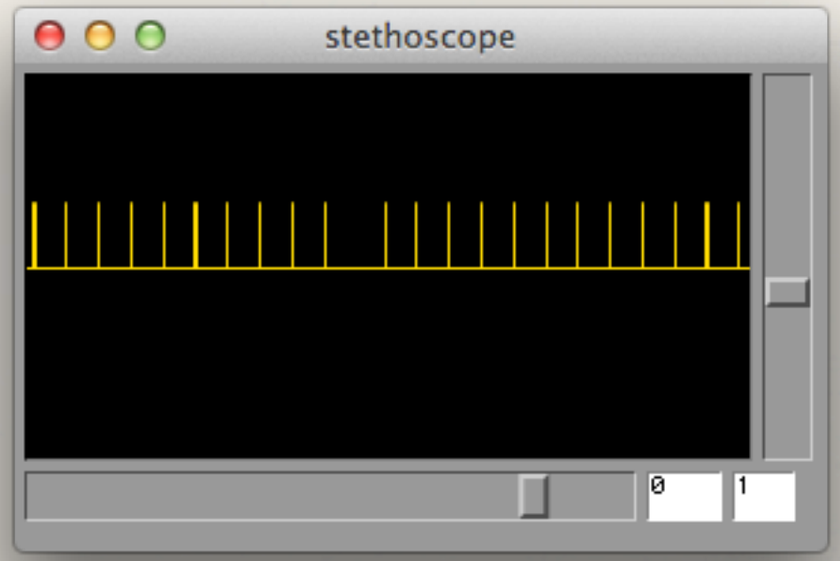
Mit Ramp

```
{
   i = Impulse.ar(10);
   r = TRand.ar(440, 880, i);
   r = Ramp.ar(r, Line.kr(0, 0.3, 5));
   SinOsc.ar(r + 440, 0.0, 0.1);
}.play;
```

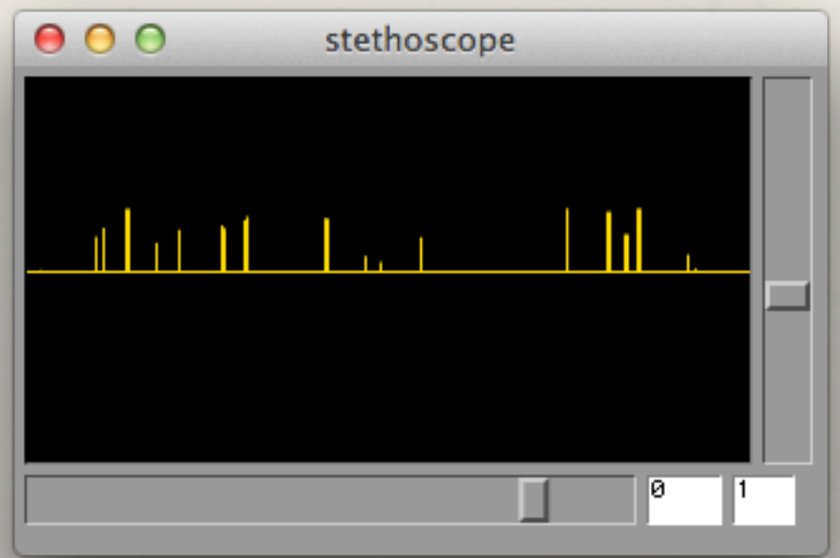
glätter und glätter

# Zwischen Regelmäßigkeit und Unregelmäßigkeit

- Impulse.ar




- Dust.ar



# TRand mit Dust

- TRand mit Dust

```
{
   d = Dust.ar(10);
   r = TRand.ar(440, 880, d);
   SinOsc.ar(r + 440, 0.0, 0.1);
}.play;
```

# TRand und TIRand

- TRand

```
{
   i = Impulse.ar(10);
   r = TRand.ar(0, 1, i) * 440;
   SinOsc.ar(440 + r, 0.0, 0.1);
}.play;
```

TIRand

Integer = Ganze Zahl

- TIRand

```
{
   i = Impulse.ar(10);
   r = TIRand.ar(0, 1, i) * 440;
   SinOsc.ar(r + 440, 0.0, 0.1);
}.play;
```

generiert entweder 0 oder 1

# Interpolation

```
{
   i = Impulse.ar(20);
   r = LinXFade2.ar(TRand.ar(0, 1, i), TIRand.ar(0, 1, i), MouseX.kr(-1,1)) * 440;

   SinOsc.ar(440 + r, 0.0, 0.1);
}.play;
```

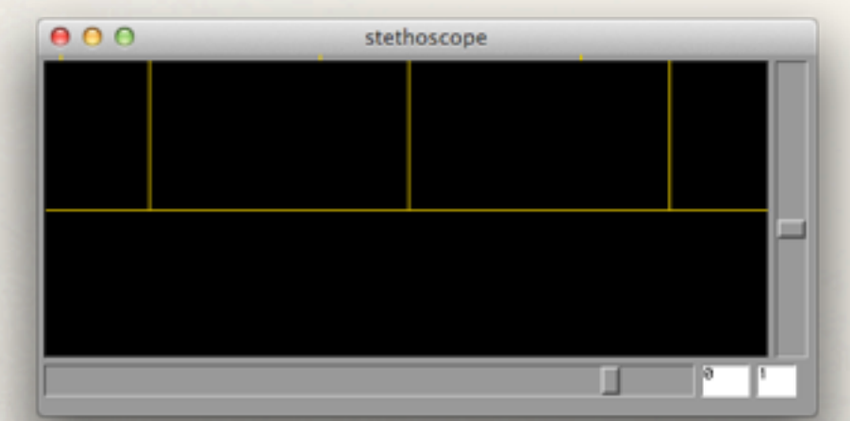
# Impulse und Decay

```
{
   i = Impulse.ar(3) * 440;
   SinOsc.ar(440 + i, 0.0, 0.1);
}.play;
```

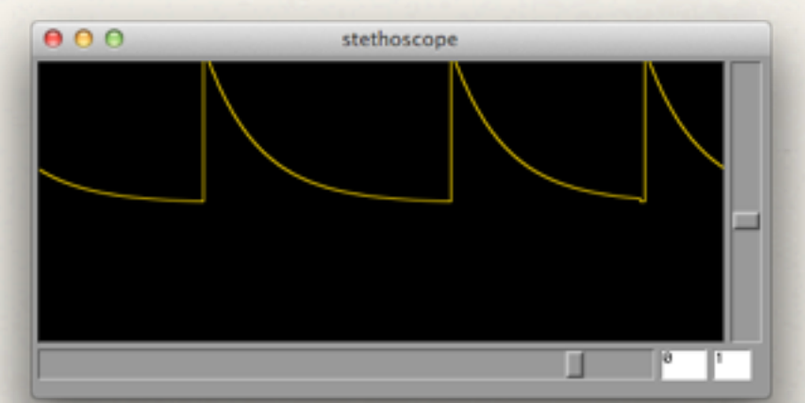
Impulse ist nur 1 Sample.
Man kann nicht hören.

```
{
   i = Impulse.ar(3) * 440;
   d = Decay.ar(i, 0.1);
   SinOsc.ar(440 + d, 0.0, 0.1);
}.play;
```

Impulse mit Decay.
Man kan schnell Glissandi (nach Unten) hören




```
{
   i = Impulse.ar(30);
}.scope;
```




```
{
   i = Impulse.ar(30);
   d = Decay.ar(i, 0.05);
}.scope;
```

# Dust und Decay

- Dust und Decay

```
{
   i = Dust.ar(10) * 440;
   d = Decay.ar(i, 0.1);
   SinOsc.ar(440 + d, 0.0, 0.1);
}.play;
```

- Decay Time auch randomisiert

```
{
   i = Dust.ar(10) * 440;
   d = Decay.ar(i, TRand.ar(0.01, 0.3, i));
   SinOsc.ar(440 + d, 0.0, 0.1);
}.play;
```

# Impulse mit CoinGate

- CoinGate.ar (prob, trig);

Wahlscheilichkeit  Trigger

```
{
   i = Impulse.ar(10) * 440;
   d = Decay.ar(i, 0.07);
   SinOsc.ar(440 + d, 0, 0.1);
}.play
```

```
{
   i = Impulse.kr(10);
   c = CoinGate.kr(0.5 , i);
   d = Decay.kr(c, 0.07);
   SinOsc.ar(440 + (d * 440), 0, 0.1);
}.play
```

```
{
   i = Impulse.kr(10);
   c = CoinGate.kr(MouseX.kr(0, 1) , i);
   d = Decay.kr(c, 0.07);
   SinOsc.ar(440 + (d * 440), 0, 0.1);
}.play
```

# Der programmierbare LFO

Demand Ugen ... Der programmierbare LFO

```
{
  i = Impulse.kr(10);
  d = Dseq([60, 62 ,64 ,65, 67].midicps , inf);
  f = Demand.kr(i,0,d);
  Saw.ar(f, 0.1);
}.play
```

Wenn Demand.kr den ersten Impuls bekommt, generiert Demand.kr 60.midicps

zweite Impuls ... 62.midicps
dritte Impuls... 64.midicps
usw.

# Drand.kr / Dxrand.kr

```
{
  i = Impulse.kr(10);
  d = Drand([60, 62 ,64 ,65, 67].midicps, inf);
  f = Demand.kr(i,0,d);
  Saw.ar(f, 0.1);
}.play
```

Dxrand generiert alle Elemente nur einmal

```
{
  i = Impulse.kr(10);
  d = Dxrand([60, 62 ,64 ,65, 67].midicps, inf);
  f = Demand.kr(i,0,d);
  Saw.ar(f, 0.1);
}.play
```

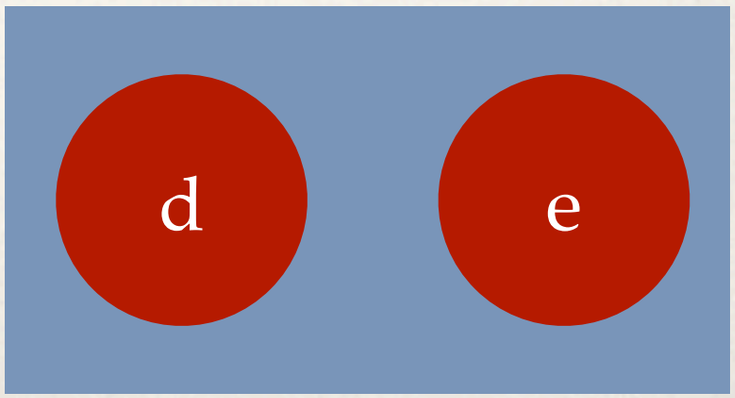
# Dseq in Dseq

```
{
  i = Impulse.kr(10);
  d = Dseq([60, 72 ,67 ,68].midicps, 4);
  f = Demand.kr(i,0,d);
  Saw.ar(f, 0.1);
}.play
```

Wiederholung

```
{
  i = Impulse.kr(10);
  d = Dseq([60, 72 ,67 ,68].midicps, 4);
  e = Dseq([61, 71 ,67 ,66].midicps, 4);
  c = Dseq([d,e],4);
  f = Demand.kr(i,0,c);
  Saw.ar(f, 0.1);
}.play
```

Dseq c



# Drand in Dseq

```
{
  i = Impulse.kr(10);
  d = Dxrand([60, 72 ,67 ,68].midicps, 16);
  e = Dxrand([61, 71 ,67 ,66].midicps, 16);
  c = Dseq([d,e],4);
  f = Demand.kr(i,0,c);
  Saw.ar(f, 0.1);
}.play
```

Randomiserte Reihenfolge

# Dseq in Drand

```
{
   i = Impulse.kr(10);
   a = Dseq([60, 72 ,67 ,68].midicps,4);
   b = Dseq([61, 71 ,67 ,66].midicps,4);
   c = Dseq([60, 70 ,63 ,64].midicps,4);
   d = Dxrand([a,b,c], inf);
   f = Demand.kr(i,0,d);
   Saw.ar(f, 0.1);
}.play
```

Dxrand wählt einen Akkord aus.

# Dwhite, Diwhite

```
{
  i = Impulse.kr(10);
  d = Dwhite(440, 880, inf);
  f = Demand.kr(i,0,d);
  Saw.ar(f, 0.1);
}.play
```

generiert eine Zahl
zwischen 440 und 880

wie TRand

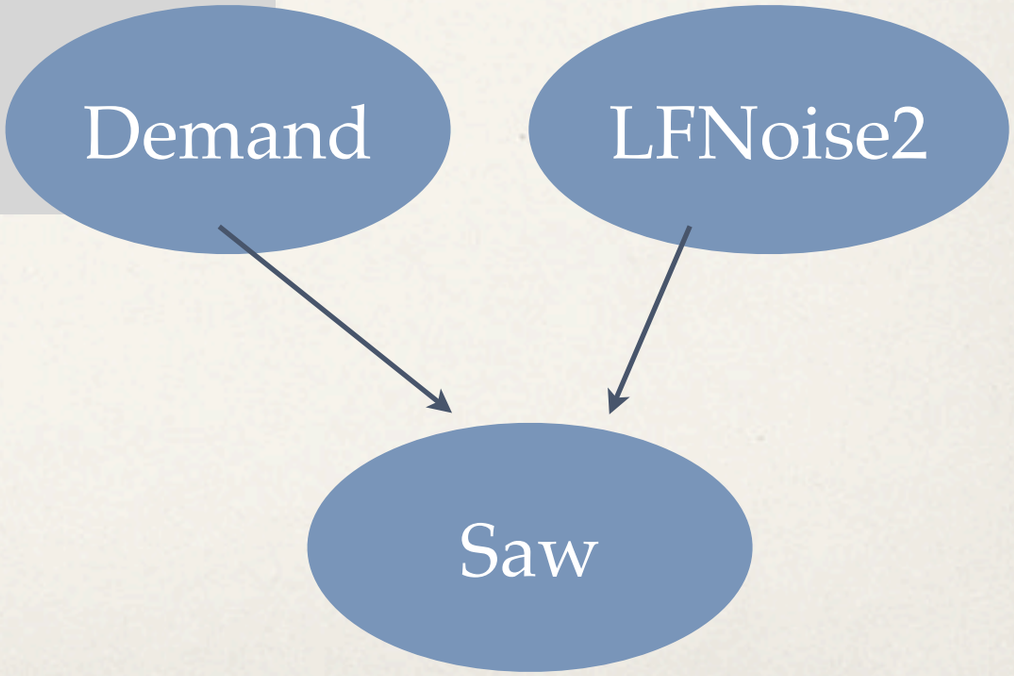
```
{
  i = Impulse.kr(10);
  d = Diwhite(60, 72, inf);
  f = Demand.kr(i,0,d.midicps);
  Saw.ar(f, 0.1);
}.play
```

generiert eine ganze Zahl
zwischen 60 und 72

# Was ist der Unterschied zwischen Routine und Demand Ugen?

```
{
  i = Impulse.kr(10);
  d = Dxrand([60, 72 ,67 ,68].midicps, 16);
  e = Dxrand([61, 71 ,67 ,66].midicps, 16);
  c = Dseq([d,e],inf);
  f = Demand.kr(i,0,c);
  r = LFNoise2.kr(5, 50);
  Saw.ar(f + r, 0.1);
}.play
```

Mischbar



# Chaos-Attraktor

- Was ist Chaos?
  - http://de.wikipedia.org/wiki/Chaosforschung

Die **Chaosforschung** (auch: Chaostheorie) ist ein Teilgebiet der Physik und befasst sich im Wesentlichen mit Ordnungen in dynamischen Systemen, deren Dynamik unter bestimmten Bedingungen empfindlich von den Anfangsbedingungen abhängt, sodass ihr **Verhalten nicht** langfristig **vorhersagbar** ist.

klingt wie Random?

# Chaos-Attraktor

- Ein Beispiel der Chaos-Formeln

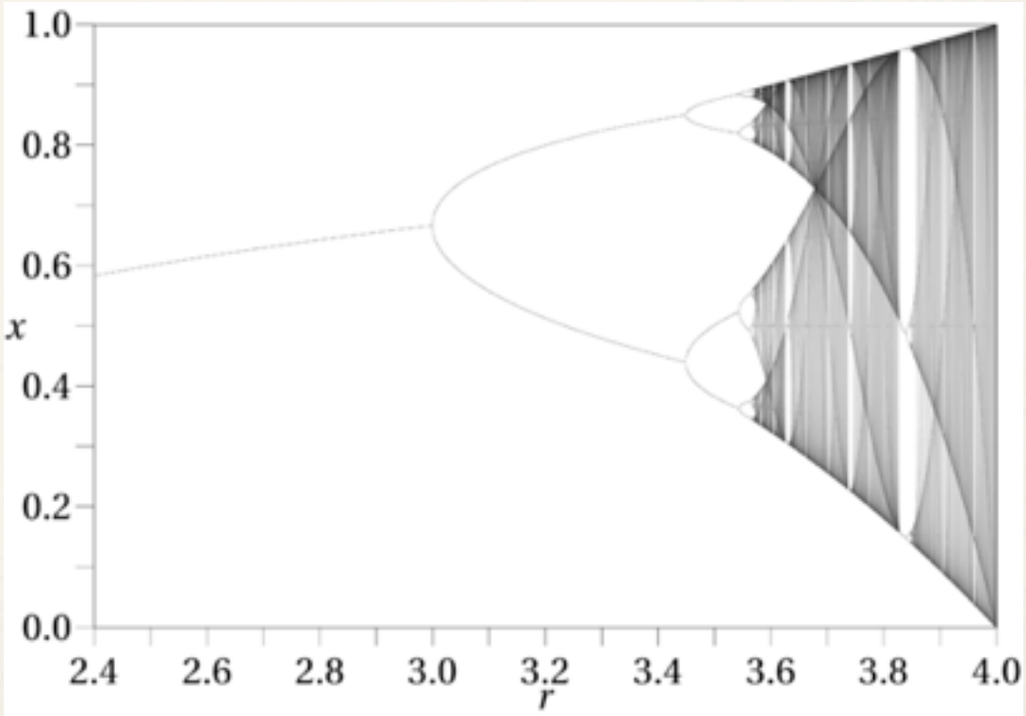
**Logistische Gleichung** http://de.wikipedia.org/wiki/Logistische_Gleichung

$$(1) \qquad x_{n+1} = r x_n (1 - x_n)$$

- Xn muss zwischen 0 und 1
- r muss zwischen 2.4 und 4.0

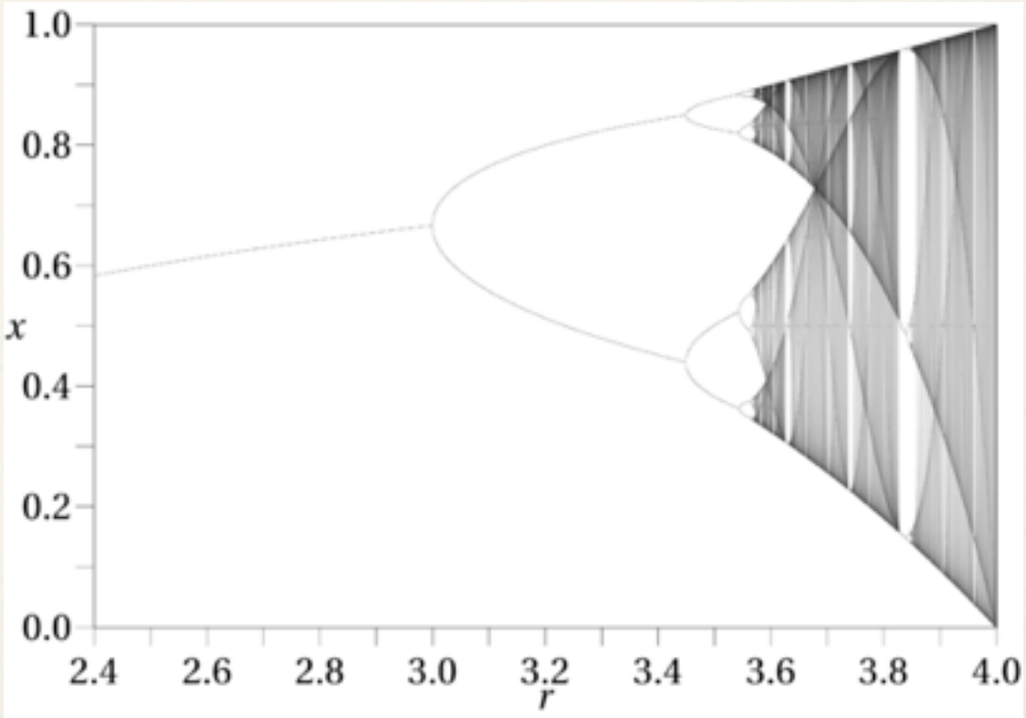
```
SynthDef("Saw",{
    arg freq;
    a = Saw.ar(freq, 0.4);
    Out.ar(0, a);
}).load;

Routine({
    r = 3.6;
    x = 0.5;
    k = Synth("Saw");
    loop{
        k.set("freq", x * 440 + 440);
        r.postln;
        x = r * x * (1-x);
        0.1.wait;
    }
}).play;
```

```
r = 2
r = 2.5
r = 3
r = 3.5
```



# Chaos-Attraktor

```
SynthDef("Saw",{
    arg freq;
    a = Saw.ar(freq, 0.4);
    Out.ar(0, a);
}).load;

(
    w = Window("Chaos", Rect(200, 200, 200, 60));
    a = Slider(w, Rect(20, 10, 160, 30));

    a.action = {
        arg slid;
        r = slid.value * 1.6 + 2.4;
        r.postln;
    };
    w.front;

    Routine({
        r = 3.6;
        x = 0.5;
        k = Synth("Saw");
        loop{
            k.set("freq", x * 440 + 440);
            x = r * x * (1-x);
            0.05.wait;
        }
    }).play;
)
```



# Logistic.ar

```
{
   l = Logistic.ar(MouseX.kr(3, 3.99), 10);
   a = Saw.ar(l * 440 + 440, 0.1);
}.play
```

mit Ramp

```
{
   l = Logistic.ar(MouseX.kr(3, 3.99), 10);
   l = Ramp.ar(l, 0.05);
   a = Saw.ar(l * 440 + 440, 0.1);
}.play
```