Johannes Quint

that word sound

für Klavier und live-generierte Zuspielung

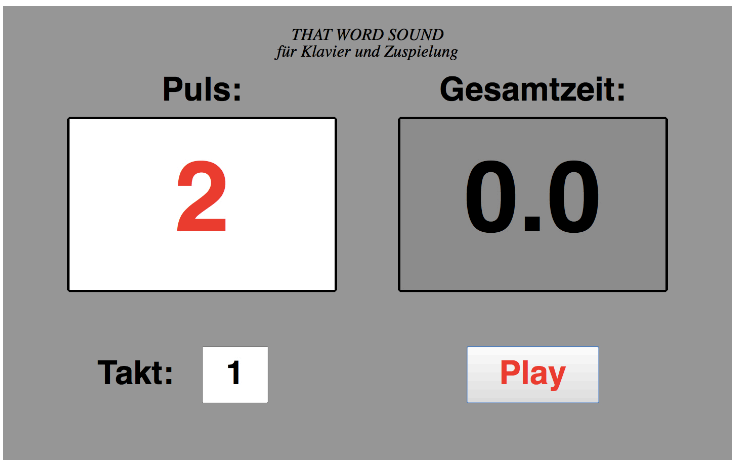
Vorwort

Für 'that word sound' wird ein Flügel und ein Laptop mit SuperCollider (http://supercollider.github.io/) sowie einer Reihe von Samples (werden auf Anfrage gerne zugeschickt: johannes.quint@web.de) benötigt.
Der Laptop muss so postiert werden, dass er vom Pianisten / von der Pianistin einsehbar ist.
Lautsprecher (Stereo) möglichst weit - und erhöht postieren:

```
                         Flügel
Lautsprecher                                  Lautsprecher

                        Pubilkum
```

Die Partitur enthält nur die Klavierstimme und (im Anhang) ein SuperCollider-Patch.
Das Patch wird in SuperCollider evaluiert und generiert dann folgende graphische Oberfläche:



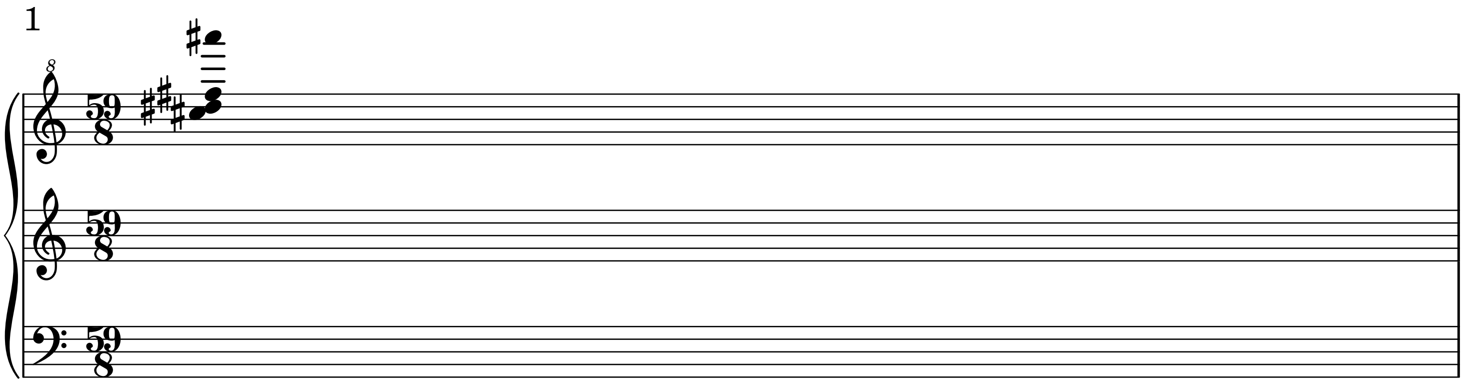
'Play' startet den Klangprozess. 'Puls' zählt den Puls der Takte (Achtel oder Viertel). Die aktuelle Taktzahl wird im Fenster neben 'Takt:' angezeigt. Man kann den Prozess durch 'Stop' stoppen und in das 'Takt'-Feld eine beliebige Zahl eingeben zu der das Patch nach 'Enter' springt.

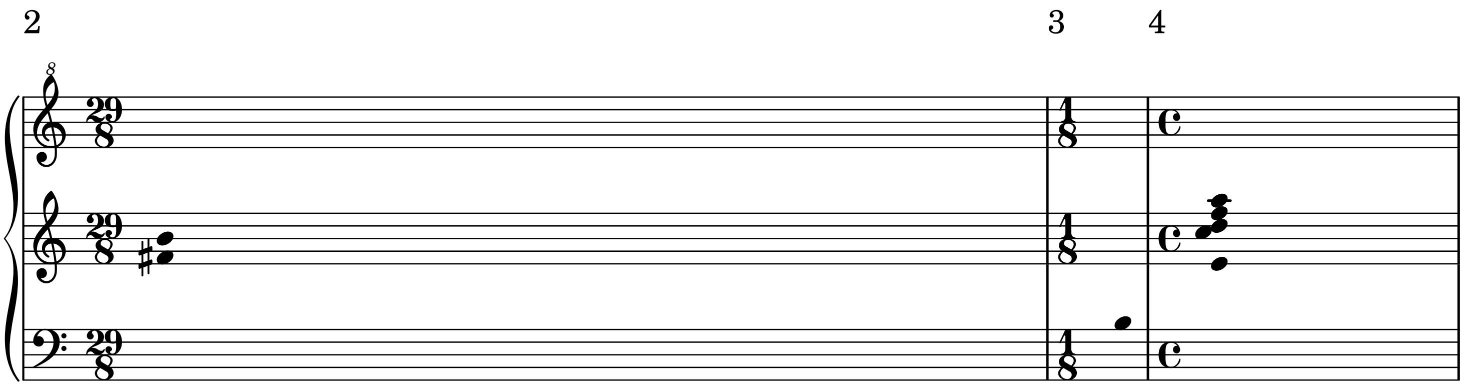
Das Klavier spielt durchgehend piano. Pedal im Prinzip durch das ganze Stück hindurch treten, ad lib können vereinzelte Klänge trocken gespielt werden. Akkorde sollen im Prinzip exakt auf der Takt-1 zusammen angeschlagen werden. Ad lib können vereinzelte Klänge arpeggiert werden. Manche Klänge *können* nicht simultan angeschlagen werden und müssen daher arpeggiert werden.

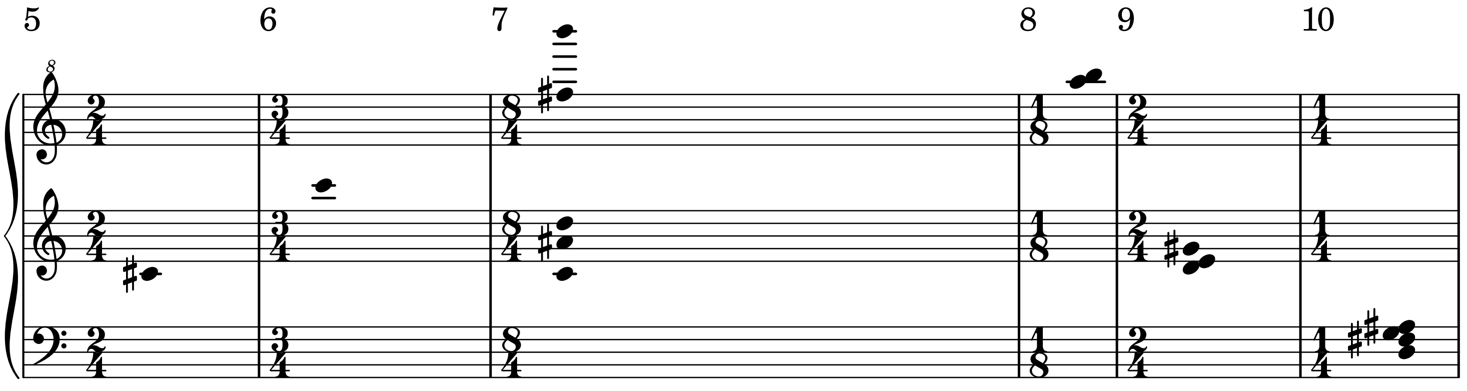
Verwendete Samples:
- Die Stimme von John Cage (Lecture 'mureau': `http://www.ubu.com/sound/cage_mureau.html`)
- Englische Zahlen: Stimme von Amy Gedgaudas: `http://www.freesound.org/people/Corsica_S/`
- Die Stimme von Hans Heinz Stuckenschmidt: Ausschnitt aus: 'Musik im technischen Zeitalter': `https://www.youtube.com/watch?v=9IAWKjvt6A4`

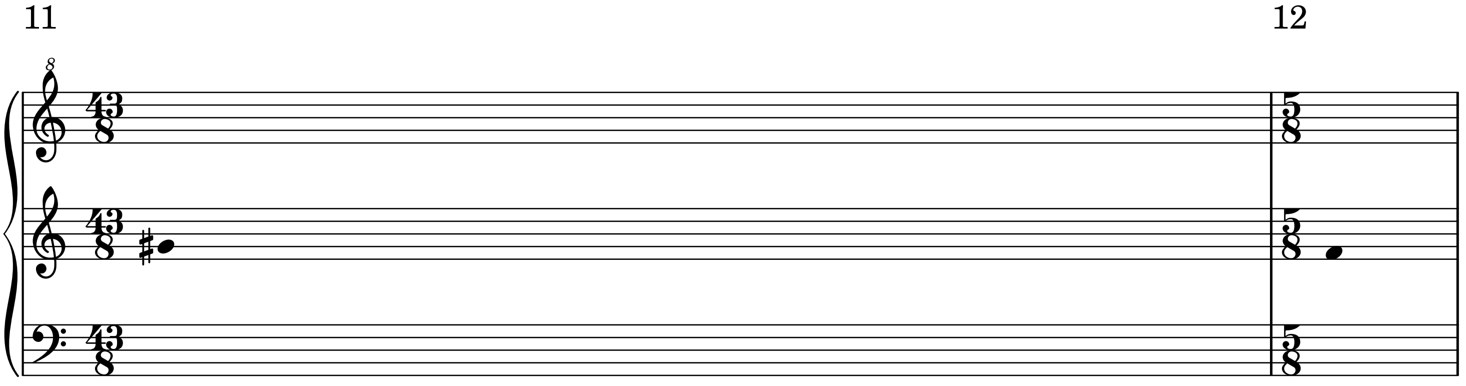
that word sound

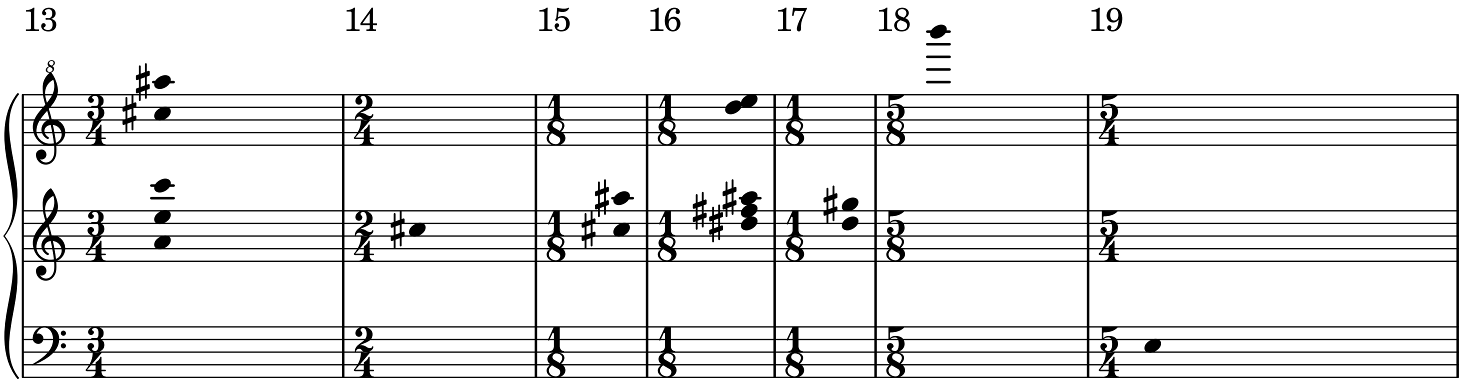
Johannes Quint 2017

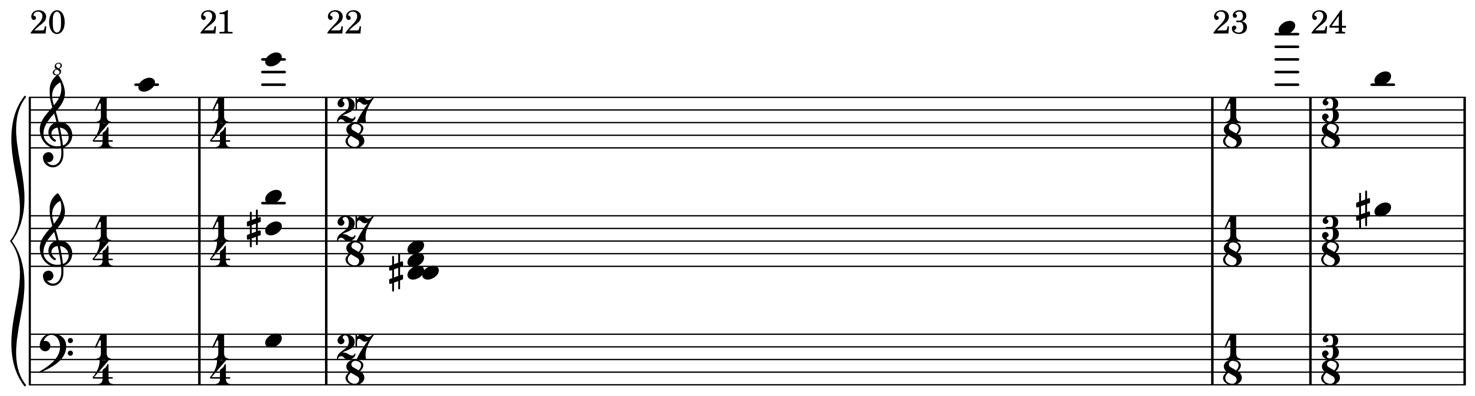
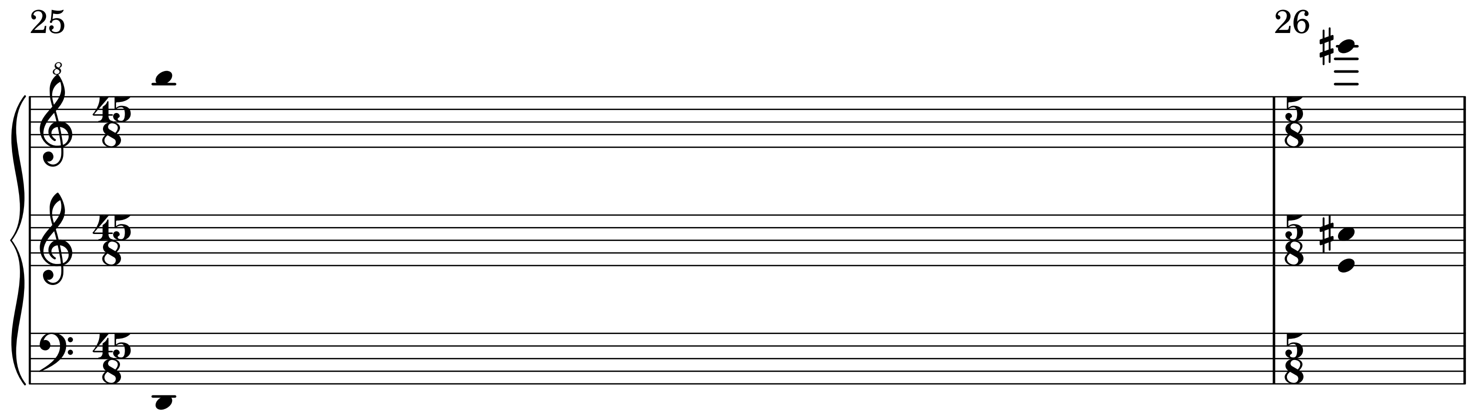
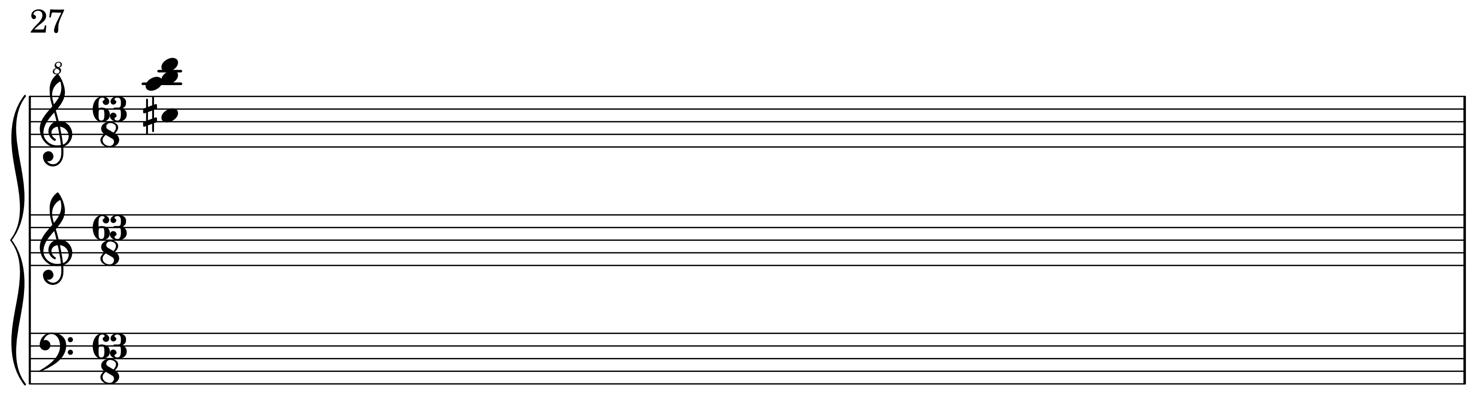
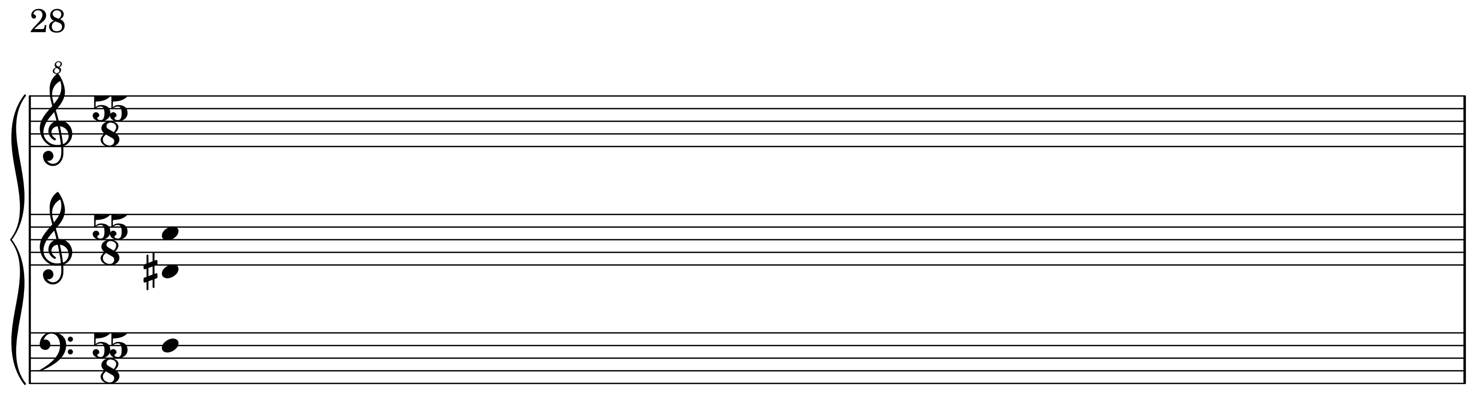
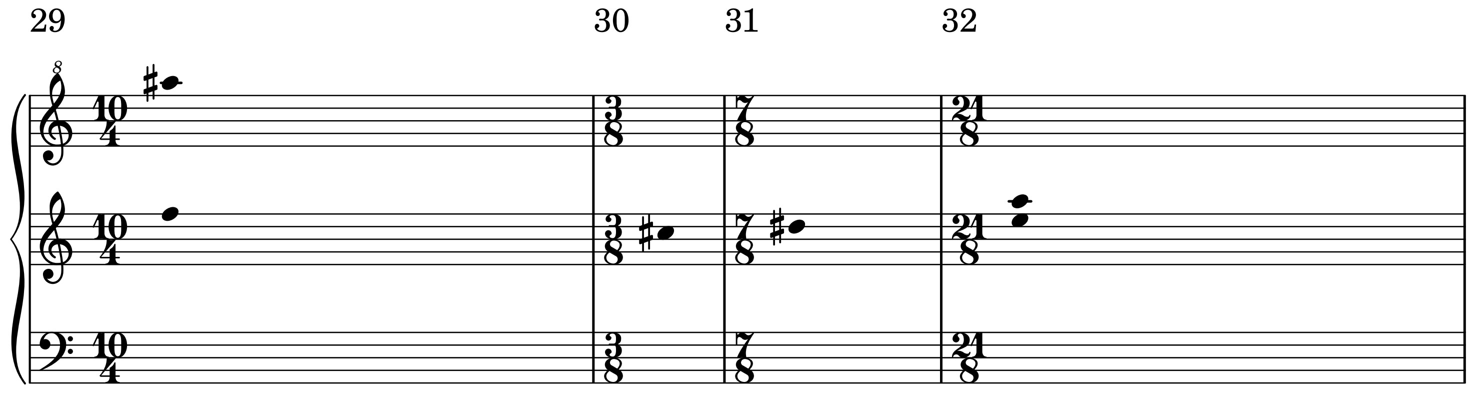


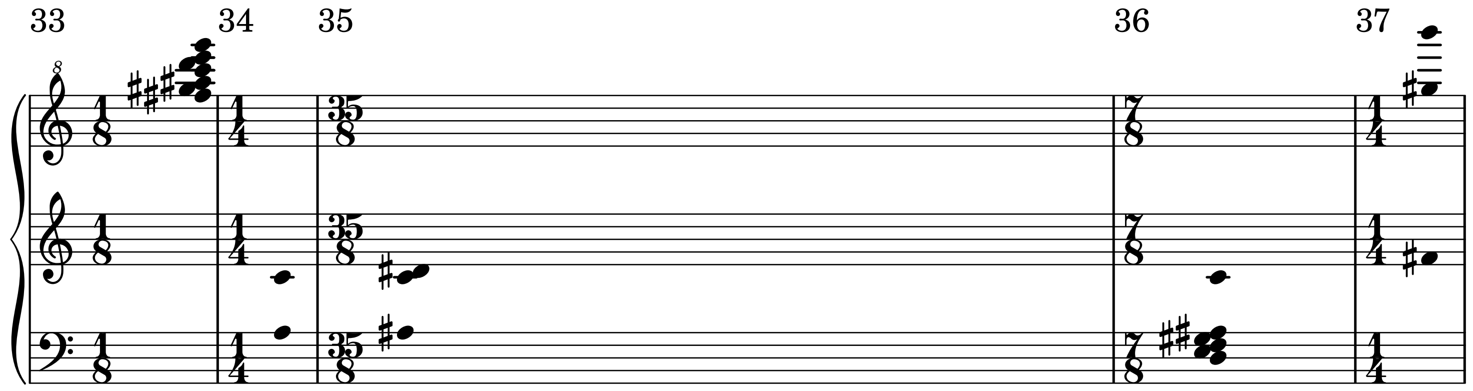


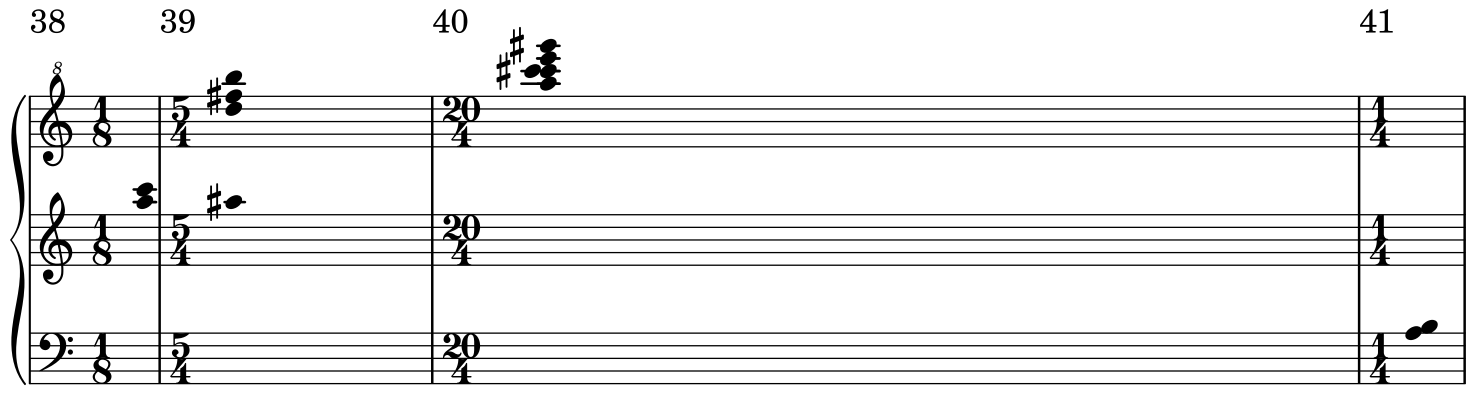


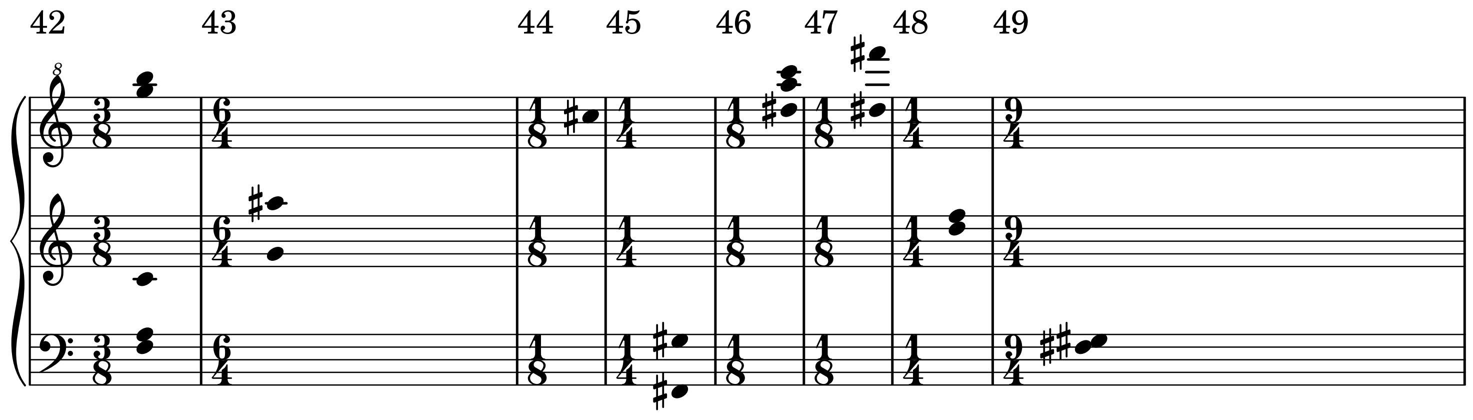


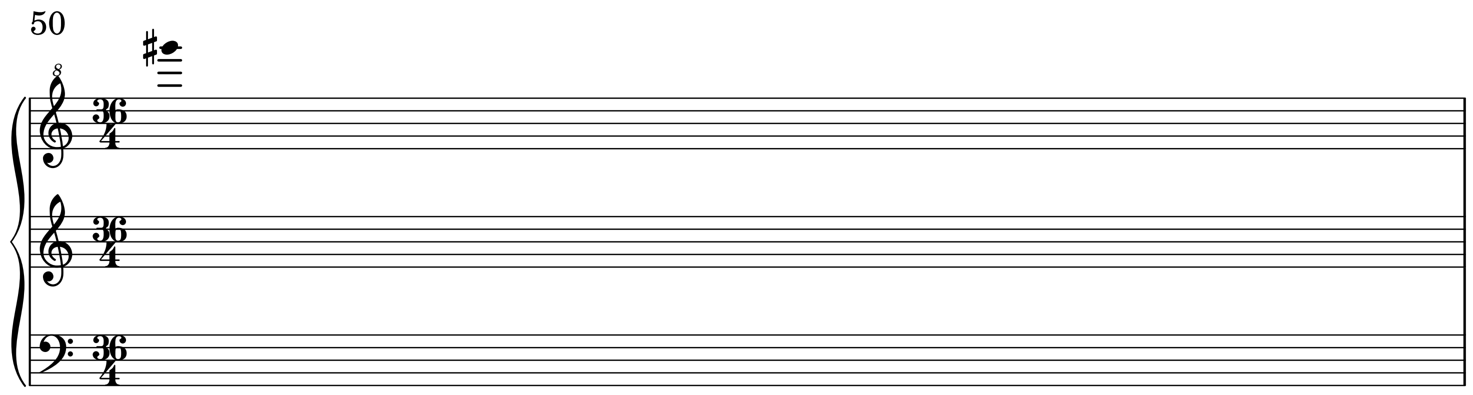


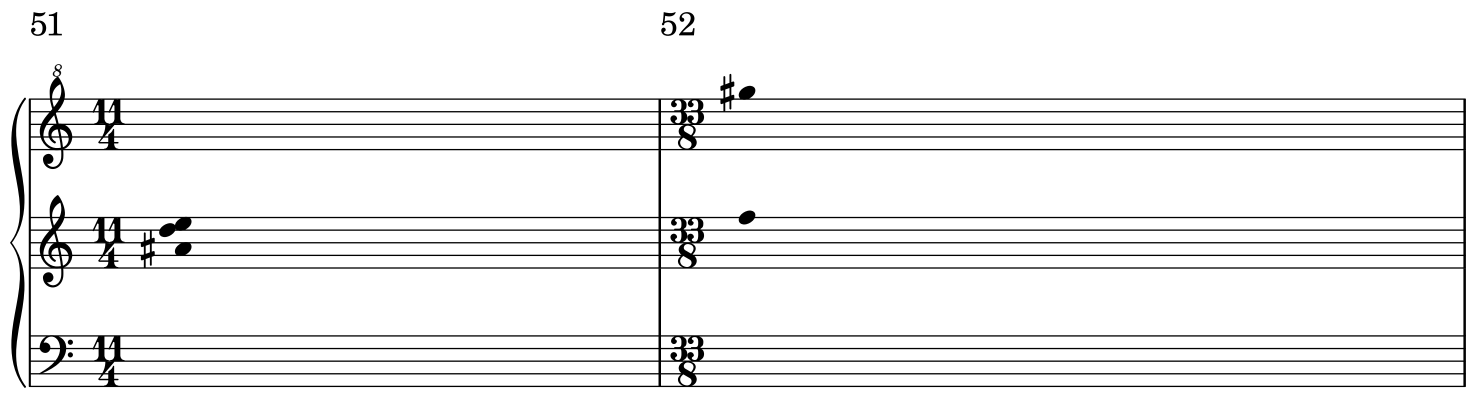

20
21
22
23
24

25
26

27

28

29
30
31
32

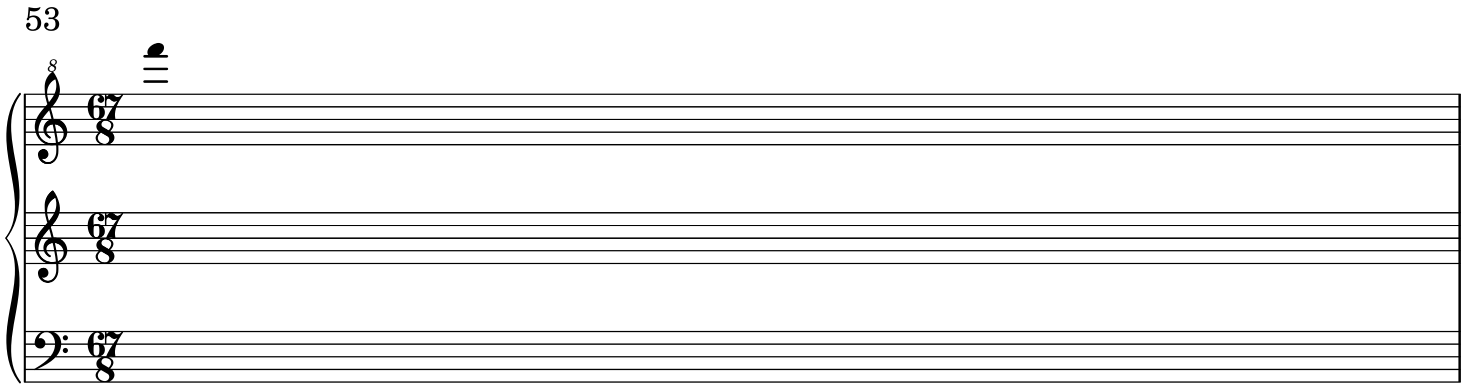

33
34
35
36
37

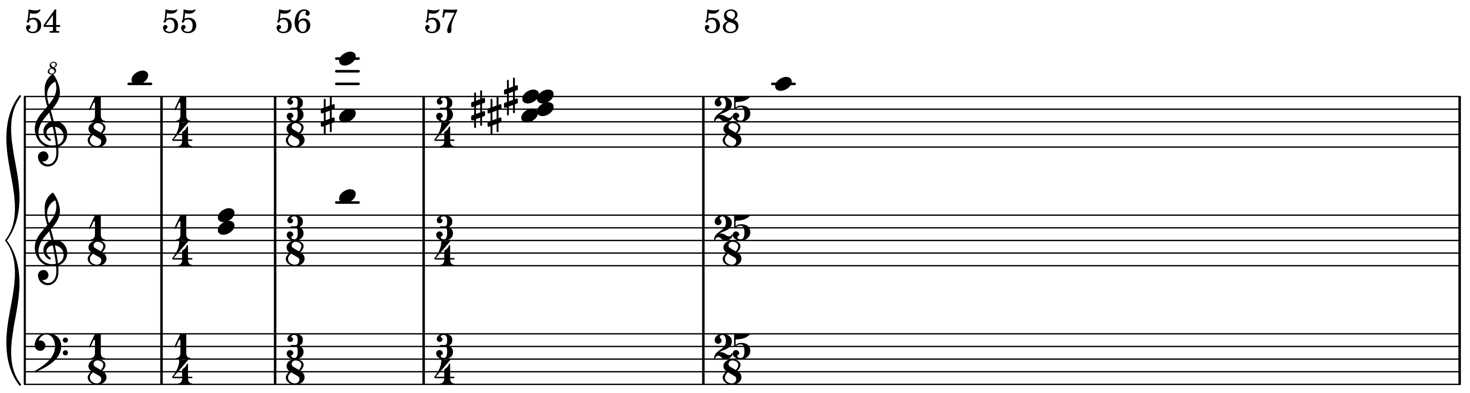

38
39
40
41

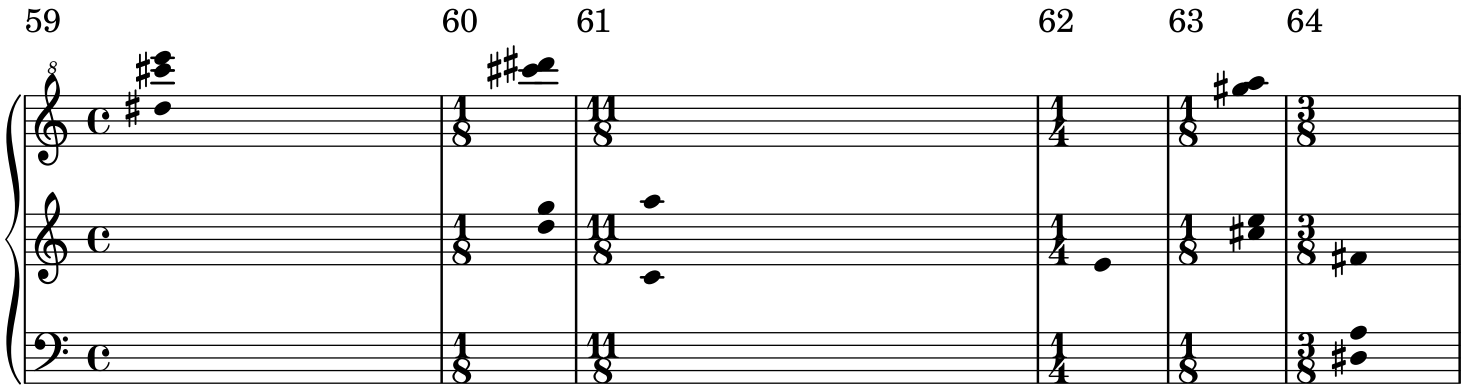

42
43
44
45
46
47
48
49

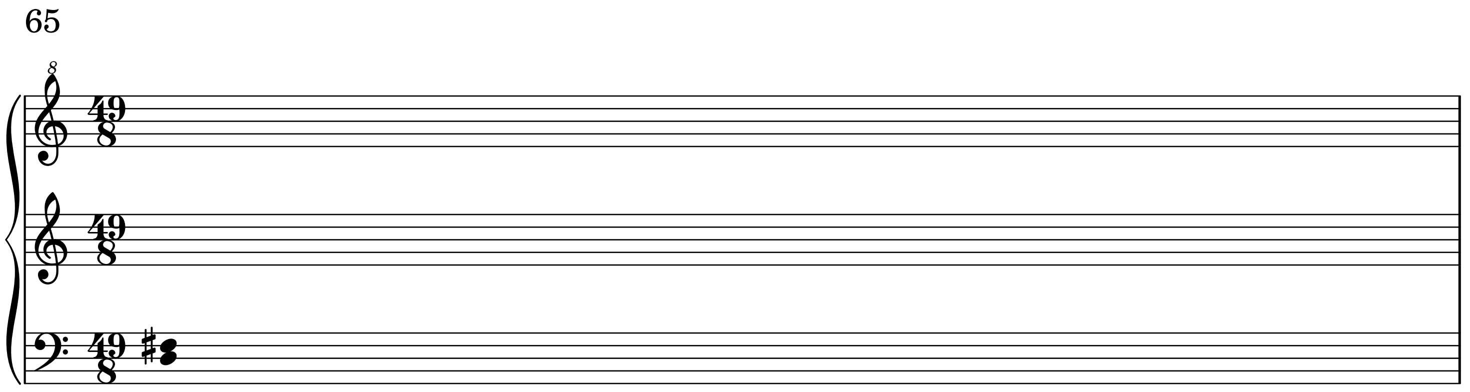

50

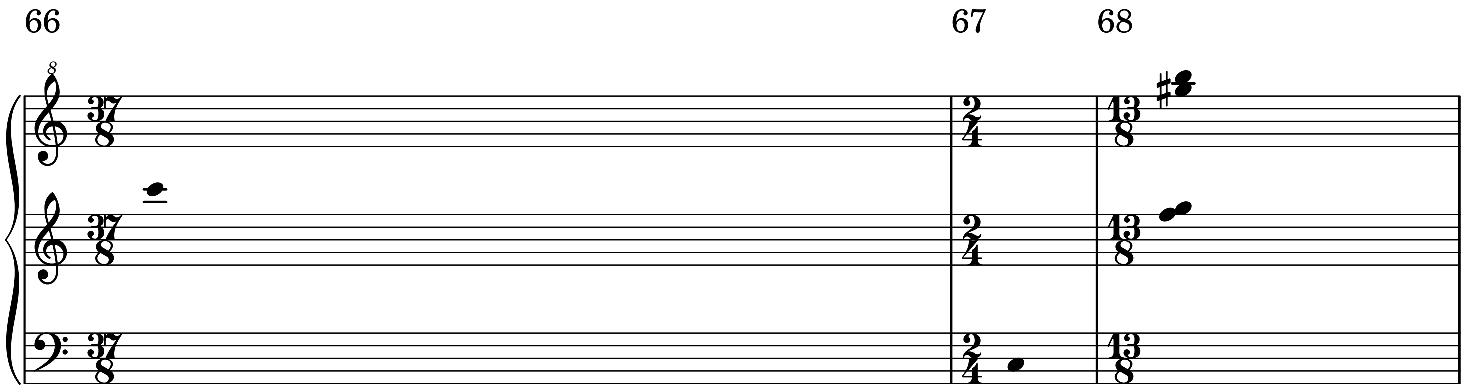

51
52

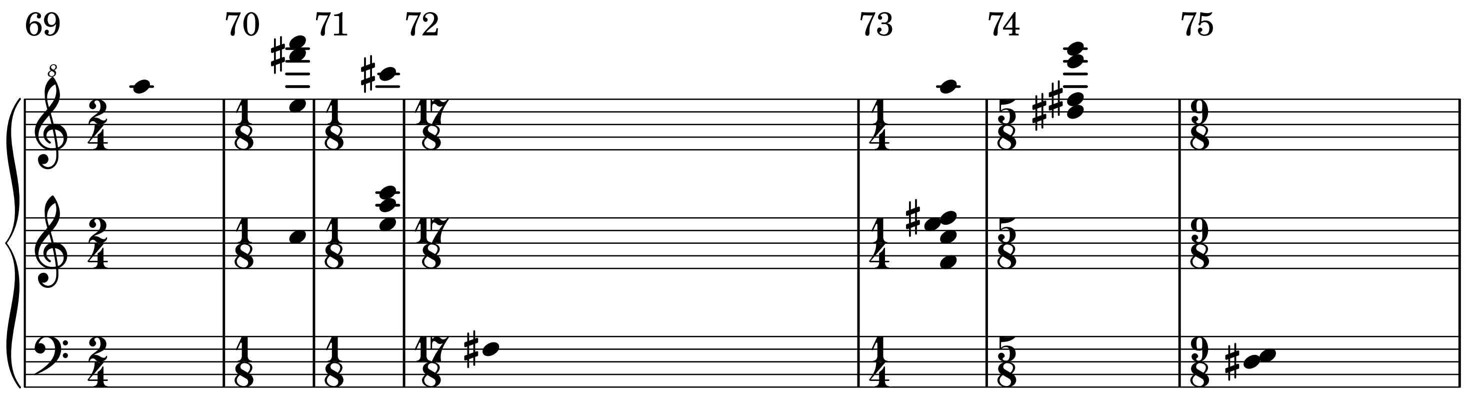
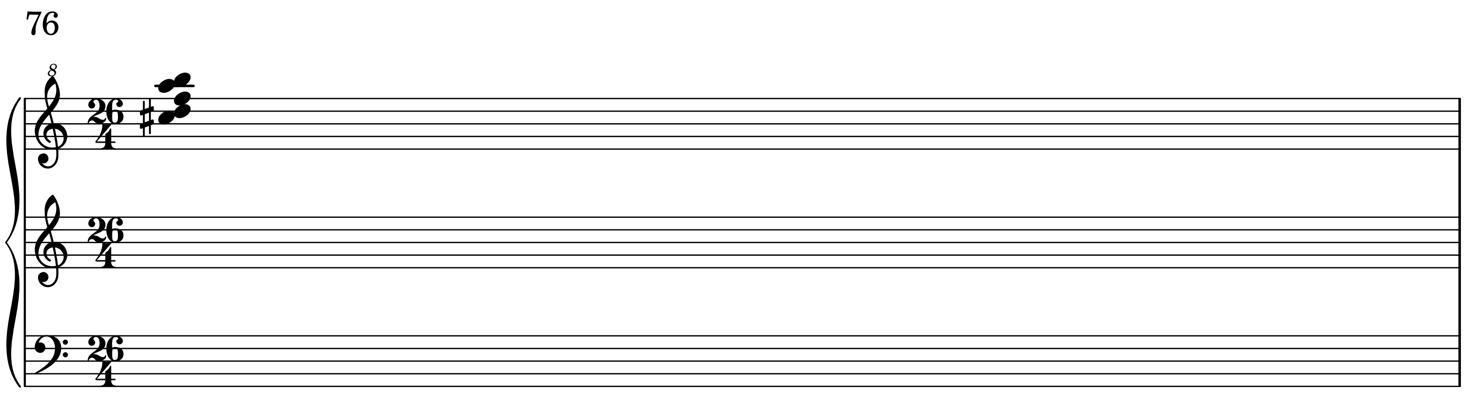
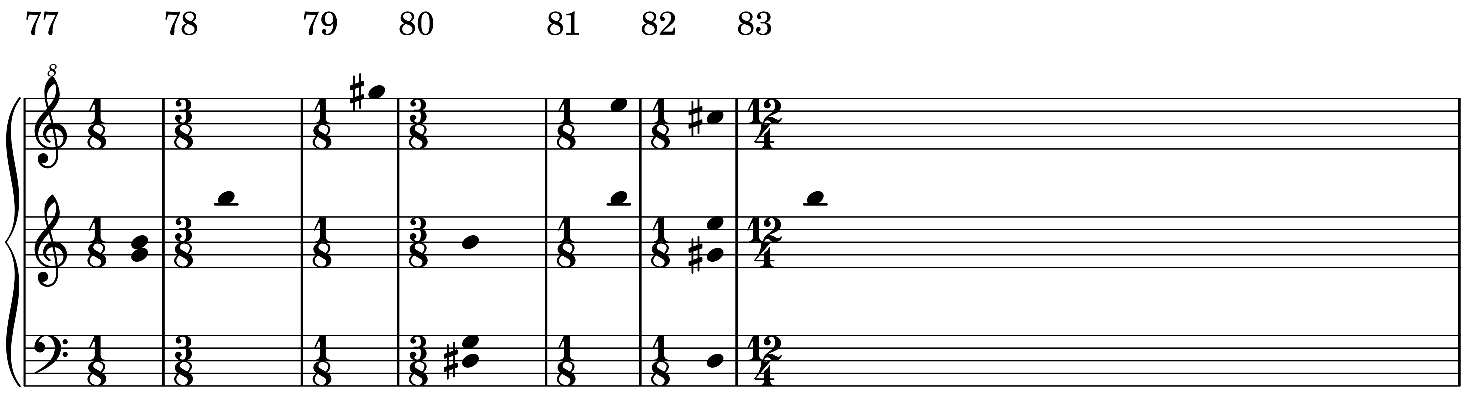
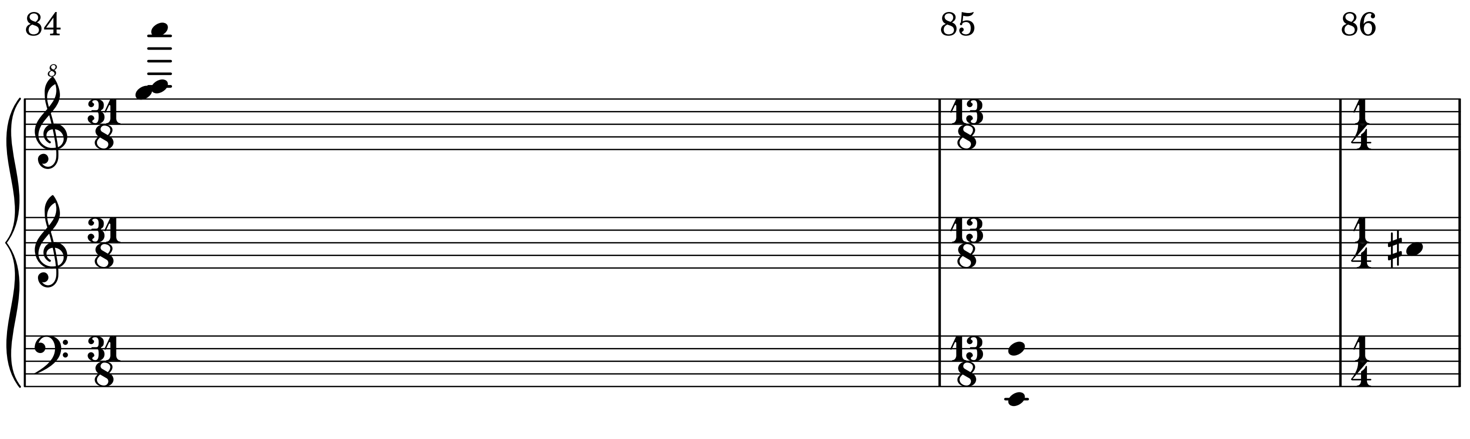
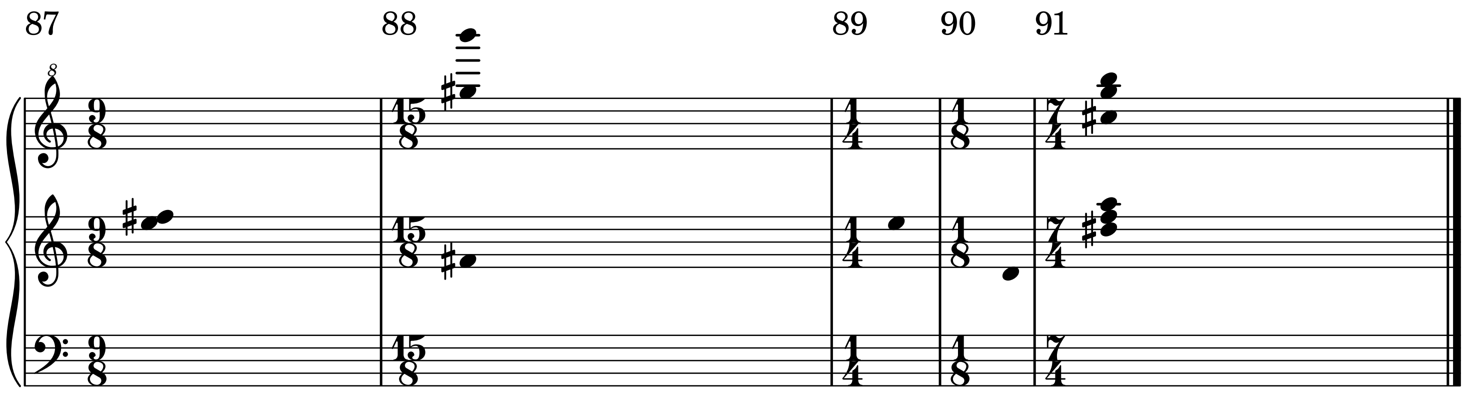

53


54
55
56
57
58


59
60
61
62
63
64


65


66
67
68


69
70
71
72
73
74
75

76

77
78
79
80
81
82
83

84
85
86

87
88
89
90
91

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////// THAT WORD SOUND ///////////////////////////////////////////////
////////////////////////////// SUPERCOLLIDER CODE [funktioniert mit SC 3.8.0] ///////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////////

// 'that word sound' für Klavier und live-generierte Zuspielung
//
// © Johannes Quint 2017
//
// Samples:
//
// John Cages Stimme
// [Lecture 'mureau': http://www.ubu.com/sound/cage_mureau.html]
//
// Zahlen: Amy Gedgaudas
// [http://www.freesound.org/people/Corsica_S/]
//
// Hans Heinz Stuckenschmidts Stimme [Ausschnitt aus: 'Musik im technischen Zeitalter'
// [https://www.youtube.com/watch?v=9IAWKjvt6A4]

s.waitForBoot{

   var
   start,
   concertPitch,
   keys,rhys,instr,
   cage,numbers,title,stuck,
   cageTimes,numbersTimes,pat,titleTimes,stuckTime,
   cageSched,numbersSched,titleSched,stuckSched,
   pnoAmp,arcoAmp,cageAmp,numbersAmp,titleAmp,stuckAmp,
   mainRout,
   bds,left,top,width,height,
   win,rect,titleTxt,startTxt,clockTxt,timerTxt,timeField,timer,but,clockView,time,
   count,
   timerView,timerColorPat;

   // GLOBALS

   concertPitch = 443;
   pnoAmp = 0.3;
   arcoAmp = 0.7;
   cageAmp = 2;
   numbersAmp = 1;
   titleAmp = 2;
   stuckAmp = 2;

   count = 2;

   // SYNTHDEFS
   SynthDef(\perc, {
      arg freq = 440, amp = 0.1, pan = 0, release = 0.5, out = 0;
      var   env,in,sig;
      env = EnvGen.kr(Env.perc(release*2),doneAction: 2);
      in = PinkNoise.ar(amp/4) * Decay2.kr(Impulse.kr(0), 0.01, 0.05);
      sig = Ringz.ar(in,freq,release);
      Out.ar(out, Pan2.ar(sig,pan))}).add;

   SynthDef(\sampler, {
      arg bufnum, rate = 1, pan = 0, amp = 1, start = 0, attack = 0, sustain = 2, release = 0,
      lowCut = 16, highCut = 20000, out = 0,
      curve = 1;
      var sig, filtersig,env;
      sig = PlayBuf.ar(1, bufnum, rate: rate, startPos: 44100 * start);
      filtersig = HPF.ar(LPF.ar(sig, highCut), lowCut);
      env = EnvGen.kr(Env.new([0,amp,amp,0],[attack,sustain, release], curve: [curve,1,curve * -1]),
                 doneAction: 2);
   Out.ar(out, env * Pan2.ar(filtersig, pan))}).add;

SynthDef(\arco, {
   arg
   freq = 440, // Grundfrequenz
   glissInt = 0, // Glissando
   glissDir = 1, // Auf oder Abglissando
   globalAmp = 1,
   amp = 0.1, // Amplitude
   ampVibLo = 1, // tiefer Wert des Amplitudenvibratos (relativ zu globAmp)
   ampVibFqA = 0.25, // Anfangs-/Endfrequenz des Amplitudenvibratos
   ampVibFqB = 1, // Änderungsfrequenz des Amplitudenvibratos
   ampVibAtt = 0.5, // Zeit der ersten Änderung (Anteil der Gesamtdauer -> IMMER <= 1!)
   ampVibCurve = 2, // Kurve der Änderung des Amplitudenvibratos
   attack = 0.1,sustain = 2, release = 0.1, // Gesamthüllkurve
   vib = 0, vibFq = 6, // Frequenzvibrato
   soft = 0.5, // Teiltönigkeit: je grösser soft, desto grundtöniger
   curve = 1, pan = 0; // Kurve der globalen Hüllkurve
   var gliss,dur,snd,envs,globAmp,ampVibEnv,globalAmpSin,globalEnv;
   globAmp = freq.linlin(36.midicps,84.midicps,1,0.5) * globalAmp;
   gliss = (glissDir * (XLine.kr(1,(glissInt+1),attack+sustain+release) -1)).midiratio;
   dur = attack + sustain + release;
   ampVibEnv = EnvGen.ar(
      Env(
         [ampVibFqA,ampVibFqB,ampVibFqA],
         [ampVibAtt*dur,(1-ampVibAtt)*dur],
         [ampVibCurve,ampVibCurve.neg]));
   globalAmpSin =
   SinOsc.ar(ampVibEnv,1.5pi).range(ampVibLo,globAmp);
   globalEnv = EnvGen.kr(Env.linen(attack,sustain,release,amp,[curve,1,curve.neg]),doneAction: 2);
   globalEnv = globalEnv * globalAmpSin;
   envs = 4.collect{ |i|
      var att,ampl,sus;
      sus = attack + sustain + release;
      att = sus / (20 - i);
      ampl = amp ** (i * soft + 1);
      EnvGen.ar(Env([0,ampl,ampl,0],[att,sus - att - att,att],[curve,1,curve.neg]))};
   snd = Mix(
      4.collect{
         |i|
         envs[i] * SinOsc.ar(freq * (i + 1) * SinOsc.kr(vibFq,1.5pi,0.001 * vib,1) * gliss, 0, globAmp)});
   Out.ar(0,Pan2.ar(snd,pan) * globalEnv)}).add;

SynthDef(\pizz1, {
   arg freq = 440, attack = 0.04, release = 2, curve = -4, amp = 0.1, pan = 0;
   var env = EnvGen.kr(Env.perc(attack,release,1,curve),doneAction: 2);
   Out.ar(0, Pan2.ar(SinOsc.ar(freq, 0, amp) * env, pan))}).add;

SynthDef(\pizz2, {
   arg out=0, freq=440, pan=0, percSustain=0.5, amp=0.3, sinAmpFac = 0.1,
   attack = 0.01, sustain = 0, release = 1, curve = 2, hiCut = 5000;
   var pluck, period, string, sin, env,res;
   env = EnvGen.ar(Env.new([0,1,1,0],[attack,sustain,release],[curve,curve.neg]),doneAction: 2);
   sin = SinOsc.ar(freq,0,amp*sinAmpFac);
   pluck = PinkNoise.ar(Decay.kr(Impulse.kr(0.005), 0.05));
   period = freq.reciprocal;
   string = CombL.ar(pluck, period, period, percSustain*6);
   string = LeakDC.ar(LPF.ar(Pan2.ar(string, pan), hiCut)) * amp;
   res = Mix([sin,string]);
   res = env * res;
   res = Pan2.ar(res,pan);
   Out.ar(out,res)
}).add;

SynthDef(\glas,{
   arg freq = 440, amp = 0.7, release = 0.8, curve = -10, pan = 0;
   var env,parts,amps,sig;
   env = EnvGen.ar(Env.perc(0.001,release,amp,curve),doneAction: 2);
   parts = [1, 2.7, 5.2, 8.4, 12.2];
   amps = [0.1, 0.3, 0.3, 0.2, 0.1];
```

```
      sig = Mix(4.collect{arg i; SinOsc.ar(freq*parts[i],0,amps[i])});
      Out.ar(0,Pan2.ar(sig*env,pan))}).add;

   // DAT

   keys = [
[95.0,86.86,90.02,92.69,85.04,88.51,91.41,105.88],[65.51,68.41,70.88],[43.04,58.51,49.41,51.88],
[73.86,77.02,79.69,72.04,63.51,66.41,80.88],[82.41,60.88],[73.69,78.04,93.51,84.41,86.88],
[60.0,63.86,67.02,69.69,74.04,89.51,92.41,106.88],[77.04,92.51,95.41,97.88],
[61.02,63.69,68.04,71.51,62.41,64.88],[53.69,58.04,49.51,52.41,54.88],[67.88],[64.88],
[38.0,53.86,69.02,83.69,76.04,79.51,94.41,84.88],[73.41,99.88],[71.69,64.04,67.51,82.41,72.88],
[75.02,77.69,82.04,85.51,88.41,102.88],[73.02,75.69,80.04,83.51,74.41,88.88],[107.41,85.88],
[51.69,56.04,71.51,86.41,88.88],[92.88],[100.02,54.69,83.04,74.51,89.41,91.88],
[70.86,62.02,64.69,69.04,60.51,63.41,65.88],[93.41,107.88],[80.41,94.88],[95.41,37.88],
[64.04,103.51,94.41,72.88],[84.69,89.04,92.51,95.41,97.88],[52.86,56.02,58.69,63.04,66.51,69.41,71.88],
[76.51,91.41,93.88],[73.41,75.88],[74.88],[75.51,78.41,80.88],
[92.0,95.86,87.02,89.69,94.04,97.51,100.41,102.88],[57.41,59.88],[57.51,60.41,62.88],
[57.86,49.02,51.69,56.04,59.51,50.41,52.88],[65.51,92.41,106.88],[81.41,83.88],[81.69,86.04,89.51,92.41,94.88],
[93.02,95.69,100.04,103.51,106.41,96.88],[56.51,59.41,61.88],[53.0,56.86,60.02,74.69,91.04,94.51,97.41,99.88],
[64.51,67.41,81.88],[84.88],[41.51,56.41,58.88],[87.04,90.51,93.41,95.88],[87.41,101.88],[74.41,76.88],
[55.69,48.04,51.51,54.41,56.88],[103.88],[70.04,73.51,76.41,102.88],[77.41,91.88],[100.88],[80.41,94.88],
[74.41,76.88],[82.51,85.41,99.88],[82.86,86.02,88.69,93.04,84.51,87.41,89.88],[93.41,107.88],
[86.69,91.04,106.51,97.41,99.88],[79.0,82.86,74.02,88.69,93.04,96.51,99.41,101.88],[60.04,63.51,78.41,80.88],
[61.41,63.88],[89.86,93.02,95.69,76.04,91.51,82.41,72.88],[57.04,48.51,51.41,65.88],[50.04,53.51,56.41,58.88],
[83.88],[48.41,50.88],[78.69,95.04,74.51,77.41,91.88],[92.51,95.41,97.88],[72.04,87.51,102.41,104.88],
[65.86,81.02,83.69,76.04,103.51,106.41,96.88],[53.88],[65.02,67.69,72.04,75.51,78.41,92.88],
[87.02,89.69,94.04,85.51,100.41,102.88],[52.02,42.69,47.04,50.51,53.41,43.88],
[94.02,84.69,89.04,92.51,95.41,85.88],[66.69,71.04,74.51,89.41,103.88],[83.04,86.51,101.41,103.88],[91.88],
[50.69,55.04,70.51,73.41,75.88],[82.51,73.41,87.88],[93.86,85.02,75.69,68.04,95.51,50.41,76.88],[82.88],
[90.51,93.41,107.88],[37.02,39.69,44.04,47.51,50.41,52.88],[69.51,72.41,74.88],[75.51,78.41,80.88],
[65.51,92.41,106.88],[67.04,70.51,73.41,75.88],[65.04,68.51,71.41,61.88],
[77.0,80.86,72.02,74.69,91.04,94.51,85.41,99.88]];

   rhys = [
2,29.5,14.5,0.5,4.0,2.0,3.0,8.0,0.5,2.0,1.0,21.5,2.5,3.0,2.0,0.5,0.5,0.5,2.5,5.0,1.0,1.0,13.5,0.5,1.5,22.5,2.5,
31.5,27.5,10.0,1.5,3.5,10.5,0.5,1.0,17.5,3.5,1.0,0.5,5.0,20.0,1.0,1.5,6.0,0.5,1.0,0.5,0.5,1.0,9.0,36.0,11.0,16.
5,33.5,0.5,1.0,1.5,3.0,12.5,4.0,0.5,5.5,1.0,0.5,1.5,24.5,18.5,2.0,6.5,2.0,0.5,0.5,8.5,1.0,2.5,4.5,26.0,0.5,1.5,
0.5,1.5,0.5,0.5,12.0,15.5,6.5,1.0,4.5,7.5,1.0,0.5,7.0];

   instr = [
[1,0,0,1,0,1,1,0],[0,1,0],[1,0,1,1],[0,0,1,0,0,1,0],[1,0],[1,1,1,0,1],[0,1,1,0,0,0,1,0],[1,0,0,1],
[1,0,0,1,0,1],[0,0,0,1,0],[0],[0],[1,1,0,0,0,1,0,0],[0,1],[1,1,1,0,0],[0,0,0,0,0,1],[1,1,0,1,0,1],[0,1],
[0,1,1,1,1],[0],[0,0,0,0,1,1],[1,0,0,0,1,0,1],[1,0],[0,0],[0,0],[0,0,1,0],[0,1,0,0,0],[0,1,1,0,1,1,0],[0,1,0],
[0,1],[0],[0,1,0],[0,0,1,0,0,0,0,0],[0,0],[0,0,0],[0,1,0,0,0,0,0],[0,0,0],[0,0],[0,0,0,1,0],[0,0,0,0,1,0],
[0,0,1],[0,0,0,1,0,0,1,1],[1,0,0],[0],[0,0,1],[0,1,0,0],[0,0],[0,0],[0,1,1,0,1],[0],[0,0,0,1],[0,0],[0],[1,0],
[0,0],[0,0,0],[1,1,0,1,0,0,0],[0,1],[0,1,1,0,0],[0,1,0,1,1,0,0,1],[0,1,1,0],[1,0],[1,0,1,0,0,1,0],[0,1,0,0],
[0,0,1,1],[0],[0,1],[0,0,1,0,0],[0,1,1],[0,0,0,0],[1,0,0,0,1,1,0],[0],[0,1,0,0,0,0],[0,0,1,1,0,0],
[0,1,1,0,1,1],[1,0,0,0,0,0],[0,0,1,1,1],[0,1,1,1],[0],[0,0,0,1,1],[0,1,0],[1,0,0,0,1,0,1],[0],[0,0,0],
[1,0,1,1,1,0],[0,1,1],[0,0,1],[0,0,0],[1,1,1,0],[1,1,1,0],[0,0,1,0,0,0,0,1]];

   // SAMPLES

   cage = PathName("samples/cage/".resolveRelative).entries.collect{
            |p,n| Buffer.read(s,p.fullPath,bufnum: 200+n)};
   numbers = PathName("samples/numbers/".resolveRelative).entries.collect{
            |p,nb| Buffer.read(s,p.fullPath,bufnum: 300+nb)};
   title = Buffer.read(s,"samples/that_word_sound.wav".resolveRelative);
   stuck = Buffer.read(s,"samples/stuck.wav".resolveRelative);

   // START

   start = 1; // 1-based!

   // CAGE TIMES

   cageTimes = [(rhys.sum / (cage.size+1))];
   (rhys.sum-cageTimes[0]).segs(cage.size,1.2).scramble.do{
      |d|
      cageTimes = cageTimes ++ (cageTimes.last + d)};
   cageTimes = cageTimes[.. (cageTimes.size-2)];
```

```
// NUMBERS TIMES

numbersTimes = [(rhys.sum / 12)];
(rhys.sum-numbersTimes[0]).segs(numbers.size-1,1.2).scramble.do{
   |d|
   numbersTimes = numbersTimes ++ (numbersTimes.last + d)};
numbersTimes = numbersTimes[.. (numbersTimes.size-2)];
numbersTimes = numbersTimes ++ (rhys[.. (rhys.size - 2)].sum + 2);

// TITLE TIMES

// titleTimes = [0,rhys.sum,rhys.sum+3];
titleTimes = [0,rhys.sum];

// STUCK TIMES
stuckTime = rhys.sum * 4/5;

// CAGE SCHEDULER

cageSched = {arg start = 1;
   var cTimes,shadow,lo,hi;
   if(start > 1){cTimes = cageTimes - (rhys[..start-1].sum)}{cTimes = cageTimes};
   cTimes.do{
      |tme,i|
      if(tme.isPositive)
      {
         SystemClock.sched(tme,{
            var buf,sus,tms;
            buf = cage[i];
            sus = buf.numFrames / buf.sampleRate;
            tms = rrand(2,4);
            Routine{
               tms.do{
                  |index|
                  Synth(\sampler, [\bufnum,buf,\amp,cageAmp/(1.7**index),\pan,[-1,1,0][index%3],
                                   \sustain,sus]);
                  rrand(0.01,0.1).wait
               }
            }.play;
            shadow = cage.choose;
            lo = rrand(100,300);
            hi = rrand(lo,2000);
            Synth(\sampler, [
               \bufnum,shadow,\amp,cageAmp,\pan,[-1,1,0].choose,\sustain,shadow.numFrames/shadow.sampleRate,
            \lowCut,lo,\highCut,hi]);
         })
}}};

// NUMBERS SCHEDULER

numbersSched = {
   arg start = 1;
   var nTimes;
   if(start > 1){nTimes = numbersTimes - (rhys[..start-1].sum)}{nTimes = numbersTimes};
   nTimes.do{
      |tme,i|
      if(tme.isPositive)
      {
         SystemClock.sched(tme,{
            var indices,bufs,sus,pan,amps;
            indices = (i.rand .. i).scramble;
            bufs = indices.collect{|n| numbers[n]};
            pan = bufs.size.collect{rrand(-1,1.0)};
            amps = indices.collect{|in| if(in == i){1}{rrand(0.2,0.4)}};
            amps = amps * numbersAmp;
            Routine{
               bufs.do{
                  |bf,index|
                  sus = bf.numFrames / bf.sampleRate;
```

```
                    // Synth(\sampler, [\bufnum,buf,\amp,numbersAmp/(1.2**index),\sustain,sus,\pan,pn])
                    Synth(\sampler, [\bufnum,bf,\rate,0.97,\amp,amps[index],\sustain,sus,\pan,pan[index]]);
                    0.2.rand.wait
                };
            }.play}
        )
}}};

// TITLE SCHEDULER

titleSched = {arg start = 1;
    var tTimes;
    if(start > 1){tTimes = titleTimes - (rhys[..start-1].sum)}{tTimes = titleTimes};
    tTimes.do{
        |tme,i|
        var buf,amp;
        buf = title;
        amp = titleAmp;
        if(tme.isPositive)
        {
            SystemClock.sched(tme,{
                var sus,pan;
                sus = buf.numFrames / buf.sampleRate;
                pan = [-1,0,1].scramble;
                Routine{
                    pan.do{
                        |pn,index|
                        Synth(\sampler, [\bufnum,buf,\amp,amp/(1.25**index),\sustain,sus,\pan,pn]);
                        rrand(0.01,0.05).wait
                    }
                }.play}
            )

}}};

// STUCK SCHEDULER

stuckSched = {arg start = 1;
    var sTime;
    if(start > 1){sTime = stuckTime - (rhys[..start-1].sum)}{sTime = stuckTime};
    if(sTime.isPositive)
    {SystemClock.sched(sTime,{
        Synth(\sampler,[\bufnum,stuck,\amp,stuckAmp,\pan,-1,\sustain,stuck.numFrames/stuck.sampleRate,
                     \lowCut,300])
})}};

// MAIN ROUTINE

mainRout = {
    arg start = 1;

    var ks,rs,is;
    ks = if(start == 1){keys[(start-1)..]}{keys[start..]};
    rs = if(start == 1){rhys[(start-1)..]}{rhys[start..]};
    is = if(start == 1){instr[(start-1)..]}{instr[start..]};

    Routine{
        ks.do{
            arg chd,i;
            var r,locR;
            (start+i).postln;
            r = rs[i];
            locR = [0,1,2].choose;
            if(chd.isNil.not)
            {chd.do{
                arg k,index;
                Routine{
                    var inst,att,rel,ky;
                    inst = is[i][index];
                    att = r * rrand(1/3,1/2) * (1 - exprand(1/8,1));
                    rel = r - att * (1 - exprand(1/8,1));
                    ky = if(inst == 0){k.round}{k};
                    [0,rrand(0.03,0.1)].choose.wait;
                    [
                        nil,
                        [
                           {Synth(\pizz2,[\freq,ky.midicps,\amp,0.2,\sustain,rrand(0.3,1)])},
                           {Synth(\pizz1,[\freq,ky.midicps,\amp,0.2,\sustain,rrand(0.3,1)])},
                           {
                              var rl = rrand(0.3,1);
                              Synth(\pizz2,[\freq,ky.midicps,\amp,0.1,\sustain,rl]);
                              Synth(\pizz1,[\freq,ky.midicps,\amp,0.25,\sustain,rl])}
                        ].choose
                    ][inst].value;
                    if(0.3.coin)
                    {
                        {
                           0.3.rand.wait;
                           [
                              {Synth(\perc,[\freq,((ky%12)+60).midicps,\amp,0.3,\release,rrand(0.2,0.4)])},
                              {Synth(\glas,[\freq,((ky%12)+[60,72,84].choose).midicps,\amp,rrand(0.2,0.5),
                                 \release,rrand(0.3,1)])}
                           ].choose.value
                        }.fork
                    };
                    if(1.coin)
                    {
                        {Synth(\arco,[\freq,ky.midicps,\attack,att,\sustain,0,\release,rel,
                           \amp,rrand(0.2,0.3) * arcoAmp,
                           \ampVibLo,rrand(0.2,1),
                           \ampVibFqA,rrand(0.125,0.25),
                           \ampVibFqB,rrand(0.5,2),
                           \ampVibAtt,rrand(0.2,0.6),
                           \soft,rrand(1.5,3)])}.value};
                }.play
            }};
            r.wait}}.play;
};

// GUI

bds = Window.screenBounds;
left = bds.left;
top = bds.top;
width = bds.width / 11;
height = bds.height / 8;

win = Window.new.fullScreen;
win.background = Color.grey(0.6);

timerColorPat = Pseq([Color.blue,Color.green(0.5)],inf).asStream;

rect = Rect(width,height*2,width*4,height*3);
timerView = StaticText(win,rect);
timerView.align = \center;
timerView.background = Color.white;
timerView.font = Font("Monaco",width*1.2,true);
timerView.stringColor = Color.red;
timerView.string = count.asString;

rect = Rect(width*6,height*2,width*4,height*3);
clockView = StaticText(win,rect);
clockView.align = \center;
clockView.background = Color.grey(0.55);
clockView.font = Font("Monaco",width*1.2,true);
clockView.string = "0.0";
time = 0;

titleTxt = StaticText(win,Rect(width*3,20,width*5,height));
titleTxt.align = \center;
titleTxt.font = Font("Times",width/4,false,true);
titleTxt.string = "THAT WORD SOUND\nfür Klavier und Zuspielung";

startTxt = StaticText(win,Rect(width,height*6,width*2,height));
startTxt.align = \center;
startTxt.font = Font("Times",width / 2,true);
startTxt.string = "Takt:";

clockTxt = StaticText(win,Rect(width*6,height,width*4,height));
clockTxt.align = \center;
clockTxt.font = Font("Times",width / 2,true);
clockTxt.string = "Gesamtzeit:";

timerTxt = StaticText(win,Rect(width*1,height,width*4,height));
timerTxt.align = \center;
timerTxt.font = Font("Times",width / 2,true);
timerTxt.string = "Puls:";

timeField = NumberBox(win,Rect(width*3,height*6,width,height));
timeField.align = \center;
timeField.font = Font("Monaco",width / 2,true);
timeField.clipLo = 1;
timeField.clipHi = keys.size;

timeField.action = {
   arg val;
   var index;
   index = val.value.asInteger;
   start = index;
   time = if(start == 1){0}{rhys[..(start-1)].sum};
   timerView.stringColor = Color.red;
   clockView.string = time.asString;
   timerView.string = count;
};

// TIMER: SCHLAEGE/TAKT, GESAMTZEIT

timer = {arg start;
   var strt,cnt;
   strt = if(start == 1){0}{start};
   cnt = start;
   Routine{
      rhys[strt..].do{
         |rh,ct|
         {
            timerView.stringColor = timerColorPat.next; timerView.string = "1";
            timeField.string = cnt.asString
         }.defer;
         if( (rh%1) == 0.5)
         {
            Routine{
               (rh*2).asInteger.do{
                  |n|
                  {
                     timerView.string = (n+1).asString;
                     clockView.string = (time.asString ++ if((time % 1) == 0)
                     {".0"}{""})}.defer;
                  0.5.wait;
                  time = time + 0.5;
            }}.play
         }
         {
            Routine{
               rh.asInteger.do{
                  |n|
                  {
                     timerView.string = (n+1).asString;
                     clockView.string = (time.asString ++ if((time % 1) == 0)
                     {".0"}{""})}.defer;
                  1.wait;
```

```
                    time = time + 1;
          }}.play};
          if(start == 1){if(ct > 1){cnt = cnt + 1}}{if(ct > 0){cnt = cnt + 1}};
          rh.wait;
   }}.play};

   but = Button(win,Rect(width*7,height*6,width*2,height));
   but.font = Font("Times",width / 2,true);
   but.states = [["Play",Color.red],["Stop",Color.green(0.5)]];
   but.action = {
      arg val;
      if(val.value == 1)
      {
         Routine{
            (count+1).do{
               |i|
               {timerView.string = (count - i).asString}.defer;
               1.wait}}.play;
         Routine{
            count.wait;
            mainRout.(start);
            cageSched.(start);
            numbersSched.(start);
            titleSched.(start);
            stuckSched.(start);
            timer.(start)
         }.play
      }
      {
         CmdPeriod.run
      }

   };
   but.focus;

   win.drawFunc = {
      Pen.width = 10;
      Pen.addRect(clockView.bounds);
      Pen.stroke;
      Pen.width = 10;
      Pen.addRect(timerView.bounds);
      Pen.stroke;
   };


   win.onClose = {CmdPeriod.run};
   win.front;
}
```