soft-rock-revisited

Claude Heiland-Allen

2012–2013

Contents

1 analyse/cepstrum-analyser.pd

```
#N canvas 0 0 1905 1014 10;
#N canvas 0 0 450 300 \$0-cepstrum 0;
#X obj 116 33 block~ 16384;
#X obj 23 30 inlet~;
#X obj 25 78 rfft~;
#X obj 25 99 *~;
#X obj 54 100 *~;
#X obj 37 148 log~;
#X obj 37 169 rfft~;
#X obj 37 190 *~;
#X obj 63 189 *~;
#X obj 50 214 +~;
#X obj 83 249 tabwrite~ \$0-cepstrum;
#X obj 213 35 inlet;
#X obj 37 127 +~ 1;
#X connect 1 0 2 0;
```

```
#X connect 2 0 3 0;
#X connect 2 0 3 1;
#X connect 2 1 4 0;
#X connect 2 1 4 1;
#X connect 3 0 12 0;
#X connect 4 0 12 0;
#X connect 5 0 6 0;
#X connect 6 0 7 0;
#X connect 6 0 7 1;
#X connect 6 1 8 0;
#X connect 6 1 8 1;
#X connect 7 0 9 0;
#X connect 8 0 9 0;
#X connect 9 0 10 0;
#X connect 11 0 10 0;
#X connect 12 0 5 0;
#X restore 434 600 pd \$0-cepstrum;
#X obj 352 485 table \$0-cepstrum 8192;
#X obj 528 599 delay 500;
#X obj 557 667 s \$0-cepstrum;
#X obj 531 558 delay 1000;
#X obj 606 439 osc~;
#X obj 607 466 s~ \$0-osc1;
#X obj 606 386 +~ 60;
#X obj 605 410 mtof~;
#X obj 805 438 osc~;
#X obj 805 385 +~ 60;
#X obj 804 409 mtof~;
#X obj 807 341 r~ \$0-osc1;
#X obj 606 338 r~ \$0-osc2;
#X obj 806 465 s~ \$0-osc2;
#X obj 607 363 *~;
#X obj 805 362 *~;
#X obj 702 226 f 0;
#X obj 768 227 + 1;
#X obj 702 288 / 8192;
#X obj 702 309 * 72;
#X obj 702 367 sig~;
#X obj 702 247 moses 8191.5;
#X obj 530 167 metro 2000;
#X obj 529 64 loadbang;
#X msg 629 139 \; pd dsp 1;
#X obj 529 85 t b b;
#X obj 528 620 t b b;
#X obj 527 727 #import (8192 8192 1) b;
#X msg 557 646 normalize 255;
#X obj 527 700 tabdump \$0-cepstrum;
#X obj 527 765 #out;
#X msg 728 711 open soft-rock-iii.png \, autoclose;
#X obj 905 293 print;
#X msg 886 217 \; pd quit;
#X connect 2 0 27 0;
#X connect 4 0 0 1;
#X connect 4 0 2 0;
#X connect 5 0 6 0;
#X connect 5 0 0 0;
#X connect 7 0 8 0;
```

```
#X connect 8 0 5 0;
#X connect 9 0 14 0;
#X connect 10 0 11 0;
#X connect 11 0 9 0;
#X connect 12 0 16 0;
#X connect 13 0 15 0;
#X connect 15 0 7 0;
#X connect 16 0 10 0;
#X connect 17 0 18 0;
#X connect 17 0 22 0;
#X connect 18 0 17 1;
#X connect 19 0 20 0;
#X connect 20 0 21 0;
#X connect 21 0 15 1;
#X connect 21 0 16 1;
#X connect 22 0 19 0;
#X connect 22 0 33 0;
#X connect 22 1 34 0;
#X connect 23 0 17 0;
#X connect 23 0 4 0;
#X connect 24 0 26 0;
#X connect 26 0 23 0;
#X connect 26 1 25 0;
#X connect 26 1 32 0;
#X connect 27 0 30 0;
#X connect 27 1 29 0;
#X connect 28 0 31 0;
#X connect 29 0 3 0;
#X connect 30 0 28 0;
#X connect 32 0 31 0;
```

2 analyse/collector.pd_lua

```
local C = pd.Class:new():register("collector")
local CC = { }

function C:initialize(sel, atoms)
  if type(atoms[1]) ~= "string" then
    self:error("expected symbol for first argument")
    return false
  end
  if type(atoms[2]) ~= "number" then
    self:error("expected number for second argument")
    return false
  end
  if type(CC[atoms[1]]) ~= "table" then
    CC[atoms[1]] = { }
  end
  self.cc = CC[atoms[1]]
  self.split = atoms[2]
  self.inlets = 2
  self.outlets = 3
  return true
end

function C:in_2(sel, atoms)
  if (self.split + 1 < #atoms) then
```

```
25      if type(self.cc[sel]) ~= "table" then
          self.cc[sel] = { }
        end
        local t = { }
        t.which  = sel
30      t.input  = { }
        t.output = { }
        local i
        for i = 1,self.split+1 do
          t.input[i] = atoms[i]
35      end
        for i = self.split+1,#atoms do
          t.output[i - self.split] = atoms[i]
        end
        self.cc[sel][#(self.cc[sel])+1] = t
40    end
  end

  function C:in_1_bang()
    local col,t
45  for col,s in pairs(self.cc) do
      self:outlet(3, "symbol", { col })
      for i,t in ipairs(s) do
        self:outlet(2, "list", t.input)
        self:outlet(1, "list", t.output)
50    end
      self:outlet(3, "bang", {})
    end
  end
```

3 analyse/controller.pd

```
#N canvas 0 0 450 300 10;
#X obj 62 170 netsend 1;
#X msg 62 85 connect localhost \$2 \, send connect localhost \$1;
#X obj 62 34 loadbang;
5 #X obj 92 132 r \$1;
#X obj 62 60 pack \$2 \$3;
#X connect 1 0 0 0;
#X connect 2 0 4 0;
#X connect 3 0 0 0;
10 #X connect 4 0 1 0;
```

4 analyse/controllers.pd

```
#N canvas 1258 57 645 421 10;
#X floatatom 46 37 8 0 0 0 - - -;
#X floatatom 107 37 8 0 0 0 - - -;
#X obj 106 75 t b f;
5 #X obj 45 174 s \$0-net;
#N canvas 0 373 450 300 \$0-container 0;
#X restore 387 269 pd \$0-container;
#X msg 409 207 clear;
#X obj 343 12 r batch;
10 #X obj 389 72 unpack f f;
#X obj 388 115 until;
```

```
#X obj 455 131 + 1;
#X obj 388 134 f 5000;
#X obj 388 155 pack f \$0;
#X obj 389 93 max 1;
#X obj 371 52 t b a b;
#X msg 346 207 loadbang;
#X obj 388 241 s pd-\$0-container;
#X obj 43 197 netreceive 4000 1;
#X msg 388 176 obj 10 10 controller \$2-net 4000 \$1;
#X obj 43 240 print netreceive:4000;
#X obj 104 221 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X obj 43 219 spigot;
#N canvas 0 0 726 579 \$0-video 0;
#X obj 19 26 inlet;
#X obj 155 29 inlet;
#X obj 215 29 inlet;
#X msg 62 75 dimen 640 480 \, create \, 1;
#X msg 19 96 0 \, destroy;
#X obj 110 99 gemhead;
#X obj 59 148 GEMglEnd;
#X obj 327 321 select chaotic pitched;
#X obj 141 238 unpack f f;
#X obj 271 297 route bang;
#X obj 127 169 t b a;
#X obj 149 145 GEMglBegin;
#X obj 268 102 GLdefine;
#X obj 302 24 loadbang;
#X msg 272 78 symbol GL_POINTS;
#X obj 71 238 unpack f f;
#X obj 142 261 -;
#X obj 142 301 expr 4 * tanh($f1);
#X obj 156 549 GEMglVertex2fv;
#X obj 156 507 gemlist;
#X obj 156 481 t b a;
#X obj 19 47 sel 0 1;
#X obj 257 143 GEMglPointSize 3;
#X obj 110 120 t a a a a a;
#X obj 372 135 GEMglEnable;
#X obj 392 103 GLdefine;
#X msg 389 74 symbol GL_POINT_SMOOTH;
#X obj 71 217 collector \$0-collection 2;
#X obj 198 168 unpack f f f f;
#X obj 195 258 -;
#X obj 195 279 /;
#X obj 142 279 /;
#X obj 201 322 expr 4 * tanh($f1);
#X obj 158 401 unpack f f f f;
#X obj 156 528 GEMglColor3fv;
#X msg 328 340 0.5;
#X msg 386 343 1;
#X obj 158 434 pack f f;
#X obj 226 435 / 12;
#X obj 226 456 wrap;
#X obj 159 359 pack f f f;
#X msg 159 380 \$2 \$3 \$1;
#X obj 226 502 hsv2rgb;
```

```
#X obj 19 123 gemwin 1;
#X obj 226 482 pack f f f;
#X connect 0 0 21 0;
#X connect 1 0 27 1;
#X connect 2 0 28 0;
#X connect 3 0 43 0;
#X connect 4 0 43 0;
#X connect 5 0 23 0;
#X connect 7 0 35 0;
#X connect 7 1 36 0;
#X connect 8 0 16 0;
#X connect 8 1 29 0;
#X connect 9 1 7 0;
#X connect 10 0 27 0;
#X connect 10 1 19 1;
#X connect 12 0 11 1;
#X connect 13 0 14 0;
#X connect 13 0 26 0;
#X connect 14 0 12 0;
#X connect 15 0 40 0;
#X connect 16 0 31 0;
#X connect 17 0 40 1;
#X connect 19 0 34 0;
#X connect 20 0 19 0;
#X connect 20 1 18 1;
#X connect 21 0 4 0;
#X connect 21 1 3 0;
#X connect 23 0 6 0;
#X connect 23 1 10 0;
#X connect 23 2 11 0;
#X connect 23 3 22 0;
#X connect 23 4 24 0;
#X connect 25 0 24 1;
#X connect 26 0 25 0;
#X connect 27 0 15 0;
#X connect 27 1 8 0;
#X connect 27 2 9 0;
#X connect 28 0 16 1;
#X connect 28 1 31 1;
#X connect 28 2 29 1;
#X connect 28 3 30 1;
#X connect 29 0 30 0;
#X connect 30 0 32 0;
#X connect 31 0 17 0;
#X connect 32 0 40 2;
#X connect 33 0 37 0;
#X connect 33 1 37 1;
#X connect 33 2 38 0;
#X connect 34 0 18 0;
#X connect 35 0 44 1;
#X connect 35 0 44 2;
#X connect 36 0 44 1;
#X connect 36 0 44 2;
#X connect 37 0 20 0;
#X connect 38 0 39 0;
#X connect 39 0 44 0;
#X connect 40 0 41 0;
```

```
#X connect 41 0 33 0;
#X connect 42 0 34 1;
#X connect 44 0 42 0;
#X restore 65 298 pd \$0-video;
#X obj 65 275 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 1
1;
#N canvas 0 0 450 300 \$0-audio 0;
#X obj 24 25 inlet;
#X msg 24 46 \; pd dsp \$1;
#X obj 149 25 inlet;
#X obj 92 115 *~;
#X obj 92 136 +~ 43;
#X obj 92 157 mtof~;
#X obj 92 178 osc~;
#X obj 92 199 s~ \$0-osc-1;
#X obj 172 115 *~;
#X obj 172 136 +~ 43;
#X obj 172 157 mtof~;
#X obj 172 178 osc~;
#X obj 172 199 s~ \$0-osc-2;
#X obj 92 94 r~ \$0-osc-2;
#X obj 172 94 r~ \$0-osc-1;
#X obj 132 227 outlet~;
#X obj 200 225 outlet~;
#X obj 149 46 unpack f f f f;
#X connect 0 0 1 0;
#X connect 2 0 17 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 5 0 6 0;
#X connect 6 0 7 0;
#X connect 6 0 15 0;
#X connect 8 0 9 0;
#X connect 9 0 10 0;
#X connect 10 0 11 0;
#X connect 11 0 12 0;
#X connect 11 0 16 0;
#X connect 13 0 3 0;
#X connect 14 0 8 0;
#X connect 17 0 3 1;
#X connect 17 0 8 1;
#X connect 17 2 9 1;
#X connect 17 2 4 1;
#X restore 217 283 pd \$0-audio;
#X obj 213 257 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 1
1;
#X obj 46 58 / 100;
#X msg 440 28 1 5001;
#N canvas 0 0 450 300 \$0-think-zinc-sink 0;
#X obj 198 55 inlet~;
#X obj 269 56 inlet~;
#X obj 132 146 noin-04-think-zinc-sink;
#X obj 199 175 outlet~;
#X obj 268 175 outlet~;
#X obj 75 124 f 0;
#X obj 107 124 + 1;
#X obj 136 10 inlet;
```

```
#X obj 108 172 outlet;
#X obj 76 99 metro 125;
#X obj 136 31 sel 1 0;
#X msg 166 74 0;
#X obj 136 52 t b b;
#X msg 153 107 2816;
#X msg 138 74 -33;
#X connect 0 0 2 1;
#X connect 1 0 2 2;
#X connect 2 1 3 0;
#X connect 2 2 4 0;
#X connect 5 0 6 0;
#X connect 5 0 2 0;
#X connect 5 0 8 0;
#X connect 6 0 5 1;
#X connect 7 0 10 0;
#X connect 9 0 5 0;
#X connect 10 0 12 0;
#X connect 10 1 11 0;
#X connect 10 1 13 0;
#X connect 11 0 9 0;
#X connect 12 0 9 0;
#X connect 12 1 14 0;
#X connect 13 0 2 0;
#X connect 14 0 5 1;
#X restore 157 310 pd \$0-think-zinc-sink;
#X obj 227 346 dac~;
#X floatatom 157 331 5 0 0 0 - - -;
#X obj 107 54 / 10;
#X obj 158 278 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X obj 45 115 pack f f f f;
#X floatatom 167 37 8 0 0 0 - - -;
#X obj 166 75 t b f;
#X floatatom 227 37 8 0 0 0 - - -;
#X obj 226 75 t b f;
#X obj 227 54 / 10;
#X msg -5 28 4670.22;
#X obj 167 54 / 100;
#X msg 45 143 send icentre \$1 \, send irange \$2 \, send bcentre \$3
\, send brange \$4;
#X obj 22 3 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 345 32 t b a;
#X msg 46 18 5787.5;
#X msg 107 19 0.05;
#X msg 225 22 0.05;
#X msg 167 22 3600;
#X connect 0 0 25 0;
#X connect 1 0 30 0;
#X connect 2 0 32 0;
#X connect 2 1 32 1;
#X connect 5 0 15 0;
#X connect 6 0 42 0;
#X connect 7 0 12 0;
#X connect 7 1 10 1;
#X connect 8 0 10 0;
```

```
#X connect 9 0 10 1;
#X connect 10 0 9 0;
#X connect 10 0 11 0;
#X connect 11 0 17 0;
#X connect 12 0 8 0;
#X connect 13 0 14 0;
#X connect 13 1 7 0;
#X connect 13 2 5 0;
#X connect 14 0 15 0;
#X connect 16 0 20 0;
#X connect 16 0 21 1;
#X connect 17 0 15 0;
#X connect 19 0 20 1;
#X connect 20 0 18 0;
#X connect 22 0 21 0;
#X connect 23 0 27 1;
#X connect 23 1 27 2;
#X connect 24 0 23 0;
#X connect 25 0 32 0;
#X connect 26 0 13 0;
#X connect 27 0 29 0;
#X connect 27 1 28 0;
#X connect 27 2 28 1;
#X connect 30 0 2 0;
#X connect 31 0 27 0;
#X connect 32 0 40 0;
#X connect 32 0 21 2;
#X connect 32 0 23 1;
#X connect 33 0 39 0;
#X connect 34 0 32 0;
#X connect 34 1 32 2;
#X connect 35 0 37 0;
#X connect 36 0 32 0;
#X connect 36 1 32 3;
#X connect 37 0 36 0;
#X connect 38 0 0 0;
#X connect 39 0 34 0;
#X connect 40 0 3 0;
#X connect 41 0 45 0;
#X connect 41 0 46 0;
#X connect 41 0 44 0;
#X connect 41 0 43 0;
#X connect 42 0 41 0;
#X connect 42 1 13 0;
#X connect 43 0 0 0;
#X connect 44 0 1 0;
#X connect 45 0 35 0;
#X connect 46 0 33 0;
```

5 analyse/loader-image.pd

```
#N canvas 0 0 450 300 10;
#X obj 36 22 loadbang;
#X msg 18 211 \; pd dsp 1;
#X obj 36 43 delay 2000;
#N canvas 0 0 450 300 \$0-container 0;
#X restore 58 183 pd \$0-container;
```

```
#X msg 90 120 clear;
#X obj 57 163 s pd-\$0-container;
#X msg 81 212 \; pd dsp 0;
#X obj 135 14 r go;
#X obj 36 81 list append;
#X obj 36 102 t b a b;
#X msg 57 141 obj 10 10 main-image \$1 \$2 \$3 \$4 \$5 \, loadbang
;
#X connect 0 0 1 0;
#X connect 0 0 2 0;
#X connect 2 0 8 0;
#X connect 4 0 5 0;
#X connect 7 0 8 1;
#X connect 8 0 9 0;
#X connect 9 0 1 0;
#X connect 9 1 10 0;
#X connect 9 2 6 0;
#X connect 10 0 5 0;
```

6 analyse/loader.pd

```
#N canvas 0 0 450 300 10;
#X obj 36 22 loadbang;
#X msg 18 211 \; pd dsp 1;
#X obj 36 43 delay 2000;
#N canvas 0 0 450 300 \$0-container 0;
#X restore 58 183 pd \$0-container;
#X msg 90 120 clear;
#X obj 57 163 s pd-\$0-container;
#X obj 135 14 r port;
#X obj 36 81 f 5000;
#X msg 57 141 obj 10 10 main \$1 \, loadbang;
#X obj 36 102 t b f b;
#X msg 81 212 \; pd dsp 0;
#X obj 135 35 t f b;
#X obj 135 56 mod 16;
#X obj 135 77 + 1;
#X obj 137 100 until;
#X msg 137 121 obj 10 10 random 10;
#X connect 0 0 1 0;
#X connect 0 0 2 0;
#X connect 2 0 7 0;
#X connect 4 0 5 0;
#X connect 6 0 7 1;
#X connect 6 0 11 0;
#X connect 7 0 9 0;
#X connect 8 0 5 0;
#X connect 9 0 1 0;
#X connect 9 1 8 0;
#X connect 9 2 10 0;
#X connect 11 0 12 0;
#X connect 11 1 4 0;
#X connect 12 0 13 0;
#X connect 13 0 14 0;
#X connect 14 0 15 0;
#X connect 15 0 5 0;
```

7 analyse/log2rgb.hs

```
import qualified Data.ByteString as BS
import Data.Fixed (mod')
import Data.Word (Word8)
import System.Environment (getArgs)

hsv2rgb :: Double -> Double -> Double -> [Double]
hsv2rgb h 0 v = [v, v, v]
hsv2rgb 1 s v = hsv2rgb 0 s v
hsv2rgb h s v =
  case i of
    0 -> [v, t, p]
    1 -> [q, v, p]
    2 -> [p, v, t]
    3 -> [p, q, v]
    4 -> [t, p, v]
    5 -> [v, p, q]
  where
    i = floor h'
    h' = 6 * (h - fromIntegral (floor h))
    f = h' - fromIntegral i
    p = v * (1 - s)
    q = v * (1 - s * f)
    t = v * (1 - s * (1 - f))

d2b :: Double -> Word8
d2b x = round (255 * x)

go :: [String] -> [Double]
go ["image:", _, _, "pitched", _, _, x, _] = hsv2rgb ((read x / 12) `mod'` 1) 1 ↙
    ↳ 1
go ["image:", _, _, "chaotic", _, _, x, _] = hsv2rgb ((read x / 12) `mod'` 1) ↙
    ↳ 0.5 0.5
go _ = []

main :: IO ()
main = do
  [inFile, outFile] <- getArgs
  BS.writeFile outFile . BS.pack . map d2b . concatMap go . map words . lines ↙
      ↳ =<< readFile inFile
```

8 analyse/main-image.pd

```
#N canvas 0 0 1186 980 10;
#X obj 82 370 osc~;
#X obj 83 397 s~ \$0-osc1;
#X obj 81 341 mtof~;
#X obj 321 369 osc~;
#X obj 320 340 mtof~;
#X obj 323 272 r~ \$0-osc1;
#X obj 82 269 r~ \$0-osc2;
#X obj 322 396 s~ \$0-osc2;
#X obj 101 5 loadbang;
#X obj 32 421 sigmund~ -npts 4096 -hop 4096;
#N canvas 0 0 450 300 \$0-statistics 0;
```

```
#X obj 18 11 inlet;
#X obj 18 108 + 1;
#X obj 82 87 f 0;
#X obj 47 87 + 1;
#X obj 18 87 f 0;
#X obj 17 33 t b f f f;
#X obj 143 92 f 0;
#X obj 108 90 + 1;
#X obj 109 60 *;
#X obj 19 157 swap;
#X obj 19 178 /;
#X obj 19 199 outlet;
#X obj 78 199 outlet;
#X obj 80 157 expr sqrt($f1 * $f3 - $f2 * $f2) / $f1;
#X obj 19 128 t f f;
#X obj 170 17 inlet;
#X obj 170 38 b;
#X obj 170 59 f 0;
#X connect 0 0 5 0;
#X connect 1 0 4 1;
#X connect 1 0 14 0;
#X connect 2 0 3 1;
#X connect 3 0 9 1;
#X connect 3 0 13 1;
#X connect 3 0 2 0;
#X connect 4 0 1 0;
#X connect 5 0 4 0;
#X connect 5 1 3 0;
#X connect 5 2 8 0;
#X connect 5 3 8 1;
#X connect 6 0 7 1;
#X connect 7 0 13 2;
#X connect 7 0 6 0;
#X connect 8 0 7 0;
#X connect 9 0 10 0;
#X connect 9 1 10 1;
#X connect 10 0 11 0;
#X connect 13 0 12 0;
#X connect 14 0 9 0;
#X connect 14 1 13 0;
#X connect 15 0 16 0;
#X connect 16 0 17 0;
#X connect 17 0 7 1;
#X connect 17 0 3 1;
#X connect 17 0 4 1;
#X restore 33 511 pd \$0-statistics;
#X obj 33 481 spigot;
#X obj 101 71 t b b b;
#X msg 106 455 0;
#X msg 58 455 1;
#X obj 36 651 route 0 1;
#X obj 197 715 swap;
#X obj 31 541 expr ($f1 > 0) && !($f2 > 1) \; $f1;
#X obj 35 607 pack 0 f f;
#X obj 37 628 route 0;
#X obj 302 712 snapshot~;
#X obj 192 693 t f b;
```

```
#N canvas 0 0 450 371 \$0-thd 0;
#X obj 21 17 inlet~;
#X obj 138 21 inlet;
#X obj 89 184 czero~;
#X obj 89 205 czero~;
#X obj 91 299 *~;
#X obj 91 320 +~;
#X obj 120 298 *~;
#X obj 89 224 cpole~;
#X obj 89 245 cpole~;
#X obj 139 49 mtof;
#X obj 309 161 * 0.99;
#X obj 141 154 * 0.99;
#X obj 137 206 * -1;
#X obj 136 246 * -1;
#X obj 236 44 loadbang;
#X obj 236 65 expr 4 * asin(1) / 48000;
#X obj 140 114 expr cos( $f2 * $f1) \; sin( $f2 * $f1 );
#X obj 90 341 lop~ 10;
#X obj 90 362 outlet~;
#X obj 214 47 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X connect 0 0 2 0;
#X connect 1 0 9 0;
#X connect 2 0 3 0;
#X connect 2 1 3 1;
#X connect 3 0 7 0;
#X connect 3 1 7 1;
#X connect 4 0 5 0;
#X connect 5 0 17 0;
#X connect 6 0 5 1;
#X connect 7 0 8 0;
#X connect 7 1 8 1;
#X connect 8 0 4 1;
#X connect 8 0 4 0;
#X connect 8 1 6 1;
#X connect 8 1 6 0;
#X connect 9 0 16 0;
#X connect 10 0 7 3;
#X connect 10 0 13 0;
#X connect 11 0 7 2;
#X connect 11 0 8 2;
#X connect 12 0 3 3;
#X connect 13 0 8 3;
#X connect 14 0 15 0;
#X connect 15 0 16 1;
#X connect 16 0 2 2;
#X connect 16 0 3 2;
#X connect 16 0 11 0;
#X connect 16 1 2 3;
#X connect 16 1 10 0;
#X connect 16 1 12 0;
#X connect 17 0 18 0;
#X connect 19 0 15 0;
#X restore 302 687 pd \$0-thd;
#N canvas 0 0 460 628 \$0-spec 0;
#X obj 41 20 inlet~;
```

```
#X obj 194 20 inlet;
#X obj 284 21 inlet;
#X obj 189 366 outlet;
#X obj 253 580 outlet;
#X obj 282 86 block~ 512 4;
#X obj 57 42 tabreceive~ \$0-hann;
#X obj 42 63 *~;
#X obj 43 85 rfft~;
#X obj 42 120 *~;
#X obj 71 119 *~;
#X obj 42 141 +~;
#X obj 42 175 *~ 0.1;
#X obj 94 176 *~ 0.9;
#X obj 93 145 tabreceive~ \$0-spectrum;
#X obj 285 108 table \$0-spectrum 512;
#X obj 79 201 +~;
#X obj 79 222 tabsend~ \$0-spectrum;
#X msg 284 42 const 0;
#X obj 284 63 s \$0-spectrum;
#X msg 213 65 256;
#X obj 213 86 until;
#X obj 213 107 f 0;
#X obj 254 109 + 1;
#X obj 254 130 mod 256;
#X obj 194 41 t b b b;
#X obj 214 166 tabread \$0-spectrum;
#X obj 214 207 +;
#X obj 246 208 f;
#X obj 214 187 t f f;
#X obj 213 128 t f f;
#X obj 326 192 *;
#X obj 326 212 +;
#X obj 358 213 f;
#X obj 189 253 pack f f f;
#X msg 189 274 \$3 \$2;
#X obj 189 295 /;
#X msg 257 71 0;
#X obj 343 166 * 93.75;
#X obj 189 342 ftom;
#X obj 252 533 sqrt;
#X msg 267 232 256;
#X obj 267 253 until;
#X obj 267 274 f 0;
#X obj 308 276 + 1;
#X obj 308 297 mod 256;
#X obj 268 333 tabread \$0-spectrum;
#X obj 270 439 +;
#X obj 304 445 f;
#X obj 268 354 t f f;
#X obj 267 295 t f f;
#X obj 361 447 *;
#X obj 360 468 +;
#X obj 404 466 f;
#X msg 311 238 0;
#X obj 397 333 * 93.75;
#X obj 378 385 -;
#X obj 190 316 t f b b f b;
```

```
#X obj 378 406 t f f;
#X obj 378 427 *;
#X obj 255 467 pack f f f;
#X msg 255 488 \$3 \$2;
#X obj 255 509 /;
#X obj 252 554 ftom;
#X obj 42 159 sqrt~;
#X connect 0 0 7 0;
#X connect 1 0 25 0;
#X connect 2 0 18 0;
#X connect 6 0 7 1;
#X connect 7 0 8 0;
#X connect 8 0 9 0;
#X connect 8 0 9 1;
#X connect 8 1 10 0;
#X connect 8 1 10 1;
#X connect 9 0 11 0;
#X connect 10 0 11 1;
#X connect 11 0 64 0;
#X connect 12 0 16 0;
#X connect 13 0 16 1;
#X connect 14 0 13 0;
#X connect 16 0 17 0;
#X connect 18 0 19 0;
#X connect 20 0 21 0;
#X connect 21 0 22 0;
#X connect 22 0 23 0;
#X connect 22 0 30 0;
#X connect 23 0 24 0;
#X connect 24 0 22 1;
#X connect 25 0 34 0;
#X connect 25 1 20 0;
#X connect 25 2 37 0;
#X connect 26 0 29 0;
#X connect 27 0 28 0;
#X connect 27 0 34 1;
#X connect 28 0 27 1;
#X connect 29 0 27 0;
#X connect 29 1 31 0;
#X connect 30 0 26 0;
#X connect 30 1 38 0;
#X connect 31 0 32 0;
#X connect 32 0 33 0;
#X connect 32 0 34 2;
#X connect 33 0 32 1;
#X connect 34 0 35 0;
#X connect 35 0 36 0;
#X connect 36 0 57 0;
#X connect 37 0 28 0;
#X connect 37 0 33 0;
#X connect 38 0 31 1;
#X connect 39 0 3 0;
#X connect 40 0 63 0;
#X connect 41 0 42 0;
#X connect 42 0 43 0;
#X connect 43 0 44 0;
#X connect 43 0 50 0;
```

```
#X connect 44 0 45 0;
#X connect 45 0 43 1;
#X connect 46 0 49 0;
#X connect 47 0 48 0;
#X connect 47 0 60 1;
#X connect 48 0 47 1;
#X connect 49 0 47 0;
#X connect 49 1 51 0;
#X connect 50 0 46 0;
#X connect 50 1 55 0;
#X connect 51 0 52 0;
#X connect 52 0 53 0;
#X connect 52 0 60 2;
#X connect 53 0 52 1;
#X connect 54 0 48 0;
#X connect 54 0 53 0;
#X connect 55 0 56 0;
#X connect 56 0 58 0;
#X connect 57 0 39 0;
#X connect 57 1 60 0;
#X connect 57 2 41 0;
#X connect 57 3 56 1;
#X connect 57 4 54 0;
#X connect 58 0 59 0;
#X connect 58 1 59 1;
#X connect 59 0 51 1;
#X connect 60 0 61 0;
#X connect 61 0 62 0;
#X connect 62 0 40 0;
#X connect 63 0 4 0;
#X connect 64 0 12 0;
#X restore 79 699 pd \$0-spec;
#N canvas 0 0 450 300 \$0-hann 0;
#X msg 194 86 0;
#X obj 88 135 osc~;
#X obj 88 87 samplerate~;
#X obj 88 158 *~ -0.5;
#X obj 88 181 +~ 0.5;
#X obj 80 206 tabwrite~ \$0-hann;
#X obj 88 111 / 512;
#X text 136 133 period 512;
#X obj 79 238 table \$0-hann 512;
#X obj 79 36 inlet;
#X connect 0 0 1 1;
#X connect 1 0 3 0;
#X connect 2 0 6 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 6 0 1 0;
#X connect 9 0 2 0;
#X connect 9 0 0 0;
#X connect 9 0 5 0;
#X restore 220 8 pd \$0-hann;
#X obj 35 741 route 0;
#X obj 40 677 t b b;
#X obj 83 294 *~ 0;
#X obj 321 293 *~ 0;
```

```
#X obj 36 761 list prepend chaotic;
#X obj 199 757 list prepend pitched;
#X obj 82 317 +~ 43;
#X obj 321 316 +~ 43;
#X obj 197 735 pack f f f f;
#X obj 35 720 pack 0 f f f f;
#X obj 45 99 delay 100;
#X obj 107 201 delay 100;
#X obj 64 127 delay 1800;
#X obj 348 89 div 2;
#X obj 391 88 div 2;
#X obj 101 26 t b b b;
#X obj 180 70 f;
#X obj 213 68 + 1;
#X obj 180 92 sel;
#X obj 309 87 *;
#X msg 180 172 \; pd quit;
#X obj 180 116 t b b;
#X obj 180 143 delay 1000;
#X obj 240 845 print image;
#X msg 253 778 done;
#X obj 307 127 div;
#X obj 342 127 mod;
#X obj 307 148 -;
#X obj 342 148 -;
#X obj 307 169 /;
#X obj 342 169 /;
#X obj 307 190 *;
#X obj 342 190 *;
#X obj 235 798 list prepend;
#X obj 252 104 t f f;
#X obj 290 777 pack f f;
#X obj 349 34 list append \$1 \$2 \$3 \$4 \$5;
#X obj 349 56 unpack f f f f f;
#X text 350 11 width height note modi size;
#X obj 307 209 +;
#X obj 342 211 +;
#X connect 0 0 1 0;
#X connect 0 0 9 0;
#X connect 0 0 22 0;
#X connect 0 0 23 0;
#X connect 2 0 0 0;
#X connect 3 0 7 0;
#X connect 4 0 3 0;
#X connect 5 0 28 0;
#X connect 6 0 27 0;
#X connect 8 0 40 0;
#X connect 9 0 11 0;
#X connect 10 0 17 0;
#X connect 10 1 17 1;
#X connect 11 0 10 0;
#X connect 11 0 22 1;
#X connect 12 0 35 0;
#X connect 12 1 10 1;
#X connect 12 2 13 0;
#X connect 12 2 41 0;
#X connect 13 0 11 1;
```

```
#X connect 14 0 11 1;
#X connect 15 0 26 0;
#X connect 15 1 21 0;
#X connect 16 0 33 0;
#X connect 16 1 33 2;
#X connect 17 0 18 1;
#X connect 17 1 18 2;
#X connect 18 0 19 0;
#X connect 19 0 15 0;
#X connect 20 0 33 3;
#X connect 21 0 16 0;
#X connect 21 1 20 0;
#X connect 22 0 20 0;
#X connect 23 0 34 3;
#X connect 23 1 34 4;
#X connect 25 0 29 0;
#X connect 26 0 34 0;
#X connect 26 1 23 1;
#X connect 27 0 31 0;
#X connect 28 0 32 0;
#X connect 29 0 58 0;
#X connect 30 0 58 0;
#X connect 31 0 2 0;
#X connect 32 0 4 0;
#X connect 33 0 30 0;
#X connect 34 0 25 0;
#X connect 35 0 14 0;
#X connect 35 0 23 2;
#X connect 35 0 37 0;
#X connect 36 0 12 0;
#X connect 37 0 18 0;
#X connect 37 0 36 0;
#X connect 38 0 52 1;
#X connect 38 0 54 1;
#X connect 39 0 53 1;
#X connect 39 0 55 1;
#X connect 40 0 12 0;
#X connect 40 1 61 0;
#X connect 40 2 24 0;
#X connect 41 0 42 0;
#X connect 41 0 43 0;
#X connect 42 0 41 1;
#X connect 43 0 46 0;
#X connect 43 1 59 0;
#X connect 44 0 43 1;
#X connect 46 0 47 0;
#X connect 46 1 49 0;
#X connect 47 0 45 0;
#X connect 49 0 48 0;
#X connect 50 0 52 0;
#X connect 50 0 60 0;
#X connect 51 0 53 0;
#X connect 51 0 60 1;
#X connect 52 0 54 0;
#X connect 53 0 55 0;
#X connect 54 0 56 0;
#X connect 55 0 57 0;
```

```
#X connect 56 0 64 0;
#X connect 57 0 65 0;
#X connect 58 0 48 0;
#X connect 59 0 50 0;
#X connect 59 1 51 0;
#X connect 60 0 58 1;
#X connect 61 0 62 0;
#X connect 62 0 38 0;
#X connect 62 0 44 0;
#X connect 62 0 50 1;
#X connect 62 0 51 1;
#X connect 62 1 39 0;
#X connect 62 1 44 1;
#X connect 62 2 64 1;
#X connect 62 3 65 1;
#X connect 62 4 57 1;
#X connect 62 4 56 1;
#X connect 64 0 33 1;
#X connect 64 0 34 2;
#X connect 64 0 32 1;
#X connect 64 0 31 1;
#X connect 65 0 16 1;
#X connect 65 0 27 1;
#X connect 65 0 28 1;
#X connect 65 0 34 1;
```

9 analyse/main.pd

```
#N canvas 0 0 549 895 10;
#X obj 82 370 osc~;
#X obj 83 397 s~ \$0-osc1;
#X obj 81 341 mtof~;
#X obj 321 369 osc~;
#X obj 320 340 mtof~;
#X obj 323 272 r~ \$0-osc1;
#X obj 82 269 r~ \$0-osc2;
#X obj 322 396 s~ \$0-osc2;
#X obj 165 36 loadbang;
#X obj 181 307 f;
#X obj 32 421 sigmund~ -npts 4096 -hop 4096;
#N canvas 0 0 450 300 \$0-statistics 0;
#X obj 18 11 inlet;
#X obj 18 108 + 1;
#X obj 82 87 f 0;
#X obj 47 87 + 1;
#X obj 18 87 f 0;
#X obj 17 33 t b f f f;
#X obj 143 92 f 0;
#X obj 108 90 + 1;
#X obj 109 60 *;
#X obj 19 157 swap;
#X obj 19 178 /;
#X obj 19 199 outlet;
#X obj 78 199 outlet;
#X obj 80 157 expr sqrt($f1 * $f3 - $f2 * $f2) / $f1;
#X obj 19 128 t f f;
#X obj 170 17 inlet;
```

```
#X obj 170 38 b;
#X obj 170 59 f 0;
#X connect 0 0 5 0;
#X connect 1 0 4 1;
#X connect 1 0 14 0;
#X connect 2 0 3 1;
#X connect 3 0 9 1;
#X connect 3 0 13 1;
#X connect 3 0 2 0;
#X connect 4 0 1 0;
#X connect 5 0 4 0;
#X connect 5 1 3 0;
#X connect 5 2 8 0;
#X connect 5 3 8 1;
#X connect 6 0 7 1;
#X connect 7 0 13 2;
#X connect 7 0 6 0;
#X connect 8 0 7 0;
#X connect 9 0 10 0;
#X connect 9 1 10 1;
#X connect 10 0 11 0;
#X connect 13 0 12 0;
#X connect 14 0 9 0;
#X connect 14 1 13 0;
#X connect 15 0 16 0;
#X connect 16 0 17 0;
#X connect 17 0 7 1;
#X connect 17 0 3 1;
#X connect 17 0 4 1;
#X restore 33 511 pd \$0-statistics;
#X obj 33 481 spigot;
#X obj 101 71 t b b b;
#X msg 106 455 0;
#X msg 58 455 1;
#X obj 36 651 route 0 1;
#X obj 197 715 swap;
#X obj 31 541 expr ($f1 > 0) && !($f2 > 1) \; $f1;
#X obj 35 607 pack 0 f f;
#X obj 37 628 route 0;
#X obj 302 712 snapshot~;
#X obj 192 693 t f b;
#N canvas 0 0 450 371 \$0-thd 0;
#X obj 21 17 inlet~;
#X obj 138 21 inlet;
#X obj 89 184 czero~;
#X obj 89 205 czero~;
#X obj 91 299 *~;
#X obj 91 320 +~;
#X obj 120 298 *~;
#X obj 89 224 cpole~;
#X obj 89 245 cpole~;
#X obj 139 49 mtof;
#X obj 309 161 * 0.99;
#X obj 141 154 * 0.99;
#X obj 137 206 * -1;
#X obj 136 246 * -1;
#X obj 236 44 loadbang;
```

```
#X obj 236 65 expr 4 * asin(1) / 48000;
#X obj 140 114 expr cos( $f2 * $f1) \; sin( $f2 * $f1 );
#X obj 90 341 lop~ 10;
#X obj 90 362 outlet~;
#X obj 214 47 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X connect 0 0 2 0;
#X connect 1 0 9 0;
#X connect 2 0 3 0;
#X connect 2 1 3 1;
#X connect 3 0 7 0;
#X connect 3 1 7 1;
#X connect 4 0 5 0;
#X connect 5 0 17 0;
#X connect 6 0 5 1;
#X connect 7 0 8 0;
#X connect 7 1 8 1;
#X connect 8 0 4 1;
#X connect 8 0 4 0;
#X connect 8 1 6 1;
#X connect 8 1 6 0;
#X connect 9 0 16 0;
#X connect 10 0 7 3;
#X connect 10 0 13 0;
#X connect 11 0 7 2;
#X connect 11 0 8 2;
#X connect 12 0 3 3;
#X connect 13 0 8 3;
#X connect 14 0 15 0;
#X connect 15 0 16 1;
#X connect 16 0 2 2;
#X connect 16 0 3 2;
#X connect 16 0 11 0;
#X connect 16 1 2 3;
#X connect 16 1 10 0;
#X connect 16 1 12 0;
#X connect 17 0 18 0;
#X connect 19 0 15 0;
#X restore 302 687 pd \$0-thd;
#N canvas 0 0 460 628 \$0-spec 0;
#X obj 41 20 inlet~;
#X obj 194 20 inlet;
#X obj 284 21 inlet;
#X obj 189 366 outlet;
#X obj 253 580 outlet;
#X obj 282 86 block~ 512 4;
#X obj 57 42 tabreceive~ \$0-hann;
#X obj 42 63 *~;
#X obj 43 85 rfft~;
#X obj 42 120 *~;
#X obj 71 119 *~;
#X obj 42 141 +~;
#X obj 42 175 *~ 0.1;
#X obj 94 176 *~ 0.9;
#X obj 93 145 tabreceive~ \$0-spectrum;
#X obj 285 108 table \$0-spectrum 512;
#X obj 79 201 +~;
```

```
#X obj 79 222 tabsend~ \$0-spectrum;
#X msg 284 42 const 0;
#X obj 284 63 s \$0-spectrum;
#X msg 213 65 256;
#X obj 213 86 until;
#X obj 213 107 f 0;
#X obj 254 109 + 1;
#X obj 254 130 mod 256;
#X obj 194 41 t b b b;
#X obj 214 166 tabread \$0-spectrum;
#X obj 214 207 +;
#X obj 246 208 f;
#X obj 214 187 t f f;
#X obj 213 128 t f f;
#X obj 326 192 *;
#X obj 326 212 +;
#X obj 358 213 f;
#X obj 189 253 pack f f f;
#X msg 189 274 \$3 \$2;
#X obj 189 295 /;
#X msg 257 71 0;
#X obj 343 166 * 93.75;
#X obj 189 342 ftom;
#X obj 252 533 sqrt;
#X msg 267 232 256;
#X obj 267 253 until;
#X obj 267 274 f 0;
#X obj 308 276 + 1;
#X obj 308 297 mod 256;
#X obj 268 333 tabread \$0-spectrum;
#X obj 270 439 +;
#X obj 304 445 f;
#X obj 268 354 t f f;
#X obj 267 295 t f f;
#X obj 361 447 *;
#X obj 360 468 +;
#X obj 404 466 f;
#X msg 311 238 0;
#X obj 397 333 * 93.75;
#X obj 378 385 -;
#X obj 190 316 t f b b f b;
#X obj 378 406 t f f;
#X obj 378 427 *;
#X obj 255 467 pack f f f;
#X msg 255 488 \$3 \$2;
#X obj 255 509 /;
#X obj 252 554 ftom;
#X obj 42 159 sqrt~;
#X connect 0 0 7 0;
#X connect 1 0 25 0;
#X connect 2 0 18 0;
#X connect 6 0 7 1;
#X connect 7 0 8 0;
#X connect 8 0 9 0;
#X connect 8 0 9 1;
#X connect 8 1 10 0;
#X connect 8 1 10 1;
```

```
#X connect 9 0 11 0;
#X connect 10 0 11 1;
#X connect 11 0 64 0;
#X connect 12 0 16 0;
#X connect 13 0 16 1;
#X connect 14 0 13 0;
#X connect 16 0 17 0;
#X connect 18 0 19 0;
#X connect 20 0 21 0;
#X connect 21 0 22 0;
#X connect 22 0 23 0;
#X connect 22 0 30 0;
#X connect 23 0 24 0;
#X connect 24 0 22 1;
#X connect 25 0 34 0;
#X connect 25 1 20 0;
#X connect 25 2 37 0;
#X connect 26 0 29 0;
#X connect 27 0 28 0;
#X connect 27 0 34 1;
#X connect 28 0 27 1;
#X connect 29 0 27 0;
#X connect 29 1 31 0;
#X connect 30 0 26 0;
#X connect 30 1 38 0;
#X connect 31 0 32 0;
#X connect 32 0 33 0;
#X connect 32 0 34 2;
#X connect 33 0 32 1;
#X connect 34 0 35 0;
#X connect 35 0 36 0;
#X connect 36 0 57 0;
#X connect 37 0 28 0;
#X connect 37 0 33 0;
#X connect 38 0 31 1;
#X connect 39 0 3 0;
#X connect 40 0 63 0;
#X connect 41 0 42 0;
#X connect 42 0 43 0;
#X connect 43 0 44 0;
#X connect 43 0 50 0;
#X connect 44 0 45 0;
#X connect 45 0 43 1;
#X connect 46 0 49 0;
#X connect 47 0 48 0;
#X connect 47 0 60 1;
#X connect 48 0 47 1;
#X connect 49 0 47 0;
#X connect 49 1 51 0;
#X connect 50 0 46 0;
#X connect 50 1 55 0;
#X connect 51 0 52 0;
#X connect 52 0 53 0;
#X connect 52 0 60 2;
#X connect 53 0 52 1;
#X connect 54 0 48 0;
#X connect 54 0 53 0;
```

```
#X connect 55 0 56 0;
#X connect 56 0 58 0;
#X connect 57 0 39 0;
#X connect 57 1 60 0;
#X connect 57 2 41 0;
#X connect 57 3 56 1;
#X connect 57 4 54 0;
#X connect 58 0 59 0;
#X connect 58 1 59 1;
#X connect 59 0 51 1;
#X connect 60 0 61 0;
#X connect 61 0 62 0;
#X connect 62 0 40 0;
#X connect 63 0 4 0;
#X connect 64 0 12 0;
#X restore 79 699 pd \$0-spec;
#N canvas 0 0 450 300 \$0-hann 0;
#X msg 194 86 0;
#X obj 88 135 osc~;
#X obj 88 87 samplerate~;
#X obj 88 158 *~ -0.5;
#X obj 88 181 +~ 0.5;
#X obj 80 206 tabwrite~ \$0-hann;
#X obj 88 111 / 512;
#X text 136 133 period 512;
#X obj 79 238 table \$0-hann 512;
#X obj 79 36 inlet;
#X connect 0 0 1 1;
#X connect 1 0 3 0;
#X connect 2 0 6 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 6 0 1 0;
#X connect 9 0 2 0;
#X connect 9 0 0 0;
#X connect 9 0 5 0;
#X restore 269 72 pd \$0-hann;
#X obj 35 741 route 0;
#X obj 40 677 t b b;
#X obj 181 285 snapshot~;
#X obj 182 200 +~ 0;
#X obj 182 178 *~ 0;
#X obj 181 156 noise~;
#X obj 83 294 *~ 0;
#X obj 321 293 *~ 0;
#X obj 300 153 f;
#X obj 352 153 f;
#X obj 300 111 netreceive \$1 1;
#X obj 36 761 list prepend chaotic;
#X obj 199 757 list prepend pitched;
#X obj 235 798 spigot;
#X obj 235 819 list prepend send;
#X obj 306 875 netsend 1;
#X obj 306 856 list trim;
#X obj 409 816 list prepend connect;
#X obj 382 736 t b a;
#X msg 382 757 1;
```

```
315 #X obj 82 317 +~ 43;
    #X obj 321 316 +~ 43;
    #X obj 242 305 f;
    #X obj 242 283 snapshot~;
    #X obj 243 198 +~ 0;
320 #X obj 243 176 *~ 0;
    #X obj 242 154 noise~;
    #X obj 300 132 route irange icentre brange bcentre connect;
    #X obj 398 154 f;
    #X obj 450 156 f;
325 #X obj 197 735 pack f f f f;
    #X obj 35 720 pack 0 f f f f;
    #X obj 45 99 delay 100;
    #X obj 107 201 delay 100;
    #X obj 64 127 delay 1800;
330 #X connect 0 0 1 0;
    #X connect 0 0 10 0;
    #X connect 0 0 23 0;
    #X connect 0 0 24 0;
    #X connect 2 0 0 0;
335 #X connect 3 0 7 0;
    #X connect 4 0 3 0;
    #X connect 5 0 33 0;
    #X connect 6 0 32 0;
    #X connect 8 0 13 0;
340 #X connect 8 0 25 0;
    #X connect 9 0 17 1;
    #X connect 9 0 32 1;
    #X connect 9 0 33 1;
    #X connect 9 0 57 1;
345 #X connect 10 0 12 0;
    #X connect 11 0 18 0;
    #X connect 11 1 18 1;
    #X connect 12 0 11 0;
    #X connect 12 0 23 1;
350 #X connect 13 0 58 0;
    #X connect 13 1 11 1;
    #X connect 13 2 14 0;
    #X connect 13 2 28 0;
    #X connect 13 2 49 0;
355 #X connect 14 0 12 1;
    #X connect 15 0 12 1;
    #X connect 16 0 27 0;
    #X connect 16 1 22 0;
    #X connect 17 0 56 0;
360 #X connect 17 1 56 2;
    #X connect 18 0 19 1;
    #X connect 18 1 19 2;
    #X connect 19 0 20 0;
    #X connect 20 0 16 0;
365 #X connect 21 0 56 3;
    #X connect 22 0 17 0;
    #X connect 22 1 21 0;
    #X connect 23 0 21 0;
    #X connect 24 0 57 3;
370 #X connect 24 1 57 4;
    #X connect 26 0 37 0;
```

```
#X connect 27 0 57 0;
#X connect 27 1 24 1;
#X connect 28 0 9 0;
#X connect 29 0 28 0;
#X connect 30 0 29 0;
#X connect 31 0 30 0;
#X connect 32 0 46 0;
#X connect 33 0 47 0;
#X connect 34 0 30 1;
#X connect 35 0 29 1;
#X connect 36 0 53 0;
#X connect 37 0 39 0;
#X connect 38 0 39 0;
#X connect 39 0 40 0;
#X connect 40 0 42 0;
#X connect 42 0 41 0;
#X connect 43 0 42 0;
#X connect 44 0 45 0;
#X connect 44 1 43 0;
#X connect 45 0 39 1;
#X connect 46 0 2 0;
#X connect 47 0 4 0;
#X connect 48 0 56 1;
#X connect 48 0 57 2;
#X connect 48 0 47 1;
#X connect 48 0 46 1;
#X connect 49 0 48 0;
#X connect 50 0 49 0;
#X connect 51 0 50 0;
#X connect 52 0 51 0;
#X connect 53 0 34 0;
#X connect 53 1 35 0;
#X connect 53 2 54 0;
#X connect 53 3 55 0;
#X connect 53 4 44 0;
#X connect 54 0 51 1;
#X connect 55 0 50 1;
#X connect 56 0 38 0;
#X connect 57 0 26 0;
#X connect 58 0 15 0;
#X connect 58 0 24 2;
#X connect 58 0 60 0;
#X connect 59 0 13 0;
#X connect 60 0 19 0;
#X connect 60 0 59 0;
```

10 analyse/noin-04-think-zinc-sink.pd

```
#N canvas 0 0 702 627 10;
#N canvas 0 0 900 323 \$0-think 0;
#X obj 101 276 outlet~;
#X obj 199 276 outlet~;
#X obj 100 245 samphold~;
#X obj 256 95 *~;
#X obj 158 184 phasor~;
#X obj 416 196 vline~;
#X obj 198 247 samphold~;
```

```
#X obj 257 184 phasor~;
#X msg 204 159 0;
#X msg 303 157 0.25;
#X msg 323 92 5 \, 6 1000 \, 7 0 1000 \, 3 0 1500 \, 6 1000 2500 \,
4 0 3500;
#X obj 323 9 inlet;
#X obj 323 30 mod 32;
#X obj 257 43 sig~ 50;
#X obj 323 71 spigot;
#X obj 419 10 inlet;
#X obj 272 135 loadbang;
#X obj 392 49 == 1;
#X obj 416 72 spigot;
#X obj 462 51 == 2;
#X msg 416 124 5 \, 6 500 \, 7 0 500 \, 3 0 750 \, 6 500 1250 \, 4
0 1750;
#X obj 323 49 sel 0 16;
#X msg 505 168 32 \, 3 16000;
#X obj 505 146 sel 3;
#X obj 95 22 inlet~;
#X obj 169 24 inlet~;
#X connect 2 0 0 0;
#X connect 3 0 4 0;
#X connect 3 0 7 0;
#X connect 4 0 2 1;
#X connect 5 0 3 1;
#X connect 6 0 1 0;
#X connect 7 0 6 1;
#X connect 8 0 4 1;
#X connect 9 0 7 1;
#X connect 10 0 5 0;
#X connect 11 0 12 0;
#X connect 12 0 21 0;
#X connect 13 0 3 0;
#X connect 14 0 10 0;
#X connect 15 0 17 0;
#X connect 15 0 19 0;
#X connect 15 0 23 0;
#X connect 16 0 9 0;
#X connect 16 0 8 0;
#X connect 17 0 14 1;
#X connect 18 0 20 0;
#X connect 19 0 18 1;
#X connect 20 0 5 0;
#X connect 21 0 14 0;
#X connect 21 0 18 0;
#X connect 21 1 18 0;
#X connect 22 0 5 0;
#X connect 23 0 22 0;
#X connect 24 0 2 0;
#X connect 25 0 6 0;
#X restore 324 355 pd \$0-think;
#N canvas 0 0 686 480 \$0-zink 0;
#X obj 218 259 inlet~;
#X obj 363 264 inlet~;
#X obj 195 327 delwrite~ \$0-L 1000;
#X obj 348 328 delwrite~ \$0-R 1000;
```

```
#X obj 111 424 outlet~;
#X obj 300 417 outlet~;
#X obj 12 7 inlet;
#X obj 59 257 vd~ \$0-R;
#X obj 110 223 vline~;
#X obj 129 257 vd~ \$0-L;
#X obj 112 364 *~;
#X obj 299 363 *~;
#X msg 106 148 250 25;
#X msg 13 149 125 25;
#X msg 67 149 8 25;
#X obj 129 279 *~ -0.9;
#X obj 59 279 *~ -0.9;
#X obj 12 74 sel 0 4 8 11 12 14 6 7;
#X obj 473 360 inlet~;
#X obj 300 393 expr~ $v3*$v1+$v2*(1-$v3);
#X obj 110 393 expr~ $v3*$v1+$v2*(1-$v3);
#X obj 100 7 inlet;
#X obj 12 30 spigot;
#X obj 13 53 mod 16;
#X obj 172 30 spigot;
#X obj 55 7 == 0;
#X obj 212 7 == 1;
#X obj 173 53 mod 16;
#X msg 267 150 250 25;
#X msg 174 151 125 25;
#X msg 228 151 8 25;
#X obj 173 76 sel 0 4 8 11 12 14 6 7;
#X obj 323 119 random 8;
#X obj 322 139 moses 4;
#X obj 321 161 + 1;
#X obj 323 229 <<;
#X msg 322 254 \$1 25;
#X obj 320 183 t b f;
#X msg 320 206 31;
#X msg 352 72 seed 1.12346e+06;
#X obj 352 50 loadbang;
#X obj 322 97 bang;
#X connect 0 0 2 0;
#X connect 0 0 10 1;
#X connect 0 0 20 1;
#X connect 1 0 3 0;
#X connect 1 0 11 1;
#X connect 1 0 19 1;
#X connect 6 0 22 0;
#X connect 6 0 24 0;
#X connect 7 0 16 0;
#X connect 8 0 7 0;
#X connect 8 0 9 0;
#X connect 9 0 15 0;
#X connect 10 0 20 0;
#X connect 11 0 19 0;
#X connect 12 0 8 0;
#X connect 13 0 8 0;
#X connect 14 0 8 0;
#X connect 15 0 3 0;
#X connect 15 0 11 0;
```

```
#X connect 16 0 2 0;
#X connect 16 0 10 0;
#X connect 17 0 13 0;
#X connect 17 1 14 0;
#X connect 17 2 12 0;
#X connect 17 3 14 0;
#X connect 17 4 12 0;
#X connect 17 5 14 0;
#X connect 17 6 13 0;
#X connect 17 7 14 0;
#X connect 18 0 19 2;
#X connect 18 0 20 2;
#X connect 19 0 5 0;
#X connect 20 0 4 0;
#X connect 21 0 25 0;
#X connect 21 0 26 0;
#X connect 22 0 23 0;
#X connect 23 0 17 0;
#X connect 24 0 27 0;
#X connect 25 0 22 1;
#X connect 26 0 24 1;
#X connect 27 0 31 0;
#X connect 28 0 8 0;
#X connect 29 0 8 0;
#X connect 30 0 8 0;
#X connect 31 0 29 0;
#X connect 31 1 30 0;
#X connect 31 2 28 0;
#X connect 31 3 30 0;
#X connect 31 4 28 0;
#X connect 31 5 30 0;
#X connect 31 6 29 0;
#X connect 31 7 30 0;
#X connect 31 8 41 0;
#X connect 32 0 33 0;
#X connect 33 0 34 0;
#X connect 34 0 37 0;
#X connect 35 0 36 0;
#X connect 36 0 8 0;
#X connect 37 0 38 0;
#X connect 37 1 35 1;
#X connect 38 0 35 0;
#X connect 39 0 32 0;
#X connect 40 0 39 0;
#X connect 41 0 32 0;
#X restore 302 395 pd \$0-zink;
#N canvas 0 0 618 663 \$0-sink 0;
#X obj 19 28 inlet;
#X obj 211 30 inlet~;
#X obj 364 31 inlet~;
#X obj 176 532 delwrite~ \$0-L2 1000;
#X obj 325 532 delwrite~ \$0-R2 1000;
#X floatatom 19 90 0 0 0 0 - - -;
#X obj 82 358 *~;
#X obj 205 295 line~;
#X floatatom 229 113 0 0 0 0 - - -;
#X obj 205 269 pack 0 200;
```

```
#X obj 19 143 exp;
#X floatatom 19 169 6 0 0 0 - - -;
#X obj 19 259 /;
#X obj 146 189 * 0.001;
#X obj 19 410 cos~;
#X obj 19 437 *~;
#X obj 19 466 +~;
#X obj 106 317 wrap~;
#X obj 251 360 *~;
#X obj 188 420 cos~;
#X obj 188 447 *~;
#X obj 146 216 t b f;
#X floatatom 19 285 6 0 0 0 - - -;
#X obj 106 290 +~ 0.5;
#X obj 19 358 -~ 0.5;
#X obj 19 384 *~ 0.5;
#X obj 188 359 -~ 0.5;
#X obj 188 392 *~ 0.5;
#X obj 19 196 - 1;
#X obj 19 117 * 0.05776;
#X obj 19 222 * -1;
#X text 53 86 <-- transposition;
#X text 96 99 (halftones);
#X text 82 163 speed;
#X text 81 177 change;
#X text 250 113 <--window (msec);
#X obj 19 316 phasor~;
#X obj 231 138 max 1;
#X obj 82 410 vd~ \$0-L2;
#X obj 251 422 vd~ \$0-L2;
#X obj 82 384 +~ 250;
#X obj 251 393 +~ 250;
#X floatatom 359 87 0 0 0 0 - - -;
#X obj 422 358 *~;
#X obj 545 295 line~;
#X obj 545 269 pack 0 200;
#X obj 359 143 exp;
#X floatatom 359 169 6 0 0 0 - - -;
#X obj 359 259 /;
#X obj 486 189 * 0.001;
#X obj 359 410 cos~;
#X obj 359 437 *~;
#X obj 381 469 +~;
#X obj 446 317 wrap~;
#X obj 591 360 *~;
#X obj 528 420 cos~;
#X obj 528 447 *~;
#X obj 486 216 t b f;
#X floatatom 359 285 6 0 0 0 - - -;
#X obj 446 290 +~ 0.5;
#X obj 359 358 -~ 0.5;
#X obj 359 384 *~ 0.5;
#X obj 528 359 -~ 0.5;
#X obj 528 392 *~ 0.5;
#X obj 359 196 - 1;
#X obj 359 117 * 0.05776;
#X obj 359 222 * -1;
```

```
#X text 392 86 <-- transposition;
#X text 436 99 (halftones);
#X text 422 163 speed;
#X text 421 177 change;
#X obj 359 316 phasor~;
#X obj 422 384 +~ 250;
#X obj 591 393 +~ 250;
#X obj 422 410 vd~ \$0-R2;
#X obj 591 422 vd~ \$0-R2;
#X obj 381 494 *~ 0.5;
#X obj 200 499 *~ 0.5;
#X obj 145 613 outlet~;
#X obj 334 606 outlet~;
#X obj 553 529 inlet~;
#X obj 334 582 expr~ $v3*$v1+$v2*(1-$v3);
#X obj 144 582 expr~ $v3*$v1+$v2*(1-$v3);
#X obj 78 29 inlet;
#X msg 231 88 40;
#X obj 231 64 loadbang;
#X connect 0 0 5 0;
#X connect 1 0 3 0;
#X connect 1 0 82 1;
#X connect 2 0 4 0;
#X connect 2 0 81 1;
#X connect 5 0 29 0;
#X connect 6 0 40 0;
#X connect 7 0 6 1;
#X connect 7 0 18 1;
#X connect 8 0 37 0;
#X connect 9 0 7 0;
#X connect 10 0 11 0;
#X connect 11 0 28 0;
#X connect 12 0 22 0;
#X connect 13 0 21 0;
#X connect 14 0 15 0;
#X connect 15 0 16 0;
#X connect 16 0 77 0;
#X connect 16 0 82 0;
#X connect 17 0 18 0;
#X connect 17 0 26 0;
#X connect 18 0 41 0;
#X connect 19 0 20 0;
#X connect 20 0 16 1;
#X connect 21 0 12 0;
#X connect 21 1 12 1;
#X connect 22 0 36 0;
#X connect 23 0 17 0;
#X connect 24 0 25 0;
#X connect 25 0 14 0;
#X connect 26 0 27 0;
#X connect 27 0 19 0;
#X connect 28 0 30 0;
#X connect 29 0 10 0;
#X connect 30 0 12 0;
#X connect 36 0 6 0;
#X connect 36 0 24 0;
#X connect 36 0 23 0;
```

```
#X connect 37 0 13 0;
#X connect 37 0 9 0;
#X connect 37 0 49 0;
#X connect 37 0 45 0;
#X connect 38 0 15 1;
#X connect 39 0 20 1;
#X connect 40 0 38 0;
#X connect 41 0 39 0;
#X connect 42 0 65 0;
#X connect 43 0 72 0;
#X connect 44 0 43 1;
#X connect 44 0 54 1;
#X connect 45 0 44 0;
#X connect 46 0 47 0;
#X connect 47 0 64 0;
#X connect 48 0 58 0;
#X connect 49 0 57 0;
#X connect 50 0 51 0;
#X connect 51 0 52 0;
#X connect 52 0 76 0;
#X connect 52 0 81 0;
#X connect 53 0 54 0;
#X connect 53 0 62 0;
#X connect 54 0 73 0;
#X connect 55 0 56 0;
#X connect 56 0 52 1;
#X connect 57 0 48 0;
#X connect 57 1 48 1;
#X connect 58 0 71 0;
#X connect 59 0 53 0;
#X connect 60 0 61 0;
#X connect 61 0 50 0;
#X connect 62 0 63 0;
#X connect 63 0 55 0;
#X connect 64 0 66 0;
#X connect 65 0 46 0;
#X connect 66 0 48 0;
#X connect 71 0 43 0;
#X connect 71 0 60 0;
#X connect 71 0 59 0;
#X connect 72 0 74 0;
#X connect 73 0 75 0;
#X connect 74 0 51 1;
#X connect 75 0 56 1;
#X connect 76 0 3 0;
#X connect 77 0 4 0;
#X connect 80 0 81 2;
#X connect 80 0 82 2;
#X connect 81 0 79 0;
#X connect 82 0 78 0;
#X connect 83 0 42 0;
#X connect 84 0 8 0;
#X connect 85 0 84 0;
#X restore 268 470 pd \$0-sink;
#X obj 440 350 vline~;
#X obj 502 349 vline~;
#X obj 29 327 div 32;
```

```
#X obj 29 374 sel 0 1 2 3;
#X obj 29 350 mod 4;
#X obj 30 305 + 8;
#X obj 128 308 + 12;
#X msg 126 402 12;
#X msg 29 403 19;
#X msg 63 404 20;
#X msg 94 405 17;
#X msg 158 403 7;
#X msg 189 404 9;
#X obj 126 352 mod 8;
#X obj 126 331 div 32;
#X obj 125 374 sel 0 2 4 6 7;
#X msg 217 404 10;
#X obj 13 78 t f f f;
#X msg 213 194 0;
#X msg 245 193 1;
#X obj 28 278 spigot;
#X msg 277 193 12;
#X msg 285 173 1 32000;
#X msg 309 194 0;
#X msg 341 194 1;
#X msg 373 151 1 32000;
#X msg 405 196 0;
#X msg 417 172 0 32000;
#X msg 437 199 1;
#X msg 414 230 0;
#X msg 469 197 1;
#X msg 481 172 1 32000;
#X msg 501 198 0;
#X obj 171 434 f;
#X obj 80 434 f;
#X msg 518 146 1;
#X msg 533 201 2;
#N canvas 0 0 450 300 \$0-fink 0;
#X obj 22 28 inlet~;
#X obj 188 27 inlet~;
#X obj 299 25 inlet~;
#X obj 188 91 expr~ tanh($v1);
#X obj 24 153 expr~ tanh($v1*10);
#X obj 188 151 expr~ tanh($v1*10);
#X obj 23 242 outlet~;
#X obj 189 242 outlet~;
#X obj 300 68 +~ 31.35;
#X obj 23 91 expr~ tanh($v1);
#X obj 301 90 mtof~;
#X obj 188 124 vcf~ 20;
#X obj 23 124 vcf~ 20;
#X obj 300 47 *~ 108;
#X obj 188 199 *~ 0.333;
#X obj 23 201 *~ 0.333;
#X connect 0 0 9 0;
#X connect 1 0 3 0;
#X connect 2 0 13 0;
#X connect 3 0 11 0;
#X connect 4 0 15 0;
#X connect 5 0 14 0;
```

```
#X connect 8 0 10 0;
#X connect 9 0 12 0;
#X connect 10 0 11 1;
#X connect 10 0 12 1;
#X connect 11 0 5 0;
#X connect 12 0 4 0;
#X connect 13 0 8 0;
#X connect 14 0 7 0;
#X connect 15 0 6 0;
#X restore 351 469 pd \$0-fink;
#X obj 422 444 vline~;
#X msg 551 147 1 64000;
#X msg 569 201 3;
#X obj 213 49 sel -33 0 256 480 512 768 1280 1536 1792 2048 2304 2432
2560 2816;
#X msg 607 202 1;
#X msg 608 169 0;
#X msg 585 276 0;
#X msg 572 301 1 32000;
#X obj 284 584 outlet~;
#X obj 346 584 outlet~;
#X obj 15 19 inlet;
#X obj 308 13 inlet~;
#X obj 358 13 inlet~;
#X obj 81 578 outlet;
#X obj 284 538 *~;
#X obj 345 542 *~;
#X obj 134 536 vline~;
#X obj 12 47 route float;
#X obj 300 521 *~ 0.375;
#X obj 367 520 *~ 0.375;
#X obj 173 560 -~ 1;
#X obj 173 581 *~ -1;
#X obj 264 558 *~;
#X obj 325 562 *~;
#X msg 83 500 1 4000;
#X msg 133 499 0 4000;
#X connect 0 0 1 2;
#X connect 0 1 1 3;
#X connect 1 0 2 2;
#X connect 1 0 40 0;
#X connect 1 1 2 3;
#X connect 1 1 40 1;
#X connect 2 0 59 0;
#X connect 2 1 60 0;
#X connect 3 0 1 4;
#X connect 4 0 2 4;
#X connect 5 0 7 0;
#X connect 6 0 11 0;
#X connect 6 1 12 0;
#X connect 6 2 11 0;
#X connect 6 3 13 0;
#X connect 7 0 6 0;
#X connect 8 0 5 0;
#X connect 9 0 17 0;
#X connect 10 0 36 0;
#X connect 11 0 37 0;
```

```
#X connect 12 0 37 0;
#X connect 13 0 37 0;
#X connect 14 0 36 0;
#X connect 15 0 36 0;
#X connect 16 0 18 0;
#X connect 17 0 16 0;
#X connect 18 0 10 0;
#X connect 18 1 14 0;
#X connect 18 2 10 0;
#X connect 18 3 15 0;
#X connect 18 4 19 0;
#X connect 19 0 36 0;
#X connect 20 0 1 0;
#X connect 20 0 0 2;
#X connect 20 0 23 0;
#X connect 20 1 44 0;
#X connect 21 0 4 0;
#X connect 21 0 3 0;
#X connect 21 0 0 3;
#X connect 21 0 23 1;
#X connect 21 0 1 1;
#X connect 21 0 41 0;
#X connect 22 0 0 3;
#X connect 23 0 8 0;
#X connect 23 0 9 0;
#X connect 24 0 2 0;
#X connect 24 0 2 1;
#X connect 25 0 4 0;
#X connect 26 0 0 3;
#X connect 27 0 0 3;
#X connect 27 0 23 1;
#X connect 28 0 3 0;
#X connect 29 0 0 3;
#X connect 30 0 4 0;
#X connect 31 0 0 3;
#X connect 32 0 23 1;
#X connect 33 0 23 1;
#X connect 34 0 4 0;
#X connect 35 0 4 0;
#X connect 36 0 2 1;
#X connect 37 0 2 0;
#X connect 38 0 1 1;
#X connect 39 0 0 3;
#X connect 40 0 59 0;
#X connect 40 1 60 0;
#X connect 41 0 40 2;
#X connect 42 0 41 0;
#X connect 43 0 0 3;
#X connect 44 0 21 0;
#X connect 44 0 65 0;
#X connect 44 1 22 0;
#X connect 44 2 24 0;
#X connect 44 2 25 0;
#X connect 44 3 26 0;
#X connect 44 4 27 0;
#X connect 44 5 28 0;
#X connect 44 6 29 0;
```

```
#X connect 44 6 30 0;
#X connect 44 6 32 0;
#X connect 44 7 31 0;
#X connect 44 8 33 0;
#X connect 44 8 34 0;
#X connect 44 9 35 0;
#X connect 44 9 38 0;
#X connect 44 9 42 0;
#X connect 44 10 39 0;
#X connect 44 10 48 0;
#X connect 44 11 43 0;
#X connect 44 12 45 0;
#X connect 44 12 46 0;
#X connect 44 12 47 0;
#X connect 44 13 66 0;
#X connect 45 0 0 3;
#X connect 46 0 1 1;
#X connect 47 0 4 0;
#X connect 48 0 4 0;
#X connect 51 0 58 0;
#X connect 52 0 0 0;
#X connect 52 0 63 1;
#X connect 53 0 0 1;
#X connect 53 0 64 1;
#X connect 55 0 49 0;
#X connect 56 0 50 0;
#X connect 57 0 55 0;
#X connect 57 0 56 0;
#X connect 57 0 61 0;
#X connect 58 0 20 0;
#X connect 59 0 55 1;
#X connect 60 0 56 1;
#X connect 61 0 62 0;
#X connect 62 0 63 0;
#X connect 62 0 64 0;
#X connect 63 0 49 0;
#X connect 64 0 50 0;
#X connect 65 0 57 0;
#X connect 66 0 57 0;
```

11 analyse/pd-0.42-6_fix_batch.patch

```
diff -ruw pd-0.42-6/src/m_sched.c pd-0.42-6-new/src/m_sched.c
--- pd-0.42-6/src/m_sched.c      2010-04-09 04:42:33.000000000 +0100
+++ pd-0.42-6-new/src/m_sched.c  2012-10-19 12:00:11.000000000 +0100
@@ -571,7 +571,10 @@
     sys_time_per_dsp_tick = (TIMEUNITPERSEC) *
         ((double)sys_schedblocksize) / sys_dacsr;
     while (sys_quit != SYS_QUIT_QUIT)
+    {
         sched_tick(sys_time + sys_time_per_dsp_tick);
+        sys_pollgui();
+    }
     return (0);
 }
```

12 analyse/start-image.sh

```
#!/bin/bash
#
# suggested usage (for a quad-core, launching 3 batches):
#
# $ ./start.sh ~/opt/pd-0.42/bin/pd -jack 3 5000 | ts > start.sh.log
#
# on my desktop with 3 cores doing batch work I get ~50 points per second
# on my laptop  with 1 cores doing batch work I get ~12 points per second
#
PD="${1}"
WIDTH="${2}"
HEIGHT="${3}"
NOTE="${4}"
MODI="${5}"
SIZE="${6}"
if [ "x${PD}" = "x" ]
then
  PD="$(which pd)"
fi
if [ "x${WIDTH}" = "x" ]
then
  WIDTH="256"
fi
if [ "x${HEIGHT}" = "x" ]
then
  HEIGHT="144"
fi
if [ "x${NOTE}" = "x" ]
then
  NOTE="36"
fi
if [ "x${MODI}" = "x" ]
then
  MODI="57.875"
fi
if [ "x${SIZE}" = "x" ]
then
  SIZE="12"
fi
"${PD}" -stderr -nrt -r 48000 -batch -nosound -open loader-image.pd -send "; go ↙
    ↳ ${WIDTH} ${HEIGHT} ${NOTE} ${MODI} ${SIZE}" >"${WIDTH}x${HEIGHT}_${NOTE}x$↙
    ↳ {MODI}_${SIZE}.log" 2>&1
```

13 analyse/start-images.sh

```
#!/bin/bash
for s in 64.0 32.0 16.0 8.0 4.0 2.0 1.0 0.5 0.25 0.125 0.0625 0.03125 0.015625 ↙
    ↳ 0.0078125 0.00390625 0.001953125
do
  echo "${s}"
done |
xargs -n 1 -P 4 ./start-image.sh ~/opt/pd-0.42/bin/pd 128 128 36 57.875
```

14 analyse/start.sh

```
#!/bin/bash
#
# suggested usage (for a quad-core, launching 3 batches):
#
# $ ./start.sh ~/opt/pd-0.42/bin/pd -jack 3 5000 | ts > start.sh.log
#
# on my desktop with 3 cores doing batch work I get ~50 points per second
# on my laptop  with 1 cores doing batch work I get ~12 points per second
#
PD="${1}"
API="${2}"
COUNT="${3}"
PORT="${4}"
if [ "x${PD}" = "x" ]
then
  PD="$(which pd)"
fi
if [ "x${API}" = "x" ]
then
  API="-alsa"
fi
if [ "x${COUNT}" = "x" ]
then
  COUNT="1"
fi
if [ "x${PORT}" = "x" ]
then
  PORT="5000"
fi
for i in $(seq 1 "${COUNT}")
do
  "${PD}" -nrt -r 48000 -batch -nosound -open loader.pd -send "; port $((PORT + i))" 2>&1 &
done
sleep 5
"${PD}" -noadc -rt -r 48000 "${API}" -channels 2 -path "$(dirname "${PD}")/../lib/pd/extra/Gem":"$(dirname "${PD}")/../lib/pd/extra/lua" -lib Gem:lua -open controllers.pd  -send "; batch ${COUNT} $((PORT + 1))" -stderr 2>&1 &
wait
```

15 .gitignore

```
-
*.log
*.data
*.hi
*.o
*.frag.c
analyse/log2rgb
perform/soft-rock-revisited
```

16 gpu/audio.c

```
/* ===========================================================================
```

```
soft-rock-revisited
Copyright (C) 2010-2012  Claude Heiland-Allen <claude@mathr.co.uk>
-------------------------------------------------------------------------------
Audio output
=============================================================================== */

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "config.h"
#include "audio.h"

volatile int audio_beat = 0;

static inline float mtoff(float f) {
  return 8.17579891564f * expf(0.0577622650f * f);
}

void audio_compute(struct audio *audio, jack_default_audio_sample_t **out, ↙
    ↳ jack_nframes_t nframes) {
  int b = audio_beat;
  float sr = audio->sr;
  float ophase[config_oscillators][2];
  float phase[config_oscillators][2];
  float note[config_oscillators];
  float modi[config_oscillators];
  note[0] = audio->map[b].note_mouse;
  modi[0] = audio->map[b].modi_mouse;
  float nradius = audio->map[b].nrange / (float) audio->map[b].win_width;
  float mradius = audio->map[b].mrange / (float) audio->map[b].win_height;
  for (int o = 1; o < config_oscillators; ++o) {
    note[o] = note[0] + nradius * cos(o * 6.283185307179586 / (↙
        ↳ config_oscillators - 1));
    modi[o] = modi[0] + mradius * sin(o * 6.283185307179586 / (↙
        ↳ config_oscillators - 1));
  }
  float pan[config_oscillators][2];
  for (int o = 0; o < config_oscillators; ++o) {
    phase[o][0] = audio->phase[o][0];
    phase[o][1] = audio->phase[o][1];
    if (o == 0) {
      pan[o][1] = sqrtf(0.5);
    } else {
      pan[o][1] = cosf(6.283185307179586f * 0.125 * (0.9 * cosf((o + 0.5) * ↙
          ↳ 6.283185307179586f / (config_oscillators - 1)) + 1.0));
    }
    pan[o][0] = sqrtf(1 - pan[o][1] * pan[o][1]);
    pan[o][0] *= (o == 0 ? 0.5 : (0.5 / (config_oscillators - 1)));
    pan[o][1] *= (o == 0 ? 0.5 : (0.5 / (config_oscillators - 1)));
  }
  for (jack_nframes_t n = 0; n < nframes; ++n) {
    out[0][n] = 0.0;
    out[1][n] = 0.0;
    // FM oscillators
    for (int o = 0; o < config_oscillators; ++o) {
```

```
55          audio->osc[o][0][audio->oscn] = cosf(6.283185307179586f * phase[o][0]);
            audio->osc[o][1][audio->oscn] = cosf(6.283185307179586f * phase[o][1]);
            for (int i = 0; i < 2; ++i) {
              int j = 1 - i;
              float p = phase[o][i] + mtoff(note[o] + modi[o] * audio->osc[o][j][(↙
                  ↳ audio->oscn + 1) % config_blocksize]) / sr;
60            ophase[o][i] = p - floorf(p);
            }
            for (int i = 0; i < 2; ++i) {
              phase[o][i] = ophase[o][i];
              out[i][n] += pan[o][i] * audio->osc[o][i][audio->oscn];
65          }
          }
          // DC blocker
          for (int i = 0; i < 2; ++i) {
            audio->x1[i] = audio->x[i];
70          audio->x[i] = out[i][n];
            out[i][n] = audio->x[i] - audio->x1[i] + 0.999 * audio->y1[i];
            audio->y1[i] = out[i][n];
          }
          // feedback delay
75        audio->oscn = (audio->oscn + 1) % config_blocksize;
        }
        // update state
        audio->sphase += nframes / (double) audio->sr;
        audio->bphase += nframes;
80      if (audio->bphase >= audio->blength) {
          audio->bphase -= audio->blength;
          audio_beat = (audio_beat + 1) % config_beats;
        }
        for (int o = 0; o < config_oscillators; ++o) {
85        audio->phase[o][0] = phase[o][0];
          audio->phase[o][1] = phase[o][1];
        }
      }

90    /* JACK audio processing callback */
      static int audio_process(jack_nframes_t nframes, void *arg) {
        struct audio *audio = arg;
        /* get JACK buffers */
        jack_default_audio_sample_t *out[2];
95      out[0] = (jack_default_audio_sample_t *)
          jack_port_get_buffer(audio->port[0], nframes);
        out[1] = (jack_default_audio_sample_t *)
          jack_port_get_buffer(audio->port[1], nframes);
        audio_compute(audio, out, nframes);
100     return 0;
      }

      /* JACK sample rate callback */
      static int audio_srate(jack_nframes_t sr, void *arg) {
105     struct audio *audio = arg;
        if ((jack_nframes_t) audio->sr != sr) {
          fprintf(stderr, "WARN: JACK sample rate: %d != %d\n", sr, audio->sr);
        }
        return 0;
110   }
```

```
/* JACK error callback */
static void audio_error(const char *desc) {
  fprintf(stderr, "JACK error: %s\n", desc);
}

/* JACK shutdown callback */
static void audio_shutdown(void *arg) {
  fprintf(stderr, "JACK shutdown\n");
}

/* JACK timebase callback */
static void audio_timebase(jack_transport_state_t state, jack_nframes_t nframes,
    jack_position_t *pos, int new_pos, void *arg) {
  //struct audio *audio = arg;
  if (new_pos) {
    pos->valid = JackPositionBBT;
    pos->beats_per_bar = 4;
    pos->beat_type = 0.25;
    pos->ticks_per_beat = 1920;
    pos->beats_per_minute = 120;
    pos->bar = 1;
    pos->beat = 1;
    pos->tick = 0;
    pos->bar_start_tick = (pos->bar - 1) * pos->beats_per_bar * pos->
        ticks_per_beat;
  } else {
    pos->tick +=nframes * pos->ticks_per_beat * pos->beats_per_minute / (pos->
        frame_rate * 60);
    while (pos->tick >= pos->ticks_per_beat) {
      pos->tick -= pos->ticks_per_beat;
      if (++pos->beat > pos->beats_per_bar) {
        pos->beat = 1;
        ++pos->bar;
        pos->bar_start_tick += pos->beats_per_bar * pos->ticks_per_beat;
      }
    }
  }
}

/* exit callback */
void audio_atexit(struct audio *audio) {
  jack_client_close(audio->client);
}

//=============================================================================
// ...
struct audio *audio_init(struct audio *audio, struct map *map) {
  if (! audio) { return 0; }
  audio->map = map;
  audio->samples = 0;
  for (int o = 0; o < config_oscillators; ++o) {
    audio->phase[o][0] = 0;
    audio->phase[o][1] = 0;
  }
  audio->oscn = 0;
  audio->sr = config_samplerate;
```

```
    audio->sphase = 0;
    audio->bphase = 0;
    audio->blength = audio->sr * 60.0 / config_bpm;
    jack_set_error_function(audio_error);
    if (!(audio->client = jack_client_new("srrv"))) {
      fprintf (stderr, "JACK server not running?\n");
    } else {
      jack_set_process_callback(audio->client, audio_process, audio);
      jack_set_sample_rate_callback(audio->client, audio_srate, audio);
      jack_on_shutdown(audio->client, audio_shutdown, audio);
      /* create ports */
      audio->port[0] = jack_port_register(
        audio->client, "output_1", JACK_DEFAULT_AUDIO_TYPE, JackPortIsOutput, 0
      );
      audio->port[1] = jack_port_register(
        audio->client, "output_2", JACK_DEFAULT_AUDIO_TYPE, JackPortIsOutput, 0
      );
      /* activate audio */
      if (jack_activate(audio->client)) {
        fprintf (stderr, "cannot activate JACK client\n");
      } else {
        /* must be activated before connecting JACK ports */
        const char **ports;
        if ((ports = jack_get_ports(
          audio->client, NULL, NULL, JackPortIsPhysical | JackPortIsInput
        ))) {
          /* connect up to two physical playback ports */
          int i = 0;
          while (ports[i] && i < 2) {
            if (jack_connect(
              audio->client, jack_port_name(audio->port[i]), ports[i]
            )) {
              fprintf(stderr, "cannot connect JACK output port\n");
            }
            i++;
          }
          free(ports);
        }
        /* become timebase master */
        if (jack_set_timebase_callback(audio->client, 0, audio_timebase, audio)) {
          fprintf(stderr, "cannot become JACK master\n");
        }
        jack_transport_start(audio->client);
      }
    }
    return audio;
}

// EOF
```

17 gpu/audio.h

```
/* ================================================================
Audio output
================================================================ */

#ifndef AUDIO_H
```

```
#define AUDIO_H 1

#include <stdio.h>
#include <jack/jack.h>

#include "config.h"
#include "map.h"

//=============================================================================
// audio state

extern volatile int audio_beat;

struct audio {
  struct map *map;
  double samples; // total samples scanned
  jack_client_t *client;
  jack_port_t *port[2];
  double sphase;
  int sr;
  float phase[config_oscillators][2];
  float osc[config_oscillators][2][config_blocksize];
  int oscn;
  double bphase;
  double blength;
  float x[2], x1[2], y1[2];
};

//=============================================================================
// prototypes
struct audio *audio_init(struct audio *audio, struct map *map);
void audio_atexit(struct audio *audio);

#endif
```

18 gpu/boids.c

```
/* NAME
 *   boids - simulate a flock of android birds from Brooklyn
 * NOTES
 *   As noted below, this program makes heavy use of some global
 *   variables.
 *
 *   The compute_new_heading() function is somewhat monolithic,
 *   but it is well documented at least.  Change it with some care.
 * RULES
 *   All of the rules have a weight option and a radius option.
 *   The radius option specifies how close a boid need to be to
 *   another in order for the rule to be acted upon.  The weight
 *   is used when combining all of the rules actions into a single
 *   new velocity vector.
 *
 *   The four rules can be simply described as follows:
 *
 *      Centering: move towards the center of any boids in my
 *                 viewing area.
```

```
 *
 *      Copying: attempt to move in the average direction of
 *               that all boids that can be seen are moving
 *               in.
 *
 *      Avoidance: ``Please don't stand so close to me.''
 *                 Move away from any close flyers.
 *
 *      Visual: move in such a way that the bonehead
 *              obstructing your view no longer interferes.
 *
 *   The four rules are then normalized and added together to make
 *   the next velocity vector of the boid.  All radii in the rules
 *   are in terms of pixels.
 *
 *   You may wish to try turning on and off different combinations
 *   of the rules to see how the boids' behaviors change.  For example,
 *   if you turn off the avoidance rule, increase the centering radius
 *   and weight, and increase the viewing angle to nearly 360 degrees,
 *   then the boids will behave like a pack of wolves fighting over
 *   the center.  Other changes can yield similar surprises.
 * BUGS
 *   No sanity checks are performed to make sure that any of the
 *   options make sense.
 * AUTHOR
 *   Copyright (c) 1997, Gary William Flake.
 *
 *   Permission granted for any use according to the standard GNU
 *   ``copyleft'' agreement provided that the author's comments are
 *   neither modified nor removed.  No warranty is given or implied.
 */


#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "misc.h"

int width = 640, height = 480, num = 20, len = 20, mag = 1;
int seed = 0, invert = 0, steps = 100000000, psdump = 0;

double angle = 270.0, vangle = 90, minv = 0.5, ddt = 0.95, dt = 3.0;
double rcopy = 80, rcent = 30, rviso = 40, rvoid = 15;
double wcopy = 0.2, wcent = 0.4, wviso = 0.8, wvoid = 1.0, wrand = 0.0;

char *term = NULL;

char help_string[] = "\
Simulate a flock of boids according to rules that determine their \
individual behaviors as well as the ``physics'' of their universe. \
A boid greedily attempts to apply four rules with respect to its \
neighbors: it wants to fly in the same direction, be in the center \
of the local cluster of boids, avoid collisions with boids too close, \
and maintain a clear view ahead by skirting around others that block \
its view.  Changing these rules can make the boids behave like birds, \
gnats, bees, fish, or magnetic particles.  See the RULES section of \
the manual pages for more details.\
```

```
";

OPTION options[] = {
  { "-width",  OPT_INT,     &width,  "Width of the plot in pixels." },
  { "-height", OPT_INT,     &height, "Height of the plot in pixels." },
  { "-num",    OPT_INT,     &num,    "Number of boids." },
  { "-steps",  OPT_INT,     &steps,  "Number of simulated steps." },
  { "-seed",   OPT_INT,     &seed,   "Random seed for initial state." },
  { "-angle",  OPT_DOUBLE,  &angle,  "Number of viewing degrees." },
  { "-vangle", OPT_DOUBLE,  &vangle, "Visual avoidance angle." },
  { "-rcopy",  OPT_DOUBLE,  &rcopy,  "Radius for copy vector." },
  { "-rcent",  OPT_DOUBLE,  &rcent,  "Radius for centroid vector." },
  { "-rvoid",  OPT_DOUBLE,  &rvoid,  "Radius for avoidance vector." },
  { "-rviso",  OPT_DOUBLE,  &rviso,  "Radius for visual avoidance vector." },
  { "-wcopy",  OPT_DOUBLE,  &wcopy,  "Weight for copy vector." },
  { "-wcent",  OPT_DOUBLE,  &wcent,  "Weight for centroid vector." },
  { "-wvoid",  OPT_DOUBLE,  &wvoid,  "Weight for avoidance vector." },
  { "-wviso",  OPT_DOUBLE,  &wviso,  "Weight for visual avoidance vector." },
  { "-wrand",  OPT_DOUBLE,  &wrand,  "Weight for random vector." },
  { "-dt",     OPT_DOUBLE,  &dt,     "Time-step increment." },
  { "-ddt",    OPT_DOUBLE,  &ddt,    "Momentum factor (0 < ddt < 1)." },
  { "-minv",   OPT_DOUBLE,  &minv,   "Minimum velocity." },
  { "-len",    OPT_INT,     &len,    "Tail length." },
  { "-psdump", OPT_SWITCH,  &psdump, "Dump PS at the very end?" },
  { "-inv",    OPT_SWITCH,  &invert, "Invert all colors?" },
  { "-mag",    OPT_INT,     &mag,    "Magnification factor." },
  { "-term",   OPT_STRING,  &term,   "How to plot points." },
  { NULL,      OPT_NULL,    NULL,    NULL }
};


/* These are global to avoid passing them around all of time.  They
   represent the boids (x, y) positions, velocity vectors, and new
   velocity vectors. */

double *xp, *yp, *xv, *yv, *xnv, *ynv;


/* Some handy macros ... */

#define LEN(x, y) sqrt(SQR(x) + SQR(y))
#define DIST(x1, y1, x2, y2) LEN(((x1)-(x2)),((y1)-(y2)))
#define DOT(x1, y1, x2, y2) ((x1)*(x2)+(y1)*(y2))

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Destructively normalize a vector. */

void norm(double *x, double *y)
{
  double len;

  len = LEN(*x, *y);
  if(len != 0.0) {
    *x /= len;
    *y /= len;
  }
```

```
135 }

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Compute the heading for a particular boid based on its current
140    environment. */

void compute_new_heading(int which)
{
  int i, j, k, numcent = 0;
145   double xa, ya, xb, yb, xc, yc, xd, yd, xt, yt;
  double mindist, mx = 0, my = 0, d;
  double cosangle, cosvangle, costemp;
  double xtemp, ytemp, maxr, u, v;

150   /* This is the maximum distance in which any rule is activated. */
  maxr = MAX(rviso, MAX(rcopy, MAX(rcent, rvoid)));

  /* These two values are used to see if a boid can "see" another
   * boid in various ways.
155    */
  cosangle = cos(angle / 2);
  cosvangle = cos(vangle / 2);

  /* These are the accumulated change vectors for the four rules. */
160   xa = ya = xb = yb = xc = yc = xd = yd = 0;

  /* For every boid... */
  for(i = 0; i < num; i++) {

165     /* Don't include self for computing new heading. */
    if(i == which) continue;

    /* Since we want boids to "see" each other around the borders of
     * the screen, we need to check if a boid on the left edge is
170      * actually "close" to a boid on the right edge, etc.  We do this
     * by searching over nine relative displacements of boid(i) and
     * pick the one that is closest to boid(which).  These coordinates
     * are then used for all remaining calculations.
     */
175     mindist = 10e10;
    for(j = -width; j <= width; j += width)
      for(k = -height; k <= height; k += height) {
        d = DIST(xp[i] + j, yp[i] + k, xp[which], yp[which]);
        if(d < mindist) {
180       mindist = d;
          mx = xp[i] + j;
          my = yp[i] + k;
        }
      }
185
    /* If that distance is farther than any of the rule radii,
     * then skip.
     */
    if(mindist > maxr) continue;
190
    /* Make a vector from boid(which) to boid(i). */
```

```
    xtemp = mx - xp[which]; ytemp = my - yp[which];

    /* Calculate the cosine of the velocity vector of boid(which)
     * and the vector from boid(which) to boid(i).
     */
    costemp = DOT(xv[which], yv[which], xtemp, ytemp) /
      (LEN(xv[which], yv[which]) * LEN(xtemp, ytemp));

    /* If this cosine is less than the cosine of one half
     * of the boid's eyesight, i.e., boid(which) cannot see
     * boid(i), then skip.
     */
    if(costemp < cosangle) continue;

    /* If the distance between the two boids is within the radius
     * of the centering rule, but outside of the radius of the
     * avoidance rule, then attempt to center in on boid(i).
     */
    if(mindist <= rcent && mindist > rvoid) {
      xa += mx - xp[which];
      ya += my - yp[which];
      numcent++;
    }

    /* If we are close enough to copy, but far enough to avoid,
     * then copy boid(i)'s velocity.
     */
    if(mindist <= rcopy && mindist > rvoid) {
      xb += xv[i];
      yb += yv[i];
    }

    /* If we are within collision range, then try to avoid boid(i). */
    if(mindist <= rvoid) {

      /* Calculate the vector which moves boid(which) away from boid(i). */
      xtemp = xp[which] - mx;
      ytemp = yp[which] - my;

      /* Make the length of the avoidance vector inversely proportional
       * to the distance between the two boids.
       */
      d = 1 / LEN(xtemp, ytemp);
      xtemp *= d; ytemp *= d;
      xc += xtemp; yc += ytemp;
    }

    /* If boid(i) is within rviso distance and the angle between this boid's
     * velocity vector and the boid(i)'s position relative to this boid is
     * less than vangle, then try to move so that vision is restored.
     */
    if(mindist <= rviso && cosvangle < costemp) {

      /* Calculate the vector which moves boid(which) away from boid(i). */
      xtemp = xp[which] - mx;
      ytemp = yp[which] - my;
```

```
        /* Calculate another vector that is orthogonal to the previous,
         * But try to make it in the same general direction of boid(which)'s
         * direction of movement.
         */
        u = v = 0;
        if(xtemp != 0 && ytemp != 0) {
          u = sqrt(SQR(ytemp / xtemp) / (1 + SQR(ytemp / xtemp)));
          v = -xtemp * u / ytemp;
        }
        else if(xtemp != 0)
          u = 1;
        else if(ytemp != 0)
          v = 1;
        if((xv[which] * u + yv[which] * v) < 0) {
          u = -u; v = -v;
        }

        /* Add the vector that moves away from boid(i). */
        u = xp[which] - mx + u;
        v = yp[which] - my + v;

        /* Make this vector's length inversely proportional to the
         * distance between the two boids.
         */
        d = LEN(xtemp, ytemp);
        if(d != 0) {
          u /= d; v /= d;
        }
        xd += u; yd += v;
      }
    }

    /* Avoid centering on only one other boid;
     * it makes you look aggressive!
     */
    if(numcent < 2)
      xa = ya = 0;

    /* Normalize all big vectors. */
    if(LEN(xa, ya) > 1.0) norm(&xa, &ya);
    if(LEN(xb, yb) > 1.0) norm(&xb, &yb);
    if(LEN(xc, yc) > 1.0) norm(&xc, &yc);
    if(LEN(xd, yd) > 1.0) norm(&xd, &yd);

    /* Compute the composite trajectory based on all of the rules. */
    xt = xa * wcent + xb * wcopy + xc * wvoid + xd * wviso;
    yt = ya * wcent + yb * wcopy + yc * wvoid + yd * wviso;

    /* Optionally add some noise. */
    if(wrand > 0) {
      xt += random_range(-1, 1) * wrand;
      yt += random_range(-1, 1) * wrand;
    }

    /* Update the velocity and renormalize if it is too small. */
    xnv[which] = xv[which] * ddt + xt * (1 - ddt);
    ynv[which] = yv[which] * ddt + yt * (1 - ddt);
```

```
    d = LEN(xnv[which], ynv[which]);
    if(d < minv) {
      xnv[which] *= minv / d;
      ynv[which] *= minv / d;
310 }
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

315 void draw_boid(int which, int color)
{
  double x1, x2, x3, y1, y2, y3, a, t;

  /* Plot a line in the direction that it is heading. */
320 x3 = xv[which]; y3 = yv[which];
  norm(&x3, &y3);
  x1 = xp[which]; y1 = yp[which];
  x2 = x1 - x3 * len;
  y2 = y1 - y3 * len;
325 plot_line(x1, y1, x2, y2, color);

  /* Plot the head of the boid, with the angle of the arrow head
   * indicating its viewing angle.
   */
330 t = (x1 - x2) / len;
  t = (t < -1) ? -1 : (t > 1) ? 1 : t;
  a = acos(t);
  a = (y1 - y2) < 0 ? -a : a;

335 /* This is for the right portion of the head. */
  x3 = x1 + cos(a + angle / 2) * len / 3.0;
  y3 = y1 + sin(a + angle / 2) * len / 3.0;
  plot_line(x1, y1, x3, y3, color);

340 /* This is for the left portion of the head. */
  x3 = x1 + cos(a - angle / 2) * len / 3.0;
  y3 = y1 + sin(a - angle / 2) * len / 3.0;
  plot_line(x1, y1, x3, y3, color);
}
345
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

int main(int argc, char **argv)
{
350 extern int plot_mag;
  extern int plot_inverse;
  int i, j;

  get_options(argc, argv, options, help_string);
355 if(!psdump) {
    plot_mag = mag;
    plot_inverse = invert;
    plot_init(width, height, 2, term);
    plot_set_all(0);
360 }
  srandom(seed);
```

```
    /* Convert angles to radians. */
    angle = angle * M_PI / 180.0;
365 vangle = vangle * M_PI / 180.0;

    /* Make space for the positions, velocities, and new velocities. */
    xp  = xmalloc(sizeof(double) * num);
    yp  = xmalloc(sizeof(double) * num);
370 xv  = xmalloc(sizeof(double) * num);
    yv  = xmalloc(sizeof(double) * num);
    xnv = xmalloc(sizeof(double) * num);
    ynv = xmalloc(sizeof(double) * num);

375 /* Set to random initial conditions. */
    for(i = 0; i < num; i++) {
      xp[i] = random() % width;
      yp[i] = random() % height;
      xv[i] = random_range(-1.0, 1.0);
380   yv[i] = random_range(-1.0, 1.0);
      norm(&xv[i], &yv[i]);
    }

    /* For each time step... */
385 for(i = 0; i < steps; i++) {

      /* For each boid, compute its new heading. */
      for(j = 0; j < num; j++)
        compute_new_heading(j);
390
      /* For each boid again... */
      for(j = 0; j < num; j++) {

        /* Undraw the boid. */
395     if(!psdump) draw_boid(j, 0);

        /* Update the velocity and position. */
        xv[j] = xnv[j];
        yv[j] = ynv[j];
400     xp[j] += xv[j] * dt;
        yp[j] += yv[j] * dt;

        /* Wrap around the screen coordinates. */
        if(xp[j] < 0) xp[j] += width;
405     else if(xp[j] >= width) xp[j] -= width;
        if(yp[j] < 0) yp[j] += height;
        else if(yp[j] >= height - 1) yp[j] -= height;

        /* Redraw the boid. */
410     if(!psdump) draw_boid(j, 1);
      }
    }
    if(!psdump) plot_finish();

415 /* If we want a PS dump of the final configuration, do it. */
    if(psdump) {
      plot_inverse = 0;
      plot_init(width, height, 2, "ps");
      for(i = 0; i < num; i++) {
```

```
420         draw_boid(i, 0);
          }
          plot_finish();
        }

425     exit(0);
    }

    /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

19 gpu/cmpsh.c

```
    #include <stdlib.h>
    #include <stdio.h>
    #include <GL/glew.h>
    #include <GL/glut.h>
 5
    #include "cmpsh.h"
    #include "cmpsh.frag.c"

    struct cmpsh *cmpsh_init(struct cmpsh *cmpsh) {
10    if (! cmpsh) { return 0; }
      if (! shader_init(&cmpsh->shader, 0, cmpsh_frag)) {
        return 0;
      }
      shader_uniform(cmpsh, tex0);
15    shader_uniform(cmpsh, tex1);
      shader_uniform(cmpsh, tex2);
      shader_uniform(cmpsh, val);
      cmpsh->value.tex0 = 0;
      cmpsh->value.tex1 = 1;
20    cmpsh->value.tex2 = 2;
      cmpsh->value.val.v[0] = 0;
      cmpsh->value.val.v[1] = 0;
      cmpsh->value.val.v[2] = 0;
      cmpsh->value.val.v[3] = 1;
25    cmpsh->width  = config_tex_width;
      cmpsh->height = config_tex_height;
      cmpsh->which = 0;
      glGenTextures(2, &cmpsh->tex[0]);
      glEnable(GL_TEXTURE_2D);
30    float *zero = calloc(1, sizeof(float) * 4 * cmpsh->width * cmpsh->height);
      for (int i = 0; i < 2; ++i) {
        glBindTexture(GL_TEXTURE_2D, cmpsh->tex[i]);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
35      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA32F, cmpsh->width, cmpsh->height, 0, ↙
            ↳ GL_RGBA, GL_FLOAT, zero);
      }
      free(zero);
40    glBindTexture(GL_TEXTURE_2D, 0);
      glDisable(GL_TEXTURE_2D);
      return cmpsh;
    }
```

```
45  void cmpsh_reset(struct cmpsh *cmpsh) {
      float *one = calloc(1, sizeof(float) * 4 * cmpsh->width * cmpsh->height);
      for (int i = 0; i < 4 * cmpsh->width * cmpsh->height; ++i) {
        one[i] = 1.0;
      }
50    glBindTexture(GL_TEXTURE_2D, cmpsh->tex[0]);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA32F, cmpsh->width, cmpsh->height, 0, ↙
          ↳ GL_RGBA, GL_FLOAT, one);
      glBindTexture(GL_TEXTURE_2D, cmpsh->tex[1]);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA32F, cmpsh->width, cmpsh->height, 0, ↙
          ↳ GL_RGBA, GL_FLOAT, one);
      glBindTexture(GL_TEXTURE_2D, 0);
55    free(one);
    }

    GLuint cmpsh_display(struct cmpsh *cmpsh, GLuint fbo, GLuint tex1, GLuint tex2, ↙
        ↳ struct vec4 *val) {
      glMatrixMode(GL_PROJECTION);
60    glLoadIdentity();
      gluOrtho2D(0, 1, 0, 1);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      glViewport(0, 0, cmpsh->width, cmpsh->height);
65    glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fbo);
      glFramebufferTexture2DEXT(
          GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT,
          GL_TEXTURE_2D, cmpsh->tex[1 - cmpsh->which], 0
      );
70    glActiveTexture(GL_TEXTURE2);
      glBindTexture(GL_TEXTURE_2D, tex2);
      glActiveTexture(GL_TEXTURE1);
      glBindTexture(GL_TEXTURE_2D, tex1);
      glActiveTexture(GL_TEXTURE0);
75    glBindTexture(GL_TEXTURE_2D, cmpsh->tex[cmpsh->which]);
      glUseProgramObjectARB(cmpsh->shader.program);
      shader_updatei(cmpsh, tex0);
      shader_updatei(cmpsh, tex1);
      shader_updatei(cmpsh, tex2);
80    cmpsh->value.val.v[0] = val->v[0];
      cmpsh->value.val.v[1] = val->v[1];
      cmpsh->value.val.v[2] = val->v[2];
      cmpsh->value.val.v[3] = val->v[3];
      shader_updatef4(cmpsh, val);
85    glBegin(GL_QUADS); { glColor4f(1,1,1,1);
        glTexCoord2f(0, 0); glVertex2f(0, 0);
        glTexCoord2f(1, 0); glVertex2f(1, 0);
        glTexCoord2f(1, 1); glVertex2f(1, 1);
        glTexCoord2f(0, 1); glVertex2f(0, 1);
90    } glEnd();
      glUseProgramObjectARB(0);
      glActiveTexture(GL_TEXTURE2);
      glBindTexture(GL_TEXTURE_2D, 0);
      glActiveTexture(GL_TEXTURE1);
95    glBindTexture(GL_TEXTURE_2D, 0);
      glActiveTexture(GL_TEXTURE0);
      glBindTexture(GL_TEXTURE_2D, 0);
      glFramebufferTexture2DEXT(
```

```
        GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT,
100     GL_TEXTURE_2D, 0, 0
      );
      glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
      cmpsh->which = 1 - cmpsh->which;
      return cmpsh->tex[cmpsh->which];
105   }
```

20 gpu/cmpsh.frag

```
#version 120
precision highp float;

uniform sampler2D tex0;
 5  uniform sampler2D tex1;
uniform sampler2D tex2;
uniform vec4 val;

void main() {
10    vec4 v0 = texture2D(tex0, gl_TexCoord[0].xy).rgba;
      vec2 p1 = texture2D(tex1, gl_TexCoord[0].xy).xy;
      vec2 p2 = texture2D(tex2, gl_TexCoord[0].xy).xy;
      vec2 d = p1 - p2;
      vec4 v = v0;
15    if (dot(d,d) < v.w) {
        v = vec4(val.xyz, dot(d,d));
      }
      gl_FragData[0] = v;
}
```

21 gpu/cmpsh.h

```
#ifndef CMPSH_H
#define CMPSH_H 1

#include "config.h"
 5  #include "shader.h"

struct cmpsh { struct shader shader;
    struct { GLint tex0, tex1, tex2, val; } uniform;
    struct { int tex0, tex1, tex2; struct vec4 val; } value;
10  GLuint tex[2];
    int which;
    int width;
    int height;
};
15
struct cmpsh *cmpsh_init(struct cmpsh *cmpsh);
void cmpsh_reset(struct cmpsh *cmpsh);
GLuint cmpsh_display(struct cmpsh *cmpsh, GLuint fbo, GLuint tex1, GLuint tex2, ↙
    ↳ struct vec4 *val);

20  #endif
```

22 gpu/colour.c

```
/* ============================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
----------------------------------------------------------------------------
False Colour Shader
============================================================================ */

#include <stdio.h>
#include <GL/glew.h>
#include <GL/glut.h>

#include "config.h"
#include "util.h"
#include "colour.h"
#include "colour.frag.c"

//============================================================================
// false colour shader initialization
struct colour *colour_init(struct colour *colour) {
  if (! colour) { return 0; }
  if (! shader_init(&colour->shader, 0, colour_frag)) {
    return 0;
  }
  shader_uniform(colour, statistic);
  shader_uniform(colour, sr);
  colour->value.statistic = 0;
  colour->value.sr = config_samplerate;
  glGenTextures(1, &colour->texture);
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, colour->texture);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
  glDisable(GL_TEXTURE_2D);
  colour->width  = 0;
  colour->height = 0;
  return colour;
}

//============================================================================
// false colour shader reshape callback
void colour_reshape(struct colour *colour, int w, int h) {
  colour->width  = roundtwo(w);
  colour->height = roundtwo(h);
  glEnable(GL_TEXTURE_2D);
```

```
  glBindTexture(GL_TEXTURE_2D, colour->texture);
  glTexImage2D(
    GL_TEXTURE_2D, 0, GL_RGBA32F_ARB, colour->width, colour->height,
    0, GL_RGBA, GL_FLOAT, 0
  );
  glDisable(GL_TEXTURE_2D);
}

//=============================================================================
// false colour shader display callback
void colour_display(struct colour *colour, GLuint fbo, GLuint texture) {
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, texture);
  glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fbo);
  glFramebufferTexture2DEXT(
    GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT, GL_TEXTURE_2D,
    colour->texture, 0
  );
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(0, 1, 0, 1);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glViewport(0, 0, colour->width, colour->height);
  glUseProgramObjectARB(colour->shader.program);
  shader_updatei(colour, statistic);
  shader_updatef(colour, sr);
  glBegin(GL_QUADS); { glColor4f(1,1,1,1);
    glTexCoord2f(0, 0); glVertex2f(0, 0);
    glTexCoord2f(1, 0); glVertex2f(1, 0);
    glTexCoord2f(1, 1); glVertex2f(1, 1);
    glTexCoord2f(0, 1); glVertex2f(0, 1);
  } glEnd();
  glUseProgramObjectARB(0);
  glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
  glBindTexture(GL_TEXTURE_2D, 0);
  glDisable(GL_TEXTURE_2D);
}

// EOF
```

23 gpu/colour.frag

```
#version 120
precision highp float;

uniform sampler2D statistic;
uniform float sr;

float ftoo(float f) {
  return log2(f);
}

void main() {
  mat3 yuv2rgb = mat3(1.0, 1.407, 0.0, 1.0, -0.677, -0.236, 1.0, 0.0, 1.848);
  vec4 s = texture2D(statistic, gl_TexCoord[0].xy);
  float mean = s.y / s.x;
```

```
    float stddev = sqrt(s.x * s.z - s.y * s.y) / s.x;
    if (! (stddev > 0.0)) {
      stddev = 0.0;
    }
    float hue = 6.283185307179586 * mean / 12.0;
    float value = max(0.0, 0.5 - stddev / 48.0);
    float satur = 0.25 * value; //clamp(0.25 * log2(mean / harm), value / 16.0, ↙
        ↳ value / 2.0);
    vec3 yuv = vec3(value, satur * sin(hue), satur * cos(hue));
    vec3 rgb = yuv * yuv2rgb;
    gl_FragData[0] = vec4(clamp(rgb, 0.0, 1.0), 1.0);
}
```

24 gpu/colour.h

```
/* ============================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
----------------------------------------------------------------------------
False Colour Shader
============================================================================ */

#ifndef COLOUR_H
#define COLOUR_H 1

#include "shader.h"

//============================================================================
// false colour shader data
struct colour { struct shader shader;
  struct { GLint  statistic; GLint sr; } uniform;
  struct { int    statistic; float sr; } value;
  GLuint texture;
  int width;
  int height;
};

//============================================================================
// protoypes
struct colour *colour_init(struct colour *colour);
void colour_reshape(struct colour *colour, int w, int h);
void colour_display(struct colour *colour, GLuint fbo, GLuint texture);

#endif
```

25 gpu/config.h

```
/* ========================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
------------------------------------------------------------------------
Configuration
======================================================================== */

#ifndef CONFIG_H
#define CONFIG_H 1

// ========================================================================
// expert settings
#define config_aspect          1
#define config_tex_width     256
#define config_tex_height    256
#define config_win_width    1024
#define config_win_height    512
#define config_scale_factor    1
#define config_samplerate  48000
#define config_oscillators     5
#define config_beats           1
#define config_bpm            60
#define config_overdrive       8
#define config_blocksize      32

#endif
```

26 gpu/.gitignore

```
soft-rock-revisited
*.o
*.frag.c
```

27 gpu/hsv.c

```
#include <math.h>
#include "util.h"
#include "hsv.h"

void hsv2rgb(struct vec3 *hsv, struct vec3 *rgb) {
  float r=0, g=0, b=0;
```

```
  float h = clamp(hsv->v[0], 0.0, 1.0);
  float s = clamp(hsv->v[1], 0.0, 1.0);
  float v = clamp(hsv->v[2], 0.0, 1.0);
  h *= 360.f; // convert hue to degrees
  if (s == 0.0) { // black and white
    r = g = b = v;
  } else {
    if (h == 360.0) { h = 0; } // 360 == 0 degrees
    h /= 60.0f; // hue is now [0, 6]
    {
      int i = (int)floor(h);
      float f = h - i; // f is the fractional part of h
      float p = v * (1 - s);
      float q = v * (1 - s * f);
      float t = v * (1 - s * (1 - f));
      switch (i) {
      case 0: r = v; g = t; b = p; break;
      case 1: r = q; g = v; b = p; break;
      case 2: r = p; g = v; b = t; break;
      case 3: r = p; g = q; b = v; break;
      case 4: r = t; g = p; b = v; break;
      case 5: r = v; g = p; b = q; break;
      }
    }
  }
  rgb->v[0] = r;
  rgb->v[1] = g;
  rgb->v[2] = b;
}
```

28 gpu/hsv.h

```
#ifndef HSV_H
#define HSV_H 1

#include "shader.h"

void hsv2rgb(struct vec3 *hsv, struct vec3 *rgb);

#endif
```

29 gpu/main.c

```
/* ==========================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
```

```
15  You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
    ------------------------------------------------------------------------
    Main Program Entry Point
    ======================================================================== */
20
    #include <stdio.h>
    #include <GL/glew.h>
    #include <GL/glut.h>

25  #include "revisited.h"
    #include "config.h"

    int main(int argc, char **argv) {
      /* initialize GLUT etc */
30    fprintf(stderr, "soft-rock-revisited (GPL) 2012 Claude Heiland-Allen <↙
          ↪ claude@mathr.co.uk>\n");
      glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
      glutInitWindowSize(config_win_width, config_win_height);
      glutInit(&argc, argv);
      glutCreateWindow("srrv");
35    glewInit();
      glClampColor(GL_CLAMP_FRAGMENT_COLOR, GL_FALSE);
      glClampColor(GL_CLAMP_READ_COLOR, GL_FALSE);
      /* activate callbacks and enter main loop */
      if (revisited_init()) {
40      glutDisplayFunc(revisited_display);
        glutReshapeFunc(revisited_reshape);
        glutIdleFunc(revisited_idle);
        glutKeyboardFunc(revisited_keynormal);
        glutSpecialFunc(revisited_keyspecial);
45      glutMouseFunc(revisited_mouse);
        glutMotionFunc(revisited_motion);
        glutPassiveMotionFunc(revisited_motion);
        glutSetCursor(GLUT_CURSOR_CROSSHAIR);
        glutMainLoop();
50      return 0; // never reached
      } else {
        return 1;
      }
    }
```

30 gpu/Makefile

```
    #============================================================================
    # kitchen
    # Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>
    #----------------------------------------------------------------------------
 5  # Makefile
    #============================================================================

    CC = gcc
    CFLAGS = -std=c99 -Wall -pedantic -Wextra -Wno-unused-parameter -O3 -march=↙
        ↪ native -pthread `pkg-config --cflags jack`
10  LIBS = -lGLEW -lGL -lGLU -lglut `pkg-config --libs jack`
    OBJS = main.o revisited.o shader.o colour.o output.o map.o util.o audio.o minmax↙
```

```
    ↪ .o stretch.o cmpsh.o hsv.o
GENC = map.frag.c colour.frag.c minmax.frag.c stretch.frag.c cmpsh.frag.c

# optional feature: debuggability
#CFLAGS += -ggdb


all: soft-rock-revisited

clean:
	-rm -f $(OBJS) $(GENC)

# no suffix rules, they cause weird issues with the .frag.c files
.SUFFIXES:
.PHONY: all clean

# link program
soft-rock-revisited: $(OBJS)
	$(CC) $(CFLAGS) -o soft-rock-revisited $(OBJS) $(LIBS)

# C source to object file
%.o: %.c
	$(CC) $(CFLAGS) -o $@ -c $<

# shader source to C source
%.frag.c: %.frag s2c.sh
	./s2c.sh $*_frag < $< > $@

# dependencies
audio.o: audio.c audio.h map.h config.h
cmpsh.o: cmpsh.c cmpsh.frag.c cmpsh.h shader.h config.h hsv.h
colour.o: colour.c colour.frag.c colour.h shader.h util.h config.h
hsv.o: hsv.c hsv.h shader.h
output.o: output.c output.h minmax.h stretch.h config.h
main.o: main.c revisited.h config.h
map.o: map.c map.frag.c map.h cmpsh.h shader.h util.h config.h
minmax.o: minmax.c minmax.frag.c minmax.h shader.h util.h config.h
revisited.o: revisited.c revisited.h map.h colour.h output.h config.h
shader.o: shader.c shader.h
stretch.o: stretch.c stretch.frag.c stretch.h shader.h
util.o: util.c util.h
```

31 gpu/map.c

```
/* ===========================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
```

```
15  You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
    ------------------------------------------------------------------------
    FM-Oscillator Map Shader
    ======================================================================== */
20
    #include <stdio.h>
    #include <stdlib.h>
    #include <math.h>
    #include <GL/glew.h>
25  #include <GL/glut.h>

    #include "config.h"
    #include "audio.h"
    #include "util.h"
30  #include "hsv.h"
    #include "map.h"
    #include "map.frag.c"

    void map_reset(struct map *map) {
35    float *zero = calloc(1, sizeof(float) * 4 * map->width * map->height);
      glEnable(GL_TEXTURE_2D);
      for (int j = 0; j < 2; ++j) {
        map->samples[j] = 0;
        map->phase_buffer[j] = 0;
40      for (int i = 0; i < config_blocksize + 1; ++i) {
          glBindTexture(GL_TEXTURE_2D, map->phase_textures[j][i]);
          glTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, map->width, map->height, GL_RGBA, ↙
              ↳ GL_FLOAT, zero);
        }
      }
45    glBindTexture(GL_TEXTURE_2D, 0);
      glDisable(GL_TEXTURE_2D);
      free(zero);
      cmpsh_reset(&map->cmpsh);
    }
50
    struct map *map_init(struct map *map) {
      if (! map) { return 0; }
      if (! cmpsh_init(&map->cmpsh)) { return 0; }
      if (! shader_init(&map->shader, 0, map_frag)) {
55      return 0;
      }
      shader_uniform(map, phases_cur);
      shader_uniform(map, phases_old);
      shader_uniform(map, sr);
60    shader_uniform(map, note_lo);
      shader_uniform(map, note_hi);
      shader_uniform(map, modi_lo);
      shader_uniform(map, modi_hi);
      map->value.phases_cur = 0;
65    map->value.phases_old = 1;
      map->value.sr = config_samplerate;
      map->note   = 29.981804;
      map->modi   = 52.463516;
      map->nrange = 32.0;
```

```
70    map->mrange = 32.0;
      map->width  = config_tex_width;
      map->height = config_tex_height;

      glGenTextures(config_blocksize + 1, map->phase_textures[0]);
75    glGenTextures(config_blocksize + 1, map->phase_textures[1]);
      glEnable(GL_TEXTURE_2D);
      float *zero = calloc(1, sizeof(float) * 4 * map->width * map->height);
      for (int j = 0; j < 2; ++j) {
        for (int i = 0; i < config_blocksize + 1; ++i) {
80        glBindTexture(GL_TEXTURE_2D, map->phase_textures[j][i]);
          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
85        glTexImage2D(GL_TEXTURE_2D, 0, GL_RG32F, map->width, map->height, 0, ↙
              ↳ GL_RGBA, GL_FLOAT, zero);
        }
      }
      free(zero);
      glBindTexture(GL_TEXTURE_2D, 0);
90    glDisable(GL_TEXTURE_2D);

      map_reset(map);
      return map;
    }
95
    void map_reshape(struct map *map, int w, int h, int winw, int winh) {
      map->width  = roundtwo(w);
      map->height = roundtwo(h);
      map->win_width = winw;
100   map->win_height = winh;
      map_reset(map);
    }

    GLuint map_display(struct map *map, GLuint fbo) {
105   GLuint retval = 0;
      for (int i = 0; i < config_overdrive; ++i) {
        int settled = 1;//map->samples[0] > 48000;
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
110     gluOrtho2D(0, 1, 0, 1);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glViewport(0, 0, map->width, map->height);
        glUseProgramObjectARB(map->shader.program);
115     shader_updatei(map, phases_cur);
        shader_updatei(map, phases_old);
        shader_updatef(map, sr);
        map->value.note_lo = map->note - config_aspect * map->nrange;
        map->value.note_hi = map->note + config_aspect * map->nrange;
120     map->value.modi_lo = map->modi - map->mrange;
        map->value.modi_hi = map->modi + map->mrange;
        shader_updatef(map, note_lo);
        shader_updatef(map, note_hi);
        shader_updatef(map, modi_lo);
125     shader_updatef(map, modi_hi);
```

```
    glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fbo);
    for (int j = 0; j < 2; ++j) {
      for (int k = 0; k <= j * settled; ++k) {
        glActiveTexture(GL_TEXTURE1);
130     glBindTexture(GL_TEXTURE_2D, map->phase_textures[j][(map->phase_buffer[j↙
            ↳ ] + 2) % (config_blocksize + 1)]);
        glActiveTexture(GL_TEXTURE0);
        glBindTexture(GL_TEXTURE_2D, map->phase_textures[j][map->phase_buffer[j↙
            ↳ ]]);
        glFramebufferTexture2DEXT(
          GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT,
135       GL_TEXTURE_2D, map->phase_textures[j][(map->phase_buffer[j] + 1) % (↙
              ↳ config_blocksize + 1)], 0
        );
        glBegin(GL_QUADS); { glColor4f(1,1,1,1);
          glTexCoord2f(0, 0); glVertex2f(0, 0);
          glTexCoord2f(1, 0); glVertex2f(1, 0);
140       glTexCoord2f(1, 1); glVertex2f(1, 1);
          glTexCoord2f(0, 1); glVertex2f(0, 1);
        } glEnd();
        map->phase_buffer[j] = (map->phase_buffer[j] + 1) % (config_blocksize + ↙
            ↳ 1);
        map->samples[j] += 1;
145     glFramebufferTexture2DEXT(
          GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT,
          GL_TEXTURE_2D, 0, 0
        );
        glActiveTexture(GL_TEXTURE1);
150     glBindTexture(GL_TEXTURE_2D, 0);
        glActiveTexture(GL_TEXTURE0);
        glBindTexture(GL_TEXTURE_2D, 0);
      }
    }
155 glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
    glUseProgramObjectARB(0);
    if (settled) {
      float o = map->samples[1] - map->samples[0];
      struct vec3 hsv;
160   hsv.v[0] = (sqrt(5.0) - 1.0)/2.0 * o / 480;
      hsv.v[1] = 1.0;
      hsv.v[2] = 1.0;//clamp(o / 16.0, 0.0, 1.0);
      struct vec3 rgb;
      hsv2rgb(&hsv, &rgb);
165   struct vec4 rgba;
      rgba.v[0] = rgb.v[0];
      rgba.v[1] = rgb.v[1];
      rgba.v[2] = rgb.v[2];
      rgba.v[3] = 1.0;
170   retval = cmpsh_display(&map->cmpsh, fbo,
        map->phase_textures[0][map->phase_buffer[0]],
        map->phase_textures[1][map->phase_buffer[1]],
        &rgba
      );
175   }
    }
    return retval;
  }
```

```
180 int map_keyspecial(struct map *map, int key, int x, int y) {
      switch (key) {
        case GLUT_KEY_UP:     map->modi += map->mrange * 0.1; break;
        case GLUT_KEY_DOWN:   map->modi -= map->mrange * 0.1; break;
        case GLUT_KEY_LEFT:   map->note -= map->nrange * 0.1; break;
185     case GLUT_KEY_RIGHT: map->note += map->nrange * 0.1; break;
        case GLUT_KEY_PAGE_UP:     map->mrange *= 0.5; break;
        case GLUT_KEY_PAGE_DOWN:   map->mrange /= 0.5; break;
        case GLUT_KEY_HOME:        map->nrange *= 0.5; break;
        case GLUT_KEY_END:         map->nrange /= 0.5; break;
190     default: return 0;
      }
      map_reset(map);
      return 1;
    }
195
    int map_keynormal(struct map *map, int key, int x, int y) {
      switch (key) {
        case 's':
          map->nrange *= 0.9726549474122855;
200       map->mrange *= 0.9726549474122855;
          system("import -window srrv \"srrv-`date --iso=s | tr \":\" \"-\"`.png\"")↙
              ↪ ;
          map_reset(map);
        case ' ': fprintf(stderr, "%f %f @ %f %f\n", map->note, map->modi, map->↙
            ↪ nrange, map->mrange); break;
        default: return 0;
205   }
      return 1;
    }

    int map_mouse(struct map *map, int button, int state, int x, int y) {
210   if (state == GLUT_DOWN) {
        float u =       (float) x       / (float) map->win_width;
        float v = 1.0 - (float) (y + 1) / (float) map->win_height;
        float note
          =     (map->note - config_aspect * map->nrange) * (1 - u)
215       + u * (map->note + config_aspect * map->nrange);
        float modi = (map->modi - map->mrange) * (1 - v) + v * (map->modi + map->↙
            ↪ mrange);
        float factor = 1.0;
        switch (button) {
          case GLUT_MIDDLE_BUTTON:
220         map->note = note;
            map->modi = modi;
            map_reset(map);
            return 1;
          case GLUT_LEFT_BUTTON:
225       case 3:
            factor = sqrt(0.5);
            break;
          case GLUT_RIGHT_BUTTON:
          case 4:
230         factor = sqrt(2.0);
            break;
          default:
```

```
        return 0;
      }
      map->note = note + (map->note - note) * factor;
      map->modi = modi + (map->modi - modi) * factor;
      map->nrange *= factor;
      map->mrange *= factor;
      map_reset(map);
      fprintf(stderr, "\t%f\t%f\n", map->note, map->modi);
      return 1;
    }
    return 0;
  }

  int map_motion(struct map *map, int x, int y) {
    if (x < 0 || map->win_width <= x || y < 0 || map->win_height <= y) {
      return 0;
    } else {
      float u =       (float) x       / (float) map->win_width;
      float v = 1.0 - (float) (y + 1) / (float) map->win_height;
      float note
        =     (map->note - config_aspect * map->nrange) * (1 - u)
        + u * (map->note + config_aspect * map->nrange);
      float modi = (map->modi - map->mrange) * (1 - v) + v * (map->modi + map->mrange);
      map->note_mouse = note;
      map->modi_mouse = modi;
      fprintf(stderr, "\t%f\t%f\r", map->note_mouse, map->modi_mouse);
      return 1;
    }
  }

  // EOF
```

32 gpu/map.frag

```
  #version 120
  precision highp float;

  uniform sampler2D phases_cur;
  uniform sampler2D phases_old;
  uniform float sr;
  uniform float note_lo;
  uniform float note_hi;
  uniform float modi_lo;
  uniform float modi_hi;

  float mtof(float f) {
    return 8.17579891564 * exp(0.0577622650 * f);
  }

  float ftom(float f) {
    return log(f / 8.17579891564) / 0.0577622650;
  }

  vec2 mtof(vec2 v) {
    return vec2(mtof(v.x), mtof(v.y));
  }
```

```
  void main() {
    vec2 param = gl_TexCoord[0].xy;
    float note = mix(note_lo, note_hi, param.x);
    float modi = mix(modi_lo, modi_hi, param.y);
    vec2 phase_cur = texture2D(phases_cur, gl_TexCoord[0].xy).xy;
    vec2 phase_old = texture2D(phases_old, gl_TexCoord[0].xy).xy;
    vec2 p = phase_cur + mtof(note + modi * cos(6.283185307179586 * phase_old.yx))
        ↳ / sr;
    vec2 phase_out = p - floor(p);
    gl_FragData[0] = vec4(phase_out, vec2(0.0));
  }
```

33 gpu/map.h

```
  /* ============================================================================
  soft-rock-revisited
  Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

  This program is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <http://www.gnu.org/licenses/>.
  ----------------------------------------------------------------------------
  FM-Oscillator Map Shader
  ============================================================================ */

  #ifndef MAP_H
  #define MAP_H 1

  #include "config.h"
  #include "shader.h"
  #include "cmpsh.h"

  //============================================================================
  // map shader data
  struct map { struct shader shader;
    struct { GLint phases_cur, phases_old; GLint sr, note_lo, note_hi, modi_lo,
        ↳ modi_hi; } uniform;
    struct { int   phases_cur, phases_old; float sr, note_lo, note_hi, modi_lo,
        ↳ modi_hi; } value;
    GLuint phase_textures[2][config_blocksize + 1];
    int phase_buffer[2];
    int width;
    int height;
    int win_width;
    int win_height;
    int frame;
    float note;
```

```
  float modi;
  float nrange;
  float mrange;
  float note_mouse;
  float modi_mouse;
  int samples[2];
  struct cmpsh cmpsh;
};

//=========================================================
// prototypes
void map_reset(struct map *map);
struct map *map_init(struct map *map);
void map_reshape(struct map *map, int w, int h, int winw, int winh);
GLuint map_display(struct map *map, GLuint fbo);
int map_keyspecial(struct map *map, int key, int x, int y);
int map_keynormal(struct map *map, int key, int x, int y);
int map_mouse(struct map *map, int button, int state, int x, int y);
int map_motion(struct map *map, int x, int y);

#endif
```

34 gpu/minmax.c

```
/* =========================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
-----------------------------------------------------------------
Min/Max Shader
========================================================= */

#include <stdio.h>
#include <GL/glew.h>
#include <GL/glut.h>

#include "config.h"
#include "util.h"
#include "minmax.h"
#include "minmax.frag.c"

//=========================================================
// min/max shader initialization
struct minmax *minmax_init(struct minmax *minmax) {
  if (! minmax) { return 0; }
```

```
  if (! shader_init(&minmax->shader, 0, minmax_frag)) {
    return 0;
  }
  shader_uniform(minmax, tmin);
  shader_uniform(minmax, tmax);
  shader_uniform(minmax, dx);
  shader_uniform(minmax, dy);
  minmax->value.tmin = 0;
  minmax->value.tmax = 1;
  minmax->value.dx = 0;
  minmax->value.dy = 0;
  minmax->count  = 0;
  minmax->width  = 0;
  minmax->height = 0;
  minmax_reshape(minmax, config_tex_width, config_tex_height);
  return minmax;
}

//=========================================================
// false minmax shader reshape callback
void minmax_reshape(struct minmax *minmax, int w, int h) {
  minmax->width  = roundtwo(w);
  minmax->height = roundtwo(h);
  int oldcount = minmax->count;
  minmax->count = logtwo(roundtwo(w > h ? w : h));
  if (minmax->count != oldcount) {
    if (oldcount > 0) {
      glDeleteTextures(oldcount, &minmax->tmins[0]);
      glDeleteTextures(oldcount, &minmax->tmaxs[0]);
    }
    glGenTextures(minmax->count, &minmax->tmins[0]);
    glGenTextures(minmax->count, &minmax->tmaxs[0]);
    glEnable(GL_TEXTURE_2D);
    for (int i = 0; i < minmax->count; ++i) {
      glBindTexture(GL_TEXTURE_2D, minmax->tmins[i]);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA32F_ARB, 1 << i, 1 << i, 0, GL_RGBA,↙
          ↳ GL_FLOAT, 0);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
      glBindTexture(GL_TEXTURE_2D, minmax->tmaxs[i]);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA32F_ARB, 1 << i, 1 << i, 0, GL_RGBA,↙
          ↳ GL_FLOAT, 0);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    }
    glBindTexture(GL_TEXTURE_2D, 0);
    glDisable(GL_TEXTURE_2D);
  }
}

//=========================================================
// min/max shader display callback
void minmax_display(struct minmax *minmax, GLuint fbo, GLuint texture) {
  glEnable(GL_TEXTURE_2D);
  glActiveTexture(GL_TEXTURE1);
  glBindTexture(GL_TEXTURE_2D, texture);
  glActiveTexture(GL_TEXTURE0);
```

```
  glBindTexture(GL_TEXTURE_2D, texture);
  glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fbo);
  GLuint buffers[2] = { GL_COLOR_ATTACHMENT0, GL_COLOR_ATTACHMENT1 };
  glDrawBuffers(2, &buffers[0]);
  glUseProgramObjectARB(minmax->shader.program);
  for (int i = minmax->count - 1; i >= 0; --i) {
    glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT, ↙
        ↳ GL_TEXTURE_2D, minmax->tmins[i], 0);
    glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT1_EXT, ↙
        ↳ GL_TEXTURE_2D, minmax->tmaxs[i], 0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 1, 0, 1);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glViewport(0, 0, 1 << i, 1 << i);
    minmax->value.dx = 1.0 / (1 << i);
    minmax->value.dy = 1.0 / (1 << i);
    shader_updatei(minmax, tmin);
    shader_updatei(minmax, tmax);
    shader_updatef(minmax, dx);
    shader_updatef(minmax, dy);
    glBegin(GL_QUADS); { glColor4f(1,1,1,1);
      glTexCoord2f(0, 0); glVertex2f(0, 0);
      glTexCoord2f(1, 0); glVertex2f(1, 0);
      glTexCoord2f(1, 1); glVertex2f(1, 1);
      glTexCoord2f(0, 1); glVertex2f(0, 1);
    } glEnd();
    glActiveTexture(GL_TEXTURE1);
    glBindTexture(GL_TEXTURE_2D, minmax->tmaxs[i]);
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, minmax->tmins[i]);
  }
  glDrawBuffers(1, &buffers[0]);
  glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT, ↙
      ↳ GL_TEXTURE_2D, 0, 0);
  glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT1_EXT, ↙
      ↳ GL_TEXTURE_2D, 0, 0);
  glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
  glUseProgramObjectARB(0);
  glActiveTexture(GL_TEXTURE1);
  glBindTexture(GL_TEXTURE_2D, 0);
  glActiveTexture(GL_TEXTURE0);
  glBindTexture(GL_TEXTURE_2D, minmax->tmaxs[0]);
  glGetTexImage(GL_TEXTURE_2D, 0, GL_RGBA, GL_FLOAT, &minmax->maxpixel[0]);
  glBindTexture(GL_TEXTURE_2D, minmax->tmins[0]);
  glGetTexImage(GL_TEXTURE_2D, 0, GL_RGBA, GL_FLOAT, &minmax->minpixel[0]);
  glBindTexture(GL_TEXTURE_2D, 0);
  glDisable(GL_TEXTURE_2D);
}

//============================================================================
// min/max shader idle callback
void minmax_idle(struct minmax *minmax) { }

// EOF
```

35 gpu/minmax.frag

```
#version 120
precision highp float;

uniform sampler2D tmin;
uniform sampler2D tmax;
uniform float dx;
uniform float dy;

void main() {
  vec4 min0 = texture2D(tmin, gl_TexCoord[0].xy + vec2(0.0, 0.0));
  vec4 min1 = texture2D(tmin, gl_TexCoord[0].xy + vec2(dx,  0.0));
  vec4 min2 = texture2D(tmin, gl_TexCoord[0].xy + vec2(0.0, dy ));
  vec4 min3 = texture2D(tmin, gl_TexCoord[0].xy + vec2(dx,  dy ));
  vec4 max0 = texture2D(tmax, gl_TexCoord[0].xy + vec2(0.0, 0.0));
  vec4 max1 = texture2D(tmax, gl_TexCoord[0].xy + vec2(dx,  0.0));
  vec4 max2 = texture2D(tmax, gl_TexCoord[0].xy + vec2(0.0, dy ));
  vec4 max3 = texture2D(tmax, gl_TexCoord[0].xy + vec2(dx,  dy ));
  gl_FragData[0] = min(min(min0, min1), min(min2, min3));
  gl_FragData[1] = max(max(max0, max1), max(max2, max3));
}
```

36 gpu/minmax.h

```
/* ============================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
----------------------------------------------------------------------------
Min/Max Shader
============================================================================ */

#ifndef MINMAX_H
#define MINMAX_H 1

#include "shader.h"

//============================================================================
// min/max shader data
struct minmax { struct shader shader;
  struct { GLint tmin, tmax; GLint dx, dy; } uniform;
  struct { int   tmin, tmax; float dx, dy; } value;
  GLuint tmins[16];
```

```
  GLuint tmaxs[16];
  int count;
  int width;
  int height;
  float minpixel[4];
  float maxpixel[4];
};

//=========================================================
// protoypes
struct minmax *minmax_init(struct minmax *minmax);
void minmax_reshape(struct minmax *minmax, int w, int h);
void minmax_display(struct minmax *minmax, GLuint fbo, GLuint texture);
void minmax_idle(struct minmax *minmax);

#endif
```

37 gpu/output.c

```
/* =========================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
-----------------------------------------------------------------
Output Module
========================================================= */

#include <math.h>
#include <stdio.h>

#include "config.h"
#include "output.h"

struct output *output_init(struct output *output) {
  if (!output) { return 0; }
  if (! minmax_init(&output->minmax)) { return 0; }
  if (! stretch_init(&output->stretch)) { return 0; }
  output->frame = 0;
  output->width = 1;
  output->height = 1;
  output->winwidth = 1;
  output->winheight = 1;
  glGenTextures(1, &output->texture);
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, output->texture);
```

```
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, config_win_width, config_win_height, 0,↙
      ↳ GL_RGB, GL_UNSIGNED_BYTE, 0); // FIXME
  glDisable(GL_TEXTURE_2D);
  return output;
}

void output_reshape(struct output *output, int ww, int hh, int w, int h) {
  minmax_reshape(&output->minmax, ww, hh);
  stretch_reshape(&output->stretch, ww, hh);
  output->width = ww;
  output->height = hh;
  output->winwidth = w;
  output->winheight = h;
}

void output_display(struct output *output, GLuint texture, GLuint fbo) {
  minmax_display(&output->minmax, fbo, texture);
  stretch_display(&output->stretch, fbo, texture, &output->minmax.minpixel[0], &↙
      ↳ output->minmax.maxpixel[0]);
  glClearColor(0,0,0,1);
  glClear(GL_COLOR_BUFFER_BIT);
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, output->stretch.texture);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(0, 1, 0, 1);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glViewport(0, 0, output->winwidth, output->winheight);
  glBegin(GL_QUADS); {
    glColor4f(1,1,1,1);
    glTexCoord2f(0, 0); glVertex2f(0, 0);
    glTexCoord2f(1, 0); glVertex2f(1, 0);
    glTexCoord2f(1, 1); glVertex2f(1, 1);
    glTexCoord2f(0, 1); glVertex2f(0, 1);
  } glEnd();
  glDisable(GL_TEXTURE_2D);

  // FIXME
  output->frame += 1;
}

// EOF
```

38 gpu/output.h

```
/* ==========================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

15  You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
    ------------------------------------------------------------------------
    Output Module
    ======================================================================== */
20
    #ifndef OUTPUT_H
    #define OUTPUT_H 1

    #include "config.h"
25  #include "minmax.h"
    #include "stretch.h"

    //========================================================================
    // output data structures
30
    struct output {
      struct minmax minmax;
      struct stretch stretch;
      unsigned int frame;
35    int width;
      int height;
      GLuint texture;
      int winwidth;
      int winheight;
40  };

    //========================================================================
    // prototypes

45  struct output *output_init(struct output *output);
    void output_reshape(struct output *output, int ww, int hh, int w, int h);
    void output_display(struct output *output, GLuint texture, GLuint fbo);

    #endif
```

39 gpu/revisited.c

```
    /* ========================================================================
    soft-rock-revisited
    Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

 5  This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

10  This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

15  You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

```
/*------------------------------------------------------------------------
Main Module
========================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "revisited.h"

//========================================================================
// main module global mutable state
static struct revisited revisited;

//========================================================================
// main module initialization
int revisited_init() {
  memset(&revisited, 0, sizeof(revisited));
  for (int b = 0; b < config_beats; ++b) {
    if (! map_init(&revisited.map[b])) { return 0; }
    if (! colour_init(&revisited.colour[b])) { return 0; }
  }
  if (! output_init(&revisited.output)) { return 0; }
  glGenFramebuffersEXT(1, &revisited.fbo);
  revisited_reshape(config_win_width, config_win_height);
  if (! audio_init(&revisited.audio, &revisited.map[0])) { return 0; }
  revisited.starttime = 0;
  revisited.fullscreen = 0;
  return 1;
}

//========================================================================
// main module reshape callback
void revisited_reshape(int w, int h) {
  for (int b = 0; b < config_beats; ++b) {
    map_reshape(&revisited.map[b], config_tex_width, config_tex_height, w, h);
    colour_reshape(&revisited.colour[b], config_tex_width, config_tex_height);
  }
  output_reshape(&revisited.output, config_win_width, config_win_height, w, h);
}

//========================================================================
// main module display callback
void revisited_display() {
  revisited.beat = audio_beat;
  GLuint texs[config_beats];
  for (int m = 0; m < config_beats; ++m) {
    GLuint tex = map_display(&revisited.map[m], revisited.fbo);
    //colour_display(&revisited.colour[m], revisited.fbo, tex);
    texs[m] = tex; //revisited.colour[m].texture;
  }
  output_display(&revisited.output, texs[revisited.beat], revisited.fbo);
  glutSwapBuffers();
  glutReportErrors();
  revisited.frame += 1;
}
```

```
//=============================================================================
// main module exit callback
void revisited_atexit(void) {
  double timeelapsed = (double) time(NULL) - (double) revisited.starttime;
  fprintf(stderr, "\n\n↙
      ↪ ------------------------------------------------------------------------\↙
      ↪ n");
  fprintf(stderr, "------------------------------ statistics ↙
      ↪ -------------------------------\n");
  fprintf(stderr, ↙
      ↪ "------------------------------------------------------------------------\↙
      ↪ n");
  fprintf(stderr, "%10d seconds elapsed (+/- 1)\n", (int) timeelapsed);
  fprintf(stderr, "%10d frames rendered (%f fps)\n", revisited.frame, revisited.↙
      ↪ frame / timeelapsed);
  fprintf(stderr, ↙
      ↪ "------------------------------------------------------------------------\↙
      ↪ n");
}

//=============================================================================
// main module idle callback
void revisited_idle() {
  if (revisited.starttime == 0) {
    revisited.starttime = time(NULL);
    revisited.frame = 0;
    atexit(revisited_atexit);
  }
  glutPostRedisplay();
}

void revisited_keynormal(unsigned char key, int x, int y) {
  switch (key) {
  case 27: // escape
    exit(0);  // FIXME
    break;
  default:
    map_keynormal(&revisited.map[revisited.beat], key, x, y);
    break;
  }
}

void revisited_keyspecial(int key, int x, int y) {
  map_keyspecial(&revisited.map[revisited.beat], key, x, y);
}

void revisited_mouse(int button, int state, int x, int y) {
  map_mouse(&revisited.map[revisited.beat], button, state, x, y);
}

void revisited_motion(int x, int y) {
  map_motion(&revisited.map[revisited.beat], x, y);
}

// EOF
```

40 gpu/revisited.h

```
/* ============================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
----------------------------------------------------------------------------
Main Module
============================================================================ */

#ifndef REVISITED_H
#define REVISITED_H 1

#include <time.h>
#include <GL/glew.h>
#include <GL/glut.h>

#include "config.h"
#include "audio.h"
#include "map.h"
#include "colour.h"
#include "output.h"

//============================================================================
// main module data
struct revisited {
  struct audio audio;
  struct map map[config_beats];
  struct colour colour[config_beats];
  struct output output;
  GLuint fbo;
  time_t starttime;
  unsigned int frame;
  int fullscreen;
  int beat;
};

//============================================================================
// prototypes
int revisited_init(void);
void revisited_reshape(int w, int h);
void revisited_display(void);
void revisited_atexit(void);
void revisited_idle(void);
void revisited_keynormal(unsigned char key, int x, int y);
```

```
void revisited_keyspecial(int key, int x, int y);
void revisited_mouse(int button, int state, int x, int y);
void revisited_motion(int x, int y);

#endif
```

41 gpu/s2c.sh

```
#!/bin/bash
echo "/* machine-generated file, do not edit */"
echo "static const char $1[] ="
sed 's|//.*||' |
sed 's|^|"|' |
sed 's|$|\\n"|'
echo ";"
```

42 gpu/shader.c

```
/* ============================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
----------------------------------------------------------------------------
Generic Shader
============================================================================ */

#include <stdio.h>
#include <stdlib.h>
#include "shader.h"

//============================================================================
// print a shader object's debug log
void shader_debug(GLhandleARB obj) {
//  return; // FIXME: only dump logs when shader compile/link failed
  int infologLength = 0;
  int maxLength;
  if (glIsShader(obj)) {
    glGetShaderiv(obj, GL_INFO_LOG_LENGTH, &maxLength);
  } else {
    glGetProgramiv(obj, GL_INFO_LOG_LENGTH, &maxLength);
  }
  char *infoLog = malloc(maxLength);
  if (!infoLog) {
    return;
```

```
  }
  if (glIsShader(obj)) {
    glGetShaderInfoLog(obj, maxLength, &infologLength, infoLog);
  } else {
    glGetProgramInfoLog(obj, maxLength, &infologLength, infoLog);
  }
  if (infologLength > 0) {
    fprintf(stderr, "%s\n", infoLog);
  }
  free(infoLog);
}

//=============================================================
// generic shader initialization
struct shader *shader_init(
  struct shader *shader, const char *vert, const char *frag
) {
  if (! shader) { return 0; }
  shader->linkStatus     = 0;
  shader->vertexSource   = vert;
  shader->fragmentSource = frag;
  if (shader->vertexSource || shader->fragmentSource) {
    shader->program = glCreateProgramObjectARB();
    if (shader->vertexSource) {
      shader->vertex =
        glCreateShaderObjectARB(GL_VERTEX_SHADER_ARB);
      glShaderSourceARB(shader->vertex,
        1, (const GLcharARB **) &shader->vertexSource, 0
      );
      glCompileShaderARB(shader->vertex);
      shader_debug(shader->vertex);
      glAttachObjectARB(shader->program, shader->vertex);
    }
    if (shader->fragmentSource) {
      shader->fragment =
        glCreateShaderObjectARB(GL_FRAGMENT_SHADER_ARB);
      glShaderSourceARB(shader->fragment,
        1, (const GLcharARB **) &shader->fragmentSource, 0
      );
      glCompileShaderARB(shader->fragment);
      shader_debug(shader->fragment);
      glAttachObjectARB(shader->program, shader->fragment);
    }
    glLinkProgramARB(shader->program);
    shader_debug(shader->program);
    glGetObjectParameterivARB(shader->program,
      GL_OBJECT_LINK_STATUS_ARB, &shader->linkStatus
    );
    if (! shader->linkStatus) { return 0; }
  } else { return 0; }
  return shader;
}
```

43 gpu/shader.h

```
/* =============================================================
soft-rock-revisited
```

```
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
-------------------------------------------------------------------------
Generic Shader
========================================================================= */

#ifndef SHADER_H
#define SHADER_H 1

#include <GL/glew.h>

//=========================================================================
// generic shader data
struct shader {
  GLint linkStatus;
  GLhandleARB program;
  GLhandleARB fragment;
  GLhandleARB vertex;
  const GLcharARB *fragmentSource;
  const GLcharARB *vertexSource;
};

struct vec4 {
  float v[4];
};

struct vec3 {
  float v[3];
};

struct vec2 {
  float v[2];
};

//=========================================================================
// generic shader uniform location access macro
#define shader_uniform(self,name) \
  (self)->uniform.name = \
    glGetUniformLocationARB((self)->shader.program, #name)

//=========================================================================
// generic shader uniform update access macro (integer)
#define shader_updatei(self,name) \
  glUniform1iARB((self)->uniform.name, (self)->value.name)
```

```
//=============================================================================
// generic shader uniform update access macro (float)
#define shader_updatef(self,name) \
  glUniform1fARB((self)->uniform.name, (self)->value.name)

//=============================================================================
// generic shader uniform update access macro (float3)
#define shader_updatef3(self,name) \
  glUniform3fARB((self)->uniform.name, (self)->value.name.v[0], (self)->value.name.v[1], (self)->value.name.v[2])

//=============================================================================
// generic shader uniform update access macro (float4)
#define shader_updatef4(self,name) \
  glUniform4fARB((self)->uniform.name, (self)->value.name.v[0], (self)->value.name.v[1], (self)->value.name.v[2], (self)->value.name.v[3])

//=============================================================================
// generic shader uniform update access macro (float4v)
#define shader_updatef4v(self,name, count) \
  glUniform4fvARB((self)->uniform.name, count, &((self)->value.name[0].v[0]))

//=============================================================================
// generic shader uniform update access macro (float2v)
#define shader_updatef2v(self,name, count) \
  glUniform2fvARB((self)->uniform.name, count, &((self)->value.name[0].v[0]))

//=============================================================================
// generic shader initialization
struct shader *shader_init(
  struct shader *shader, const char *vert, const char *frag
);

#endif
```

44 gpu/stretch.c

```
/* =============================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
-----------------------------------------------------------------------------
RGB Stretch Shader
============================================================================= */

```

```
#include <stdio.h>
#include <GL/glew.h>
#include <GL/glut.h>

#include "config.h"
#include "util.h"
#include "stretch.h"
#include "stretch.frag.c"

//============================================================
// rgb stretch shader initialization
struct stretch *stretch_init(struct stretch *stretch) {
  if (! stretch) { return 0; }
  if (! shader_init(&stretch->shader, 0, stretch_frag)) {
    return 0;
  }
  shader_uniform(stretch, tex);
  shader_uniform(stretch, sub);
  shader_uniform(stretch, mul);
  stretch->value.tex = 0;
  stretch->value.sub.v[0] = 0;
  stretch->value.sub.v[1] = 0;
  stretch->value.sub.v[2] = 0;
  stretch->value.mul.v[0] = 1;
  stretch->value.mul.v[1] = 1;
  stretch->value.mul.v[2] = 1;
  glGenTextures(1, &stretch->texture);
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, stretch->texture);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR)↙
    ↳ ;
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
  glDisable(GL_TEXTURE_2D);
  stretch->width  = 0;
  stretch->height = 0;
  return stretch;
}

//============================================================
// false stretch shader reshape callback
void stretch_reshape(struct stretch *stretch, int w, int h) {
  stretch->width  = roundtwo(w);
  stretch->height = roundtwo(h);
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, stretch->texture);
  glTexImage2D(
    GL_TEXTURE_2D, 0, GL_RGBA32F_ARB, stretch->width, stretch->height,
    0, GL_RGBA, GL_FLOAT, 0
  );
  glDisable(GL_TEXTURE_2D);
}

//============================================================
// false stretch shader display callback
void stretch_display(struct stretch *stretch, GLuint fbo, GLuint texture, float ↙
    ↳ *mi, float *ma) {
  glEnable(GL_TEXTURE_2D);
```

```
  glBindTexture(GL_TEXTURE_2D, texture);
  glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fbo);
  glFramebufferTexture2DEXT(
    GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT, GL_TEXTURE_2D,
    stretch->texture, 0
  );
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(0, 1, 0, 1);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glViewport(0, 0, stretch->width, stretch->height);
  glUseProgramObjectARB(stretch->shader.program);
  stretch->value.sub.v[0] = mi[0];
  stretch->value.sub.v[1] = mi[1];
  stretch->value.sub.v[2] = mi[2];
  stretch->value.mul.v[0] = 1.0 / (ma[0] - mi[0]);
  stretch->value.mul.v[1] = 1.0 / (ma[1] - mi[1]);
  stretch->value.mul.v[2] = 1.0 / (ma[2] - mi[2]);
  shader_updatei(stretch, tex);
  shader_updatef3(stretch, sub);
  shader_updatef3(stretch, mul);
  glBegin(GL_QUADS); { glColor4f(1,1,1,1);
    glTexCoord2f(0, 0); glVertex2f(0, 0);
    glTexCoord2f(1, 0); glVertex2f(1, 0);
    glTexCoord2f(1, 1); glVertex2f(1, 1);
    glTexCoord2f(0, 1); glVertex2f(0, 1);
  } glEnd();
  glUseProgramObjectARB(0);
  glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT, ↙
      ↳ GL_TEXTURE_2D, 0, 0);
  glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
  glBindTexture(GL_TEXTURE_2D, stretch->texture);
  glGenerateMipmap(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, 0);
  glDisable(GL_TEXTURE_2D);
}

//=============================================================================
// false stretch shader idle callback
void stretch_idle(struct stretch *stretch) { }

// EOF
```

45 gpu/stretch.frag

```
#version 120
precision highp float;

uniform sampler2D tex;
uniform vec3 sub;
uniform vec3 mul;

void main() {
  vec3 t = texture2D(tex, gl_TexCoord[0].xy).rgb;
  gl_FragData[0] = vec4(mix(t, (t - sub) * mul, 0.25), 1.0);
}
```

46 gpu/stretch.h

```
/* ================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
----------------------------------------------------------------------
RGB Stretch Shader
================================================================ */

#ifndef STRETCH_H
#define STRETCH_H 1

#include "shader.h"

//================================================================
// stretch shader data
struct stretch { struct shader shader;
  struct { GLint tex; GLint sub, mul; } uniform;
  struct { int   tex; struct vec3 sub, mul; } value;
  GLuint texture;
  int width;
  int height;
};

//================================================================
// protoypes
struct stretch *stretch_init(struct stretch *stretch);
void stretch_reshape(struct stretch *stretch, int w, int h);
void stretch_display(struct stretch *stretch, GLuint fbo, GLuint texture, float
    ↳ *mi, float *ma);
void stretch_idle(struct stretch *stretch);

#endif
```

47 gpu/TODO

```
TODO
====

2012-10-30      split screen mode with recursion
```

48 gpu/util.c

```
/* ================================================================
kitchen
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
----------------------------------------------------------------
Utility Functions
================================================================ */

#include <assert.h>
#include <math.h>
#include "util.h"

//================================================================
// round 'x' up to the nearest power of two (which may be 'x' itself)
unsigned int roundtwo(unsigned int x) {
  assert(x <= 1u << 31u); // termination condition
  unsigned int y = 1;
  while (y < x) y <<= 1;
  return y;
}

//================================================================
// find log2 of the nearest power of two above 'x'
unsigned int logtwo(unsigned int x) {
  assert(x <= 1u << 31u); // termination condition
  unsigned int y = 1, z = 0;
  while (y < x) { y <<= 1; z += 1; };
  return z;
}

float ftom(float f) {
  return log(f / 8.17579891564) / 0.0577622650;
}

float clamp(float f, float lo, float hi) {
  return fmaxf(lo, fminf(f, hi));
}

// EOF
```

49 gpu/util.h

```
/* ================================================================
kitchen
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>
```

```
5   This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

10  This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

15  You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
    ------------------------------------------------------------------------
    Utility Functions
    ======================================================================== */
20
    #ifndef UTIL_H
    #define UTIL_H 1

    unsigned int roundtwo(unsigned int x);
25  unsigned int logtwo(unsigned int x);

    float ftom(float f);
    float clamp(float x, float lo, float hi);

30  #endif
```

50 perform/audio.c

```
    /* ========================================================================
    soft-rock-revisited
    Copyright (C) 2010-2012  Claude Heiland-Allen <claude@mathr.co.uk>
    ------------------------------------------------------------------------
5   Audio output
    ======================================================================== */

    #include <math.h>
    #include <stdio.h>
10  #include <stdlib.h>
    #include <string.h>
    #include <sys/types.h>

    #include "config.h"
15  #include "audio.h"

    /* {{{ copy/pasted from pd-0.42, hardcoded little-endian {{{ */

    #define UNITBIT32 1572864.  /* 3*2^19; bit 32 has place value 1 */
20  #define HIOFFSET 1
    #define LOWOFFSET 0
    #define int32 int32_t
    #define LOGCOSTABSIZE 9
    #define COSTABSIZE (1<<LOGCOSTABSIZE)
25
    float cos_table[COSTABSIZE+1];
```

```
union tabfudge
{
    double tf_d;
    int32 tf_i[2];
};

static inline float costab(float p) {
        union tabfudge tf;
        double dphase = (double)(p * (float)(COSTABSIZE)) + UNITBIT32;
        tf.tf_d = UNITBIT32;
        int normhipart = tf.tf_i[HIOFFSET];
        tf.tf_d = dphase;
        float *addr = cos_table + (tf.tf_i[HIOFFSET] & (COSTABSIZE-1));
        tf.tf_i[HIOFFSET] = normhipart;
        float frac = tf.tf_d - UNITBIT32;
        float f1 = addr[0];
        float f2 = addr[1];
        return f1 + frac * (f2 - f1);
}

static void cos_maketable(void)
{
    int i;
    float *fp, phase, phsinc = (2. * 3.14159) / COSTABSIZE;
    for (i = COSTABSIZE + 1, fp = cos_table, phase = 0; i--;
        fp++, phase += phsinc)
            *fp = cos(phase);
}

/* }}} copy/pasted from pd-0.42 }}} */

static inline float mtoff(float f) {
  return 8.17579891564f * expf(0.0577622650f * f);
}

void audio_compute(struct audio *audio, jack_default_audio_sample_t **out, ↙
    ↳ jack_nframes_t nframes) {
  float sr = audio->sr;
  float ophase[config_oscillators][2];
  float phase[config_oscillators][2];
  float note[config_oscillators];
  float modi[config_oscillators];
  note[0] = audio->zoomer->note;
  modi[0] = audio->zoomer->modi;
  float nradius = audio->zoomer->noter / ZOOMER_WIN_SIZE;
  float mradius = audio->zoomer->modir / ZOOMER_WIN_SIZE;
  for (int o = 1; o < config_oscillators; ++o) {
    note[o] = note[0] + nradius * cos(o * 6.283185307179586 / (↙
        ↳ config_oscillators - 1));
    modi[o] = modi[0] + mradius * sin(o * 6.283185307179586 / (↙
        ↳ config_oscillators - 1));
  }
  float pan[config_oscillators][2];
  for (int o = 0; o < config_oscillators; ++o) {
    phase[o][0] = audio->phase[o][0];
    phase[o][1] = audio->phase[o][1];
    if (o == 0) {
```

```
      pan[o][1] = sqrtf(0.5);
    } else {
      pan[o][1] = cosf(6.283185307179586f * 0.125 * (0.9 * cosf((o + 0.5) * ↙
          ↳ 6.283185307179586f / (config_oscillators - 1)) + 1.0));
85  }
    pan[o][0] = sqrtf(1 - pan[o][1] * pan[o][1]);
    pan[o][0] *= (o == 0 ? 0.5 : (0.5 / (config_oscillators - 1)));
    pan[o][1] *= (o == 0 ? 0.5 : (0.5 / (config_oscillators - 1)));
  }
90  for (jack_nframes_t n = 0; n < nframes; ++n) {
    out[0][n] = 0.0;
    out[1][n] = 0.0;
    // FM oscillators
    for (int o = 0; o < config_oscillators; ++o) {
95    audio->osc[o][0][audio->oscn] = costab(phase[o][0]);
      audio->osc[o][1][audio->oscn] = costab(phase[o][1]);
      for (int i = 0; i < 2; ++i) {
        int j = 1 - i;
        float p = phase[o][i] + mtoff(note[o] + modi[o] * audio->osc[o][j][(↙
            ↳ audio->oscn + 1) % config_blocksize]) / sr;
100     ophase[o][i] = p - floorf(p);
      }
      for (int i = 0; i < 2; ++i) {
        phase[o][i] = ophase[o][i];
        out[i][n] += pan[o][i] * audio->osc[o][i][audio->oscn];
105   }
    }
    // DC blocker
    for (int i = 0; i < 2; ++i) {
      audio->x1[i] = audio->x[i];
110   audio->x[i] = out[i][n];
      out[i][n] = audio->x[i] - audio->x1[i] + 0.999 * audio->y1[i];
      audio->y1[i] = out[i][n];
      if (audio->scope_phase < SCOPE_SIZE) {
        audio->scope->buf[i][audio->scope_phase] = out[i][n];
115   }
    }
    audio->scope_phase = (audio->scope_phase + 1) % SCOPE_INTERVAL;
    // feedback delay
    audio->oscn = (audio->oscn + 1) % config_blocksize;
120 }
  // update state
  audio->sphase += nframes / (double) audio->sr;
  for (int o = 0; o < config_oscillators; ++o) {
    audio->phase[o][0] = phase[o][0];
125 audio->phase[o][1] = phase[o][1];
  }
}

/* JACK audio processing callback */
130 static int audio_process(jack_nframes_t nframes, void *arg) {
  struct audio *audio = arg;
  /* get JACK buffers */
  jack_default_audio_sample_t *out[2];
  out[0] = (jack_default_audio_sample_t *)
135   jack_port_get_buffer(audio->port[0], nframes);
  out[1] = (jack_default_audio_sample_t *)
```

```
    jack_port_get_buffer(audio->port[1], nframes);
  audio_compute(audio, out, nframes);
  return 0;
}

/* JACK sample rate callback */
static int audio_srate(jack_nframes_t sr, void *arg) {
  struct audio *audio = arg;
  if ((jack_nframes_t) audio->sr != sr) {
    fprintf(stderr, "WARN: JACK sample rate: %d != %d\n", sr, audio->sr);
  }
  return 0;
}

/* JACK error callback */
static void audio_error(const char *desc) {
  fprintf(stderr, "JACK error: %s\n", desc);
}

/* JACK shutdown callback */
static void audio_shutdown(void *arg) {
  fprintf(stderr, "JACK shutdown\n");
}

/* JACK timebase callback */
static void audio_timebase(jack_transport_state_t state, jack_nframes_t nframes,↙
    ↪ jack_position_t *pos, int new_pos, void *arg) {
  //struct audio *audio = arg;
  if (new_pos) {
    pos->valid = JackPositionBBT;
    pos->beats_per_bar = 4;
    pos->beat_type = 0.25;
    pos->ticks_per_beat = 1920;
    pos->beats_per_minute = 120;
    pos->bar = 1;
    pos->beat = 1;
    pos->tick = 0;
    pos->bar_start_tick = (pos->bar - 1) * pos->beats_per_bar * pos->↙
        ↪ ticks_per_beat;
  } else {
    pos->tick +=nframes * pos->ticks_per_beat * pos->beats_per_minute / (pos->↙
        ↪ frame_rate * 60);
    while (pos->tick >= pos->ticks_per_beat) {
      pos->tick -= pos->ticks_per_beat;
      if (++pos->beat > pos->beats_per_bar) {
        pos->beat = 1;
        ++pos->bar;
        pos->bar_start_tick += pos->beats_per_bar * pos->ticks_per_beat;
      }
    }
  }
}

/* exit callback */
void audio_atexit(struct audio *audio) {
  jack_client_close(audio->client);
}
```

```
//=============================================================================
// ...
struct audio *audio_init(struct audio *audio, struct zoomer *zoomer, struct ↙
    ↳ scope *scope) {
195   if (! audio) { return 0; }
      cos_maketable();
      audio->zoomer = zoomer;
      audio->scope = scope;
      audio->scope_phase = 0;
200   audio->samples = 0;
      for (int o = 0; o < config_oscillators; ++o) {
        audio->phase[o][0] = 0;
        audio->phase[o][1] = 0;
      }
205   audio->oscn = 0;
      audio->sr = config_samplerate;
      audio->sphase = 0;
      audio->bphase = 0;
      audio->blength = audio->sr * 60.0 / config_bpm;
210   jack_set_error_function(audio_error);
      if (!(audio->client = jack_client_new("srrv"))) {
        fprintf (stderr, "JACK server not running?\n");
      } else {
        jack_set_process_callback(audio->client, audio_process, audio);
215     jack_set_sample_rate_callback(audio->client, audio_srate, audio);
        jack_on_shutdown(audio->client, audio_shutdown, audio);
        /* create ports */
        audio->port[0] = jack_port_register(
          audio->client, "output_1", JACK_DEFAULT_AUDIO_TYPE, JackPortIsOutput, 0
220     );
        audio->port[1] = jack_port_register(
          audio->client, "output_2", JACK_DEFAULT_AUDIO_TYPE, JackPortIsOutput, 0
        );
        /* activate audio */
225     if (jack_activate(audio->client)) {
          fprintf (stderr, "cannot activate JACK client\n");
        } else {
          /* must be activated before connecting JACK ports */
          const char **ports;
230       if ((ports = jack_get_ports(
            audio->client, NULL, NULL, JackPortIsPhysical | JackPortIsInput
          ))) {
            /* connect up to two physical playback ports */
            int i = 0;
235         while (ports[i] && i < 2) {
              if (jack_connect(
                audio->client, jack_port_name(audio->port[i]), ports[i]
              )) {
                fprintf(stderr, "cannot connect JACK output port\n");
240           }
              i++;
            }
            free(ports);
          }
245       /* become timebase master */
          if (jack_set_timebase_callback(audio->client, 0, audio_timebase, audio)) {
```

```
        fprintf(stderr, "cannot become JACK master\n");
      }
      jack_transport_start(audio->client);
    }
  }
  return audio;
}

// EOF
```

51 perform/audio.h

```
#ifndef AUDIO_H
#define AUDIO_H 1

#include <stdio.h>
#include <jack/jack.h>

#include "config.h"
#include "scope.h"
#include "zoomer.h"

struct audio {
  struct zoomer *zoomer;
  struct scope *scope;
  double samples; // total samples scanned
  jack_client_t *client;
  jack_port_t *port[2];
  double sphase;
  int sr;
  float phase[config_oscillators][2];
  float osc[config_oscillators][2][config_blocksize];
  int oscn;
  double bphase;
  double blength;
  float x[2], x1[2], y1[2];
  int scope_phase;
};

struct audio *audio_init(struct audio *audio, struct zoomer *zoomer, struct ↙
    ↳ scope *scope);
void audio_atexit(struct audio *audio);

#endif
```

52 perform/config.h

```
/* ==========================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
```

```
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
    ------------------------------------------------------------------------
    Configuration
    ======================================================================== */

#ifndef CONFIG_H
#define CONFIG_H 1

// ============================================================================
// expert settings
#define config_aspect         1
#define config_tex_width    256
#define config_tex_height   256
#define config_win_width   1024
#define config_win_height   512
#define config_scale_factor   1
#define config_samplerate 48000
#define config_oscillators    5
#define config_beats          1
#define config_bpm           60
#define config_overdrive      8
#define config_blocksize     32

#endif
```

53 perform/main.c

```
/* ============================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
    ------------------------------------------------------------------------
    Main Program Entry Point
    ======================================================================== */

#include <stdio.h>
#include <GL/glew.h>
#include <GL/glut.h>
```

```
#include "revisited.h"
#include "config.h"

int main(int argc, char **argv) {
  /* initialize GLUT etc */
  fprintf(stderr, "soft-rock-revisited (GPL) 2012 Claude Heiland-Allen <↙
      ↳ claude@mathr.co.uk>\n");
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutInitWindowSize(config_win_width, config_win_height);
  glutInit(&argc, argv);
  glutCreateWindow("srrv");
  glewInit();
  glClampColor(GL_CLAMP_FRAGMENT_COLOR, GL_FALSE);
  glClampColor(GL_CLAMP_READ_COLOR, GL_FALSE);
  /* activate callbacks and enter main loop */
  if (revisited_init()) {
    glutDisplayFunc(revisited_display);
    glutReshapeFunc(revisited_reshape);
    glutIdleFunc(revisited_idle);
    glutKeyboardFunc(revisited_keynormal);
    glutSpecialFunc(revisited_keyspecial);
    glutMouseFunc(revisited_mouse);
    glutMotionFunc(revisited_motion);
    glutPassiveMotionFunc(revisited_motion);
    glutSetCursor(GLUT_CURSOR_CROSSHAIR);
    glutMainLoop();
    return 0; // never reached
  } else {
    return 1;
  }
}
```

54 perform/Makefile

```
CC = gcc
CFLAGS = -std=c99 -Wall -pedantic -Wextra -Wno-unused-parameter -O3 -march=↙
    ↳ native -pthread `pkg-config --cflags jack`
LIBS = -lGLEW -lGL -lGLU -lglut `pkg-config --libs jack`
OBJS = main.o revisited.o shader.o util.o audio.o scope.o zoomer.o
GENC = zoomer.frag.c

# optional feature: debuggability
#CFLAGS += -ggdb

all: soft-rock-revisited

clean:
	-rm -f $(OBJS) $(GENC)

# no suffix rules, they cause weird issues with the .frag.c files
.SUFFIXES:
.PHONY: all clean

# link program
soft-rock-revisited: $(OBJS)
	$(CC) $(CFLAGS) -o soft-rock-revisited $(OBJS) $(LIBS)
```

```
# C source to object file
%.o: %.c
	$(CC) $(CFLAGS) -o $@ -c $<

# shader source to C source
%.frag.c: %.frag s2c.sh
	./s2c.sh $*_frag < $< > $@

# dependencies
audio.o: audio.c audio.h scope.h zoomer.h config.h
main.o: main.c revisited.h config.h
revisited.o: revisited.c revisited.h audio.h scope.h zoomer.h config.h
scope.o: scope.c scope.h scope.frag.c config.h shader.h
shader.o: shader.c shader.h
util.o: util.c util.h
zoomer.o: zoomer.c zoomer.h zoomer.frag.c util.h shader.h config.h
```

55 perform/revisited.c

```
/* ===========================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
---------------------------------------------------------------------------
Main Module
=========================================================================== */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "revisited.h"

//============================================================================
// main module global mutable state
static struct revisited revisited;

//============================================================================
// main module initialization
int revisited_init() {
  memset(&revisited, 0, sizeof(revisited));
  if (! zoomer_init(&revisited.zoomer, "srrv.data", 64.0, 64.0, 57.875, 36.0)) {↙
      ↪ return 0; }
  if (! scope_init(&revisited.scope)) { return 0; }
  if (! audio_init(&revisited.audio, &revisited.zoomer, &revisited.scope)) { ↙
```

```
      ↪ return 0; }
  revisited.starttime = 0;
  revisited.fullscreen = 0;
  return 1;
}

//=============================================================================
// main module reshape callback
void revisited_reshape(int w, int h) {
}

//=============================================================================
// main module display callback
void revisited_display() {
  zoomer_display(&revisited.zoomer);
  scope_display(&revisited.scope);
  glutSwapBuffers();
  glutReportErrors();
  revisited.frame += 1;
}

//=============================================================================
// main module exit callback
void revisited_atexit(void) {
  double timeelapsed = (double) time(NULL) - (double) revisited.starttime;
  fprintf(stderr, "\n\n↙
      ↪ ------------------------------------------------------------------------\↙
      ↪ n");
  fprintf(stderr, "------------------------------ statistics ↙
      ↪ -------------------------------\n");
  fprintf(stderr, ↙
      ↪ "------------------------------------------------------------------------\↙
      ↪ n");
  fprintf(stderr, "%10d seconds elapsed (+/- 1)\n", (int) timeelapsed);
  fprintf(stderr, "%10d frames rendered (%f fps)\n", revisited.frame, revisited.↙
      ↪ frame / timeelapsed);
  fprintf(stderr, ↙
      ↪ "------------------------------------------------------------------------\↙
      ↪ n");
}

//=============================================================================
// main module idle callback
void revisited_idle() {
  if (revisited.starttime == 0) {
    revisited.starttime = time(NULL);
    revisited.frame = 0;
    atexit(revisited_atexit);
  }
  glutPostRedisplay();
}

void revisited_keynormal(unsigned char key, int x, int y) {
  switch (key) {
  case 27: // escape
    exit(0);  // FIXME
    break;
```

```
    default:
      zoomer_key(&revisited.zoomer, key);
      break;
    }
}

void revisited_keyspecial(int key, int x, int y) {
}

void revisited_mouse(int button, int state, int x, int y) {
  zoomer_mouse(&revisited.zoomer, x, y);
}

void revisited_motion(int x, int y) {
  zoomer_mouse(&revisited.zoomer, x, y);
}

// EOF
```

56 perform/revisited.h

```
#ifndef REVISITED_H
#define REVISITED_H 1

#include <time.h>
#include <GL/glew.h>
#include <GL/glut.h>

#include "config.h"
#include "audio.h"
#include "scope.h"
#include "zoomer.h"

//=============================================================================
// main module data
struct revisited {
  struct audio audio;
  struct scope scope;
  struct zoomer zoomer;
  time_t starttime;
  unsigned int frame;
  int fullscreen;
};

//=============================================================================
// prototypes
int revisited_init(void);
void revisited_reshape(int w, int h);
void revisited_display(void);
void revisited_atexit(void);
void revisited_idle(void);
void revisited_keynormal(unsigned char key, int x, int y);
void revisited_keyspecial(int key, int x, int y);
void revisited_mouse(int button, int state, int x, int y);
void revisited_motion(int x, int y);

#endif
```

57 perform/s2c.sh

```
#!/bin/bash
echo "/* machine-generated file, do not edit */"
echo "static const char $1[] ="
sed 's|//.*||' |
sed 's|^|"|' |
sed 's|$|\\n"|'
echo ";"
```

58 perform/scope.c

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glew.h>

#include "scope.h"
#include "scope.frag.c"
#include "util.h"

struct scope *scope_init(struct scope *scope) {
  if (! scope) { return 0; }
  if (! shader_init(&scope->shader, 0, scope_frag)) { return 0; }
  shader_uniform(scope, tex0);
  shader_uniform(scope, tex1);
  scope->value.tex0 = 0;
  scope->value.tex1 = 1;
  glGenTextures(2, &scope->tex[0]);
  for (int i = 0; i < 2; ++i) {
    glBindTexture(GL_TEXTURE_1D, scope->tex[i]);
    glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexImage1D(GL_TEXTURE_1D, 0, GL_R32F, SCOPE_SIZE, 0, GL_RED, GL_FLOAT, &↙
        ↳ scope->buf[0][0]);
    glBindTexture(GL_TEXTURE_1D, 0);
  }
  return scope;
}

void scope_display(struct scope *scope) {
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(0, 1, 0, 1);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glViewport(SCOPE_WIN_X, SCOPE_WIN_Y, SCOPE_WIN_SIZE, SCOPE_WIN_SIZE);
  glActiveTexture(GL_TEXTURE1);
  glBindTexture(GL_TEXTURE_1D, scope->tex[1]);
  glTexSubImage1D(GL_TEXTURE_1D, 0, 0, SCOPE_SIZE, GL_RED, GL_FLOAT, &scope->buf↙
      ↳ [1][0]);
  glActiveTexture(GL_TEXTURE0);
  glBindTexture(GL_TEXTURE_1D, scope->tex[0]);
  glTexSubImage1D(GL_TEXTURE_1D, 0, 0, SCOPE_SIZE, GL_RED, GL_FLOAT, &scope->buf↙
      ↳ [0][0]);
  glUseProgramObjectARB(scope->shader.program);
```

```
  shader_updatei(scope, tex0);
  shader_updatei(scope, tex1);
  glBegin(GL_QUADS); { glColor4f(1,1,1,1);
    glTexCoord2f(0, 0); glVertex2f(0, 0);
    glTexCoord2f(1, 0); glVertex2f(1, 0);
    glTexCoord2f(1, 1); glVertex2f(1, 1);
    glTexCoord2f(0, 1); glVertex2f(0, 1);
  } glEnd();
  glUseProgramObjectARB(0);
  glActiveTexture(GL_TEXTURE1);
  glBindTexture(GL_TEXTURE_1D, 0);
  glActiveTexture(GL_TEXTURE0);
  glBindTexture(GL_TEXTURE_1D, 0);
}
```

59 perform/scope.frag

```
#version 120
precision highp float;

uniform sampler1D tex0;
uniform sampler1D tex1;

void main() {
  vec2 p = gl_TexCoord[0].xy;
  float a = texture1D(tex0, p.x).x;
  float b = texture1D(tex1, p.y).x;
  vec3 yuv = vec3(sqrt(abs(a * b)), 0.5 * a, 0.5 * b);
  vec3 rgb = yuv * mat3(1.0, 1.407, 0.0, 1.0, -0.677, -0.236, 1.0, 0.0, 1.848);
  gl_FragColor = vec4(clamp(rgb, vec3(0.0), vec3(1.0)), 0.0);
}
```

60 perform/scope.h

```
#ifndef SCOPE_H
#define SCOPE_H 1

#include "shader.h"

#define SCOPE_SIZE 512
// 60fps at 48kHz
#define SCOPE_INTERVAL 800

#define SCOPE_WIN_X 512
#define SCOPE_WIN_Y 0
#define SCOPE_WIN_SIZE 512

struct scope { struct shader shader;
  struct { GLint tex0, tex1; } uniform;
  struct { int tex0, tex1; } value;
  GLuint tex[2];
  float buf[2][SCOPE_SIZE];
};

struct scope *scope_init(struct scope *scope);
void scope_display(struct scope *scope);
```

```
#endif
```

61 perform/shader.c

```
/* ================================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
----------------------------------------------------------------------
Generic Shader
================================================================ */

#include <stdio.h>
#include <stdlib.h>
#include "shader.h"

//================================================================
// print a shader object's debug log
void shader_debug(GLhandleARB obj) {
//  return; // FIXME: only dump logs when shader compile/link failed
  int infologLength = 0;
  int maxLength;
  if (glIsShader(obj)) {
    glGetShaderiv(obj, GL_INFO_LOG_LENGTH, &maxLength);
  } else {
    glGetProgramiv(obj, GL_INFO_LOG_LENGTH, &maxLength);
  }
  char *infoLog = malloc(maxLength);
  if (!infoLog) {
    return;
  }
  if (glIsShader(obj)) {
    glGetShaderInfoLog(obj, maxLength, &infologLength, infoLog);
  } else {
    glGetProgramInfoLog(obj, maxLength, &infologLength, infoLog);
  }
  if (infologLength > 0) {
    fprintf(stderr, "%s\n", infoLog);
  }
  free(infoLog);
}

//================================================================
// generic shader initialization
```

```
struct shader *shader_init(
  struct shader *shader, const char *vert, const char *frag
) {
  if (! shader) { return 0; }
  shader->linkStatus     = 0;
  shader->vertexSource   = vert;
  shader->fragmentSource = frag;
  if (shader->vertexSource || shader->fragmentSource) {
    shader->program = glCreateProgramObjectARB();
    if (shader->vertexSource) {
      shader->vertex =
        glCreateShaderObjectARB(GL_VERTEX_SHADER_ARB);
      glShaderSourceARB(shader->vertex,
        1, (const GLcharARB **) &shader->vertexSource, 0
      );
      glCompileShaderARB(shader->vertex);
      shader_debug(shader->vertex);
      glAttachObjectARB(shader->program, shader->vertex);
    }
    if (shader->fragmentSource) {
      shader->fragment =
        glCreateShaderObjectARB(GL_FRAGMENT_SHADER_ARB);
      glShaderSourceARB(shader->fragment,
        1, (const GLcharARB **) &shader->fragmentSource, 0
      );
      glCompileShaderARB(shader->fragment);
      shader_debug(shader->fragment);
      glAttachObjectARB(shader->program, shader->fragment);
    }
    glLinkProgramARB(shader->program);
    shader_debug(shader->program);
    glGetObjectParameterivARB(shader->program,
      GL_OBJECT_LINK_STATUS_ARB, &shader->linkStatus
    );
    if (! shader->linkStatus) { return 0; }
  } else { return 0; }
  return shader;
}
```

62 perform/shader.h

```
/* ===========================================================
soft-rock-revisited
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

```
-------------------------------------------------------------------------
Generic Shader
========================================================================= */

#ifndef SHADER_H
#define SHADER_H 1

#include <GL/glew.h>

//=========================================================================
// generic shader data
struct shader {
  GLint linkStatus;
  GLhandleARB program;
  GLhandleARB fragment;
  GLhandleARB vertex;
  const GLcharARB *fragmentSource;
  const GLcharARB *vertexSource;
};

struct vec4 {
  float v[4];
};

struct vec3 {
  float v[3];
};

struct vec2 {
  float v[2];
};

//=========================================================================
// generic shader uniform location access macro
#define shader_uniform(self,name) \
  (self)->uniform.name = \
     glGetUniformLocationARB((self)->shader.program, #name)

//=========================================================================
// generic shader uniform update access macro (integer)
#define shader_updatei(self,name) \
  glUniform1iARB((self)->uniform.name, (self)->value.name)

//=========================================================================
// generic shader uniform update access macro (float)
#define shader_updatef(self,name) \
  glUniform1fARB((self)->uniform.name, (self)->value.name)

//=========================================================================
// generic shader uniform update access macro (float3)
#define shader_updatef3(self,name) \
  glUniform3fARB((self)->uniform.name, (self)->value.name.v[0], (self)->value.↙
     ↳ name.v[1], (self)->value.name.v[2])

//=========================================================================
// generic shader uniform update access macro (float4)
#define shader_updatef4(self,name) \
```

```
  glUniform4fARB((self)->uniform.name, (self)->value.name.v[0], (self)->value.↙
      ↳ name.v[1], (self)->value.name.v[2], (self)->value.name.v[3])

//=============================================================================
// generic shader uniform update access macro (float4v)
#define shader_updatef4v(self,name, count) \
  glUniform4fvARB((self)->uniform.name, count, &((self)->value.name[0].v[0]))

//=============================================================================
// generic shader uniform update access macro (float2v)
#define shader_updatef2v(self,name, count) \
  glUniform2fvARB((self)->uniform.name, count, &((self)->value.name[0].v[0]))

//=============================================================================
// generic shader initialization
struct shader *shader_init(
  struct shader *shader, const char *vert, const char *frag
);

#endif
```

63 perform/util.c

```
/* ============================================================================
kitchen
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
----------------------------------------------------------------------------
Utility Functions
============================================================================ */

#include <assert.h>
#include <math.h>
#include "util.h"

//=============================================================================
// round 'x' up to the nearest power of two (which may be 'x' itself)
unsigned int roundtwo(unsigned int x) {
  assert(x <= 1u << 31u); // termination condition
  unsigned int y = 1;
  while (y < x) y <<= 1;
  return y;
}

//=============================================================================
```

```
// find log2 of the nearest power of two above 'x'
unsigned int logtwo(unsigned int x) {
  assert(x <= 1u << 31u); // termination condition
  unsigned int y = 1, z = 0;
  while (y < x) { y <<= 1; z += 1; };
  return z;
}

float ftom(float f) {
  return log(f / 8.17579891564) / 0.0577622650;
}

float clamp(float f, float lo, float hi) {
  return fmaxf(lo, fminf(f, hi));
}

// EOF
```

64 perform/util.h

```
/* ==========================================================================
kitchen
Copyright (C) 2008-2012  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
--------------------------------------------------------------------------
Utility Functions
========================================================================== */

#ifndef UTIL_H
#define UTIL_H 1

unsigned int roundtwo(unsigned int x);
unsigned int logtwo(unsigned int x);

float ftom(float f);
float clamp(float x, float lo, float hi);

#endif
```

65 perform/zoomer.c

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#include <GL/glew.h>

#include "zoomer.h"
#include "zoomer.frag.c"
#include "util.h"

struct zoomer *zoomer_init(struct zoomer *zoomer, const char *datafile, double nsize, double msize, double cnote, double cmodi) {
  if (! zoomer) { return 0; }
  if (! shader_init(&zoomer->shader, 0, zoomer_frag)) { return 0; }
  shader_uniform(zoomer, tex0);
  shader_uniform(zoomer, tex1);
  shader_uniform(zoomer, zoom);
  shader_uniform(zoomer, blend);
  zoomer->value.tex0 = 0;
  zoomer->value.tex1 = 1;
  zoomer->phase = 0;
  zoomer->speed = 0;
  zoomer->nsize = nsize;
  zoomer->msize = msize;
  zoomer->cnote = cnote;
  zoomer->cmodi = cmodi;
  zoomer->note = cnote;
  zoomer->modi = cmodi;
  zoomer->mousex = ZOOMER_WIN_SIZE / 2;
  zoomer->mousey = ZOOMER_WIN_SIZE / 2;
  struct zoomer_pixels *pixels = calloc(1, sizeof(struct zoomer_pixels));
  if (! pixels) { return 0; }
  FILE *f = fopen(datafile, "rb");
  if (! f) { return 0; }
  if (1 != fread(&pixels->pixels[0][0][0][0], sizeof(struct zoomer_pixels), 1, f)) { return 0; }
  fclose(f);
  glGenTextures(ZOOMER_COUNT, &zoomer->tex[0]);
  for (int i = 0; i < ZOOMER_COUNT; ++i) {
    glBindTexture(GL_TEXTURE_2D, zoomer->tex[i]);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, ZOOMER_SIZE, ZOOMER_SIZE, 0, GL_RGB, GL_UNSIGNED_BYTE, &pixels->pixels[i][0][0][0]);
    glGenerateMipmap(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, 0);
  }
  free(pixels);
  return zoomer;
}

void zoomer_display(struct zoomer *zoomer) {
  zoomer->phase += zoomer->speed;
  int w = floor(clamp(zoomer->phase, 0.0, ZOOMER_COUNT - 2.0));
  double s = pow(0.5, zoomer->phase);
  double t = zoomer->phase - w;
  zoomer->value.zoom = pow(0.5, t);
  zoomer->value.blend = (1 - pow(0.25, t)) / (pow(0.25, t - 1) - pow(0.25, t));
```

```
  zoomer->noter = zoomer->nsize * s;
  zoomer->modir = zoomer->msize * s;
  zoomer->note = zoomer->noter * 2.0 * (zoomer->mousey / (double) ↙
      ↳ ZOOMER_WIN_SIZE - 0.5) + zoomer->cnote;
  zoomer->modi = zoomer->modir * 2.0 * (zoomer->mousex / (double) ↙
      ↳ ZOOMER_WIN_SIZE - 0.5) + zoomer->cmodi;
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(0, 1, 0, 1);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glViewport(ZOOMER_WIN_X, ZOOMER_WIN_Y, ZOOMER_WIN_SIZE, ZOOMER_WIN_SIZE);
  glActiveTexture(GL_TEXTURE1);
  glBindTexture(GL_TEXTURE_2D, zoomer->tex[w + 1]);
  glActiveTexture(GL_TEXTURE0);
  glBindTexture(GL_TEXTURE_2D, zoomer->tex[w]);
  glUseProgramObjectARB(zoomer->shader.program);
  shader_updatei(zoomer, tex0);
  shader_updatei(zoomer, tex1);
  shader_updatef(zoomer, zoom);
  shader_updatef(zoomer, blend);
  glBegin(GL_QUADS); { glColor4f(1,1,1,1);
    glTexCoord2f(0, 0); glVertex2f(0, 1);
    glTexCoord2f(1, 0); glVertex2f(1, 1);
    glTexCoord2f(1, 1); glVertex2f(1, 0);
    glTexCoord2f(0, 1); glVertex2f(0, 0);
  } glEnd();
  glUseProgramObjectARB(0);
  glActiveTexture(GL_TEXTURE1);
  glBindTexture(GL_TEXTURE_2D, 0);
  glActiveTexture(GL_TEXTURE0);
  glBindTexture(GL_TEXTURE_2D, 0);
}

int zoomer_key(struct zoomer *zoomer, unsigned char key) {
  if (key == ' ') {
    zoomer->speed = ZOOMER_COUNT / (15.0 * 60 * 60);
    return 1;
  }
  return 0;
}

int zoomer_mouse(struct zoomer *zoomer, int x, int y) {
  if (ZOOMER_WIN_X <= x && x < ZOOMER_WIN_X + ZOOMER_WIN_SIZE &&
      ZOOMER_WIN_Y <= y && y < ZOOMER_WIN_Y + ZOOMER_WIN_SIZE) {
    zoomer->mousex = x - ZOOMER_WIN_X;
    zoomer->mousey = y - ZOOMER_WIN_Y;
    return 1;
  }
  return 0;
}
```

66 perform/zoomer.frag

```
#version 120
precision highp float;
```

```
uniform sampler2D tex0;
uniform sampler2D tex1;
uniform float zoom;
uniform float blend;

void main() {
  vec2 p = vec2(0.5) + zoom * (gl_TexCoord[0].xy - vec2(0.5));
  vec2 q = vec2(0.5) + 0.5 * (p - vec2(0.5));
  gl_FragColor = mix(texture2D(tex0, q), texture2D(tex1, p), blend);
}
```

67 perform/zoomer.h

```
#ifndef ZOOMER_H
#define ZOOMER_H 1

#include "shader.h"

#define ZOOMER_WIN_X 0
#define ZOOMER_WIN_Y 0
#define ZOOMER_WIN_SIZE 512
#define ZOOMER_SIZE 128
#define ZOOMER_COUNT 16

struct zoomer_pixels {
  unsigned char pixels[ZOOMER_COUNT][ZOOMER_SIZE][ZOOMER_SIZE][3];
};

struct zoomer { struct shader shader;
  struct { GLint tex0, tex1, zoom, blend; } uniform;
  struct { int tex0, tex1; float zoom, blend; } value;
  GLuint tex[ZOOMER_COUNT];
  double phase;
  double speed;
  double nsize;
  double msize;
  double cnote;
  double cmodi;
  int mousex;
  int mousey;
  double note;
  double modi;
  double noter;
  double modir;
};

struct zoomer *zoomer_init(struct zoomer *zoomer, const char *datafile, double ↙
    ↳ nsize, double msize, double cnote, double cmodi);
void zoomer_display(struct zoomer *zoomer);
int zoomer_key(struct zoomer *zoomer, unsigned char key);
int zoomer_mouse(struct zoomer *zoomer, int x, int y);

#endif
```

68 soft-rock-ep/Ejecta2.pd

```
#N canvas 92 136 539 512 10;
#X obj 37 159 osc~;
#X obj 38 96 *~ 12;
#X obj 38 139 mtof~;
#X obj 72 208 delwrite~ \$0-delay-1 1000;
#X obj 69 247 dac~;
#X obj 55 183 *~ 0.9;
#X obj 105 183 *~ 0.1;
#X floatatom 142 138 5 0 0 0 - - -;
#X floatatom 95 141 5 0 0 0 - - -;
#X floatatom 93 39 5 0 0 0 - - -;
#X obj 247 159 osc~;
#X obj 248 96 *~ 12;
#X obj 248 139 mtof~;
#X obj 265 183 *~ 0.9;
#X obj 315 183 *~ 0.1;
#X obj 282 209 delwrite~ \$0-delay-2 1000;
#X floatatom 279 44 5 0 0 0 - - -;
#X obj 49 274 writesf~ 2;
#X msg 66 337 stop;
#X obj 101 60 vd~ \$0-delay-1;
#X obj 310 60 vd~ \$0-delay-2;
#X obj 173 6 *~ 10;
#X obj 172 29 +~ 100;
#X obj 323 6 *~ 10;
#X obj 322 29 +~ 100;
#X obj 129 241 s~ \$0-osc1;
#X obj 240 237 s~ \$0-osc2;
#X obj 369 5 r~ \$0-osc1;
#X obj 231 5 r~ \$0-osc2;
#X msg 66 308 open -bytes 4 Lava-CrustyMix.wav \, start;
#X floatatom 225 33 5 0 0 0 - - -;
#X obj 39 119 +~;
#X obj 249 119 +~;
#X obj 141 108 line~;
#X msg 143 85 72 300000;
#X floatatom 116 84 5 0 0 0 - - -;
#X floatatom 238 51 5 0 0 0 - - -;
#X connect 0 0 5 0;
#X connect 0 0 25 0;
#X connect 1 0 31 0;
#X connect 2 0 0 0;
#X connect 5 0 3 0;
#X connect 6 0 3 0;
#X connect 7 0 6 1;
#X connect 7 0 14 1;
#X connect 8 0 5 1;
#X connect 8 0 13 1;
#X connect 9 0 1 1;
#X connect 9 0 11 1;
#X connect 10 0 13 0;
#X connect 10 0 26 0;
#X connect 11 0 32 0;
#X connect 12 0 10 0;
#X connect 13 0 15 0;
#X connect 14 0 15 0;
#X connect 16 0 24 1;
```

```
#X connect 18 0 17 0;
#X connect 19 0 1 0;
#X connect 19 0 6 0;
#X connect 19 0 4 0;
#X connect 19 0 17 0;
#X connect 20 0 11 0;
#X connect 20 0 14 0;
#X connect 20 0 4 1;
#X connect 20 0 17 1;
#X connect 21 0 22 0;
#X connect 22 0 19 0;
#X connect 23 0 24 0;
#X connect 24 0 20 0;
#X connect 27 0 23 0;
#X connect 28 0 21 0;
#X connect 29 0 17 0;
#X connect 30 0 21 1;
#X connect 30 0 23 1;
#X connect 31 0 2 0;
#X connect 32 0 12 0;
#X connect 33 0 31 1;
#X connect 33 0 32 1;
#X connect 34 0 33 0;
#X connect 35 0 33 0;
#X connect 36 0 22 1;
```

69 soft-rock-ep/Ejecta.pd

```
#N canvas 92 136 539 512 10;
#X obj 37 159 osc~;
#X obj 38 96 *~ 12;
#X obj 38 139 mtof~;
#X obj 72 208 delwrite~ \$0-delay-1 1000;
#X obj 69 247 dac~;
#X obj 55 183 *~ 0.9;
#X obj 105 183 *~ 0.1;
#X floatatom 142 138 5 0 0 0 - - -;
#X floatatom 95 141 5 0 0 0 - - -;
#X floatatom 93 39 5 0 0 0 - - -;
#X obj 247 159 osc~;
#X obj 248 96 *~ 12;
#X obj 248 139 mtof~;
#X obj 265 183 *~ 0.9;
#X obj 315 183 *~ 0.1;
#X obj 282 209 delwrite~ \$0-delay-2 1000;
#X floatatom 271 30 5 0 0 0 - - -;
#X obj 49 274 writesf~ 2;
#X msg 66 337 stop;
#X obj 101 60 vd~ \$0-delay-1;
#X obj 310 60 vd~ \$0-delay-2;
#X obj 173 6 *~ 10;
#X obj 172 29 +~ 100;
#X obj 323 6 *~ 10;
#X obj 322 29 +~ 100;
#X obj 129 241 s~ \$0-osc1;
#X obj 240 237 s~ \$0-osc2;
#X obj 369 5 r~ \$0-osc1;
```

```
#X obj 231 5 r~ \$0-osc2;
#X msg 66 308 open -bytes 4 Lava-CrustyMix.wav \, start;
#X floatatom 225 33 5 0 0 0 - - -;
#X obj 39 119 +~;
#X obj 249 119 +~;
#X obj 141 108 line~;
#X msg 143 85 72 300000;
#X floatatom 116 84 5 0 0 0 - - -;
#X connect 0 0 5 0;
#X connect 0 0 25 0;
#X connect 1 0 31 0;
#X connect 2 0 0 0;
#X connect 5 0 3 0;
#X connect 6 0 3 0;
#X connect 7 0 6 1;
#X connect 7 0 14 1;
#X connect 8 0 5 1;
#X connect 8 0 13 1;
#X connect 9 0 1 1;
#X connect 9 0 11 1;
#X connect 10 0 13 0;
#X connect 10 0 26 0;
#X connect 11 0 32 0;
#X connect 12 0 10 0;
#X connect 13 0 15 0;
#X connect 14 0 15 0;
#X connect 16 0 22 1;
#X connect 16 0 24 1;
#X connect 18 0 17 0;
#X connect 19 0 1 0;
#X connect 19 0 6 0;
#X connect 19 0 4 0;
#X connect 19 0 17 0;
#X connect 20 0 11 0;
#X connect 20 0 14 0;
#X connect 20 0 4 1;
#X connect 20 0 17 1;
#X connect 21 0 22 0;
#X connect 22 0 19 0;
#X connect 23 0 24 0;
#X connect 24 0 20 0;
#X connect 27 0 23 0;
#X connect 28 0 21 0;
#X connect 29 0 17 0;
#X connect 30 0 21 1;
#X connect 30 0 23 1;
#X connect 31 0 2 0;
#X connect 32 0 12 0;
#X connect 33 0 31 1;
#X connect 33 0 32 1;
#X connect 34 0 33 0;
#X connect 35 0 33 0;
```

70 soft-rock-ep/Lava-CrustyMix.pd

```
#N canvas 92 136 539 512 10;
#X obj 37 159 osc~;
```

```
#X obj 38 96 *~ 12;
#X obj 39 119 +~ 60;
#X obj 38 139 mtof~;
#X obj 72 208 delwrite~ \$0-delay-1 1000;
#X obj 69 247 dac~;
#X obj 55 183 *~ 0.9;
#X obj 105 183 *~ 0.1;
#X floatatom 142 138 5 0 0 0 - - -;
#X floatatom 95 141 5 0 0 0 - - -;
#X floatatom 130 118 5 0 0 0 - - -;
#X floatatom 130 89 5 0 0 0 - - -;
#X obj 247 159 osc~;
#X obj 248 96 *~ 12;
#X obj 249 119 +~ 60;
#X obj 248 139 mtof~;
#X obj 265 183 *~ 0.9;
#X obj 315 183 *~ 0.1;
#X obj 282 209 delwrite~ \$0-delay-2 1000;
#X floatatom 271 30 5 0 0 0 - - -;
#X obj 49 274 writesf~ 2;
#X msg 66 337 stop;
#X obj 101 60 vd~ \$0-delay-1;
#X obj 310 60 vd~ \$0-delay-2;
#X obj 173 6 *~ 10;
#X obj 172 29 +~ 100;
#X obj 323 6 *~ 10;
#X obj 322 29 +~ 100;
#X obj 129 241 s~ \$0-osc1;
#X obj 240 237 s~ \$0-osc2;
#X obj 369 5 r~ \$0-osc1;
#X obj 231 5 r~ \$0-osc2;
#X msg 66 308 open -bytes 4 Lava-CrustyMix.wav \, start;
#X connect 0 0 6 0;
#X connect 0 0 5 0;
#X connect 0 0 20 0;
#X connect 0 0 28 0;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 3 0 0 0;
#X connect 6 0 4 0;
#X connect 7 0 4 0;
#X connect 8 0 7 1;
#X connect 8 0 17 1;
#X connect 9 0 6 1;
#X connect 9 0 16 1;
#X connect 10 0 2 1;
#X connect 10 0 14 1;
#X connect 11 0 1 1;
#X connect 11 0 13 1;
#X connect 12 0 16 0;
#X connect 12 0 5 1;
#X connect 12 0 20 1;
#X connect 12 0 29 0;
#X connect 13 0 14 0;
#X connect 14 0 15 0;
#X connect 15 0 12 0;
#X connect 16 0 18 0;
```

```
#X connect 17 0 18 0;
#X connect 19 0 25 1;
#X connect 19 0 27 1;
#X connect 21 0 20 0;
#X connect 22 0 1 0;
#X connect 22 0 7 0;
#X connect 23 0 13 0;
#X connect 23 0 17 0;
#X connect 24 0 25 0;
#X connect 25 0 22 0;
#X connect 26 0 27 0;
#X connect 27 0 23 0;
#X connect 30 0 26 0;
#X connect 31 0 24 0;
#X connect 32 0 20 0;
```

71 soft-rock-ep/Magma-10000CMix.pd

```
#N canvas 0 0 450 300 10;
#X obj 54 130 osc~;
#X obj 55 157 s~ \$0-osc1;
#X obj 54 77 +~ 60;
#X obj 53 101 mtof~;
#X obj 253 129 osc~;
#X obj 253 76 +~ 60;
#X obj 252 100 mtof~;
#X obj 255 32 r~ \$0-osc1;
#X obj 54 29 r~ \$0-osc2;
#X obj 254 156 s~ \$0-osc2;
#X obj 101 210 dac~;
#X floatatom 143 104 5 0 0 0 - - -;
#X obj 150 58 line~;
#X obj 55 54 *~;
#X obj 253 53 *~;
#X msg 142 26 0 \, 72 300000;
#X obj 92 267 writesf~ 2;
#X msg 180 235 stop;
#X msg 163 212 open -bytes 4 Magma-10000CMix.wav \, start;
#X connect 0 0 1 0;
#X connect 0 0 10 0;
#X connect 0 0 16 0;
#X connect 2 0 3 0;
#X connect 3 0 0 0;
#X connect 4 0 9 0;
#X connect 4 0 10 1;
#X connect 4 0 16 1;
#X connect 5 0 6 0;
#X connect 6 0 4 0;
#X connect 7 0 14 0;
#X connect 8 0 13 0;
#X connect 11 0 2 1;
#X connect 11 0 5 1;
#X connect 12 0 13 1;
#X connect 12 0 14 1;
#X connect 13 0 2 0;
#X connect 14 0 5 0;
#X connect 15 0 12 0;
```

```
#X connect 17 0 16 0;
#X connect 18 0 16 0;
```

72 soft-rock-ep/Magma-120bpmMix.pd

```
#N canvas 92 136 539 512 10;
#X obj 37 159 osc~;
#X obj 100 60 delread~ \$0-delay-1 125;
#X obj 38 96 *~ 12;
#X obj 39 119 +~ 60;
#X obj 38 139 mtof~;
#X obj 72 208 delwrite~ \$0-delay-1 1000;
#X obj 69 247 dac~;
#X obj 55 183 *~ 0.9;
#X obj 105 183 *~ 0.1;
#X floatatom 142 138 5 0 0 0 - - -;
#X floatatom 95 141 5 0 0 0 - - -;
#X floatatom 130 118 5 0 0 0 - - -;
#X floatatom 130 89 5 0 0 0 - - -;
#X obj 247 159 osc~;
#X obj 248 96 *~ 12;
#X obj 249 119 +~ 60;
#X obj 248 139 mtof~;
#X obj 265 183 *~ 0.9;
#X obj 315 183 *~ 0.1;
#X obj 282 209 delwrite~ \$0-delay-2 1000;
#X obj 310 60 delread~ \$0-delay-2 125;
#X floatatom 271 30 5 0 0 0 - - -;
#X obj 49 274 writesf~ 2;
#X msg 66 308 open -bytes 4 Magma-120bpmMix.wav \, start;
#X msg 66 337 stop;
#X connect 0 0 7 0;
#X connect 0 0 6 0;
#X connect 0 0 22 0;
#X connect 1 0 2 0;
#X connect 1 0 8 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 4 0 0 0;
#X connect 7 0 5 0;
#X connect 8 0 5 0;
#X connect 9 0 8 1;
#X connect 9 0 18 1;
#X connect 10 0 7 1;
#X connect 10 0 17 1;
#X connect 11 0 3 1;
#X connect 11 0 15 1;
#X connect 12 0 2 1;
#X connect 12 0 14 1;
#X connect 13 0 17 0;
#X connect 13 0 6 1;
#X connect 13 0 22 1;
#X connect 14 0 15 0;
#X connect 15 0 16 0;
#X connect 16 0 13 0;
#X connect 17 0 19 0;
#X connect 18 0 19 0;
```

```
#X connect 20 0 14 0;
#X connect 20 0 18 0;
#X connect 21 0 1 0;
#X connect 21 0 20 0;
#X connect 23 0 22 0;
#X connect 24 0 22 0;
```

73 soft-rock-ep/Magma-CoolMix.pd

```
#N canvas 0 0 450 300 10;
#X obj 54 130 osc~;
#X obj 55 157 s~ \$0-osc1;
#X obj 54 77 +~ 60;
#X obj 53 101 mtof~;
#X obj 253 129 osc~;
#X obj 253 76 +~ 60;
#X obj 252 100 mtof~;
#X obj 255 32 r~ \$0-osc1;
#X obj 54 29 r~ \$0-osc2;
#X obj 254 156 s~ \$0-osc2;
#X obj 101 210 dac~;
#X floatatom 143 104 5 0 0 0 - - -;
#X obj 150 58 line~;
#X obj 55 54 *~;
#X obj 253 53 *~;
#X obj 92 267 writesf~ 2;
#X msg 180 235 stop;
#X msg 201 71 0;
#X msg 163 212 open -bytes 4 Magma-CoolMix.wav \, start;
#X msg 142 26 72 \, 0 300000;
#X connect 0 0 1 0;
#X connect 0 0 10 0;
#X connect 0 0 15 0;
#X connect 2 0 3 0;
#X connect 3 0 0 0;
#X connect 4 0 9 0;
#X connect 4 0 10 1;
#X connect 4 0 15 1;
#X connect 5 0 6 0;
#X connect 6 0 4 0;
#X connect 7 0 14 0;
#X connect 8 0 13 0;
#X connect 11 0 2 1;
#X connect 11 0 5 1;
#X connect 12 0 13 1;
#X connect 12 0 14 1;
#X connect 13 0 2 0;
#X connect 14 0 5 0;
#X connect 16 0 15 0;
#X connect 17 0 12 0;
#X connect 18 0 15 0;
#X connect 19 0 12 0;
```

74 soft-rock-ep/Magma-HotMix.pd

```
#N canvas 0 0 707 413 10;
```

```
#X obj 54 130 osc~;
#X obj 55 157 s~ \$0-osc1;
#X obj 53 101 mtof~;
#X obj 253 129 osc~;
#X obj 252 100 mtof~;
#X obj 255 32 r~ \$0-osc1;
#X obj 254 156 s~ \$0-osc2;
#X obj 150 58 line~;
#X obj 55 54 *~;
#X obj 253 53 *~;
#X obj 92 267 writesf~ 2;
#X msg 180 235 stop;
#X obj 150 118 line~;
#X obj 54 77 +~;
#X obj 253 76 +~;
#X msg 163 212 open -bytes 4 Magma-HotMix.wav \, start;
#X obj 344 130 osc~;
#X obj 343 101 mtof~;
#X obj 543 129 osc~;
#X obj 542 100 mtof~;
#X obj 344 29 r~ \$0-osc2;
#X obj 345 54 *~;
#X obj 543 53 *~;
#X obj 344 77 +~;
#X obj 543 76 +~;
#X obj 345 157 s~ \$0-osc3;
#X obj 545 32 r~ \$0-osc3;
#X obj 543 155 s~ \$0-osc4;
#X obj 54 29 r~ \$0-osc4;
#X obj 149 93 pack f 100;
#X obj 149 33 pack f 100;
#X floatatom 115 73 5 0 0 0 - - -;
#X floatatom 134 5 5 0 0 0 - - -;
#X obj 84 190 *~ 0.5;
#X obj 85 224 dac~;
#X obj 134 190 *~ 0.5;
#X connect 0 0 1 0;
#X connect 0 0 33 0;
#X connect 2 0 0 0;
#X connect 3 0 6 0;
#X connect 3 0 35 0;
#X connect 4 0 3 0;
#X connect 5 0 9 0;
#X connect 7 0 8 1;
#X connect 7 0 9 1;
#X connect 7 0 21 1;
#X connect 7 0 22 1;
#X connect 8 0 13 0;
#X connect 9 0 14 0;
#X connect 11 0 10 0;
#X connect 12 0 13 1;
#X connect 12 0 14 1;
#X connect 12 0 23 1;
#X connect 12 0 24 1;
#X connect 13 0 2 0;
#X connect 14 0 4 0;
#X connect 15 0 10 0;
```

```
#X connect 16 0 25 0;
#X connect 16 0 33 0;
#X connect 17 0 16 0;
#X connect 18 0 27 0;
#X connect 18 0 35 0;
#X connect 19 0 18 0;
#X connect 20 0 21 0;
#X connect 21 0 23 0;
#X connect 22 0 24 0;
#X connect 23 0 17 0;
#X connect 24 0 19 0;
#X connect 26 0 22 0;
#X connect 28 0 8 0;
#X connect 29 0 12 0;
#X connect 30 0 7 0;
#X connect 31 0 29 0;
#X connect 32 0 30 0;
#X connect 33 0 34 0;
#X connect 33 0 10 0;
#X connect 35 0 34 1;
#X connect 35 0 10 1;
```

75 soft-rock-ep/Magma-WarmMix.pd

```
#N canvas 0 0 707 413 10;
#X obj 54 130 osc~;
#X obj 55 157 s~ \$0-osc1;
#X obj 53 101 mtof~;
#X obj 253 129 osc~;
#X obj 252 100 mtof~;
#X obj 255 32 r~ \$0-osc1;
#X obj 254 156 s~ \$0-osc2;
#X obj 150 58 line~;
#X obj 55 54 *~;
#X obj 253 53 *~;
#X obj 92 267 writesf~ 2;
#X msg 180 235 stop;
#X obj 150 118 line~;
#X obj 54 77 +~;
#X obj 253 76 +~;
#X msg 163 212 open -bytes 4 Magma-HotMix.wav \, start;
#X obj 344 130 osc~;
#X obj 343 101 mtof~;
#X obj 543 129 osc~;
#X obj 542 100 mtof~;
#X obj 344 29 r~ \$0-osc2;
#X obj 345 54 *~;
#X obj 543 53 *~;
#X obj 344 77 +~;
#X obj 543 76 +~;
#X obj 345 157 s~ \$0-osc3;
#X obj 545 32 r~ \$0-osc3;
#X obj 543 155 s~ \$0-osc4;
#X obj 54 29 r~ \$0-osc4;
#X obj 149 93 pack f 100;
#X obj 149 33 pack f 100;
#X floatatom 115 73 5 0 0 0 - - -;
```

```
#X floatatom 134 5 5 0 0 0 - - -;
#X obj 84 190 *~ 0.5;
#X obj 85 224 dac~;
#X obj 134 190 *~ 0.5;
#X connect 0 0 1 0;
#X connect 0 0 33 0;
#X connect 2 0 0 0;
#X connect 3 0 6 0;
#X connect 3 0 35 0;
#X connect 4 0 3 0;
#X connect 5 0 9 0;
#X connect 7 0 8 1;
#X connect 7 0 9 1;
#X connect 7 0 21 1;
#X connect 7 0 22 1;
#X connect 8 0 13 0;
#X connect 9 0 14 0;
#X connect 11 0 10 0;
#X connect 12 0 13 1;
#X connect 12 0 14 1;
#X connect 12 0 23 1;
#X connect 12 0 24 1;
#X connect 13 0 2 0;
#X connect 14 0 4 0;
#X connect 15 0 10 0;
#X connect 16 0 25 0;
#X connect 16 0 33 0;
#X connect 17 0 16 0;
#X connect 18 0 27 0;
#X connect 18 0 35 0;
#X connect 19 0 18 0;
#X connect 20 0 21 0;
#X connect 21 0 23 0;
#X connect 22 0 24 0;
#X connect 23 0 17 0;
#X connect 24 0 19 0;
#X connect 26 0 22 0;
#X connect 28 0 8 0;
#X connect 29 0 12 0;
#X connect 30 0 7 0;
#X connect 31 0 29 0;
#X connect 32 0 30 0;
#X connect 33 0 34 0;
#X connect 33 0 10 0;
#X connect 35 0 34 1;
#X connect 35 0 10 1;
```

76 soft-rock-ep/Quicksand.pd

```
#N canvas 0 0 1255 690 10;
#X obj 4 27 delay 100;
#X obj 310 2 bng 15 250 50 0 empty empty empty 0 -6 0 8 -262144 -1
-1;
#X obj 93 63 t b b;
#X obj 95 211 vline~;
#X obj 77 237 *~;
#X obj 37 198 osc~;
```

```
#X obj 37 174 mtof;
#X floatatom 9 144 5 0 0 0 - - -;
#X msg 96 184 1 5 \, 0 \$1 10;
#X obj 96 158 + 10;
#X obj 61 138 + 15;
#X obj 394 27 delay 100;
#X obj 483 63 t b b;
#X obj 485 211 vline~;
#X obj 467 237 *~;
#X obj 427 198 osc~;
#X obj 427 174 mtof;
#X floatatom 399 144 5 0 0 0 - - -;
#X obj 486 158 + 10;
#X obj 451 138 + 15;
#X obj 484 127 / 4;
#X obj 94 127 / 4;
#X obj 91 88 random 150;
#X obj 481 88 random 150;
#X msg 486 184 1 5 \, 0 \$1 5;
#X obj 584 27 delay 100;
#X obj 673 63 t b b;
#X obj 675 211 vline~;
#X obj 657 237 *~;
#X obj 617 198 osc~;
#X obj 617 174 mtof;
#X floatatom 589 144 5 0 0 0 - - -;
#X obj 676 158 + 10;
#X obj 641 138 + 15;
#X obj 674 127 / 4;
#X obj 671 88 random 150;
#X msg 676 184 1 5 \, 0 \$1 5;
#X obj 204 27 delay 100;
#X obj 293 63 t b b;
#X obj 295 211 vline~;
#X obj 277 237 *~;
#X obj 237 198 osc~;
#X obj 237 174 mtof;
#X floatatom 209 144 5 0 0 0 - - -;
#X msg 296 184 1 5 \, 0 \$1 10;
#X obj 296 158 + 10;
#X obj 261 138 + 15;
#X obj 294 127 / 4;
#X obj 291 88 random 150;
#X obj 278 651 dac~;
#X floatatom 342 -2 5 0 0 0 - - -;
#X obj 24 307 delay 100;
#X obj 330 282 bng 15 250 50 0 empty empty empty 0 -6 0 8 -262144 -1
-1;
#X obj 113 343 t b b;
#X obj 115 491 vline~;
#X obj 97 517 *~;
#X obj 57 478 osc~;
#X obj 57 454 mtof;
#X floatatom 29 424 5 0 0 0 - - -;
#X msg 116 464 1 5 \, 0 \$1 10;
#X obj 116 438 + 10;
#X obj 81 418 + 15;
```

```
#X obj 414 307 delay 100;
#X obj 503 343 t b b;
#X obj 505 491 vline~;
#X obj 487 517 *~;
#X obj 447 478 osc~;
#X obj 447 454 mtof;
#X floatatom 419 424 5 0 0 0 - - -;
#X obj 506 438 + 10;
#X obj 471 418 + 15;
#X obj 504 407 / 4;
#X obj 114 407 / 4;
#X obj 111 368 random 150;
#X obj 501 368 random 150;
#X msg 506 464 1 5 \, 0 \$1 5;
#X obj 604 307 delay 100;
#X obj 693 343 t b b;
#X obj 695 491 vline~;
#X obj 677 517 *~;
#X obj 637 478 osc~;
#X obj 637 454 mtof;
#X floatatom 609 424 5 0 0 0 - - -;
#X obj 696 438 + 10;
#X obj 661 418 + 15;
#X obj 694 407 / 4;
#X obj 691 368 random 150;
#X msg 696 464 1 5 \, 0 \$1 5;
#X obj 224 307 delay 100;
#X obj 313 343 t b b;
#X obj 315 491 vline~;
#X obj 297 517 *~;
#X obj 257 478 osc~;
#X obj 257 454 mtof;
#X floatatom 216 423 5 0 0 0 - - -;
#X msg 316 464 1 5 \, 0 \$1 10;
#X obj 316 438 + 10;
#X obj 281 418 + 15;
#X obj 314 407 / 4;
#X obj 311 368 random 150;
#X floatatom 362 278 5 0 0 0 - - -;
#X obj 325 603 /~ 4;
#X obj 265 603 /~ 4;
#X floatatom 292 277 5 0 0 0 - - -;
#X floatatom 264 -4 5 0 0 0 - - -;
#X connect 0 0 2 0;
#X connect 1 0 0 0;
#X connect 1 0 11 0;
#X connect 1 0 25 0;
#X connect 1 0 37 0;
#X connect 1 0 52 0;
#X connect 2 0 0 0;
#X connect 2 1 22 0;
#X connect 3 0 4 1;
#X connect 4 0 102 0;
#X connect 5 0 4 0;
#X connect 6 0 5 0;
#X connect 7 0 6 0;
#X connect 8 0 3 0;
```

```
#X connect 9 0 8 0;
#X connect 10 0 0 1;
#X connect 11 0 12 0;
#X connect 12 0 11 0;
#X connect 12 1 23 0;
#X connect 13 0 14 1;
#X connect 14 0 101 0;
#X connect 15 0 14 0;
#X connect 16 0 15 0;
#X connect 17 0 16 0;
#X connect 18 0 24 0;
#X connect 19 0 11 1;
#X connect 20 0 18 0;
#X connect 21 0 9 0;
#X connect 22 0 10 0;
#X connect 22 0 21 0;
#X connect 23 0 19 0;
#X connect 23 0 20 0;
#X connect 24 0 13 0;
#X connect 25 0 26 0;
#X connect 26 0 25 0;
#X connect 26 1 35 0;
#X connect 27 0 28 1;
#X connect 28 0 101 0;
#X connect 29 0 28 0;
#X connect 30 0 29 0;
#X connect 31 0 30 0;
#X connect 32 0 36 0;
#X connect 33 0 25 1;
#X connect 34 0 32 0;
#X connect 35 0 33 0;
#X connect 35 0 34 0;
#X connect 36 0 27 0;
#X connect 37 0 38 0;
#X connect 38 0 37 0;
#X connect 38 1 48 0;
#X connect 39 0 40 1;
#X connect 40 0 102 0;
#X connect 41 0 40 0;
#X connect 42 0 41 0;
#X connect 43 0 42 0;
#X connect 44 0 39 0;
#X connect 45 0 44 0;
#X connect 46 0 37 1;
#X connect 47 0 45 0;
#X connect 48 0 46 0;
#X connect 48 0 47 0;
#X connect 50 0 48 1;
#X connect 50 0 22 1;
#X connect 50 0 23 1;
#X connect 50 0 35 1;
#X connect 50 0 100 0;
#X connect 51 0 53 0;
#X connect 52 0 51 0;
#X connect 52 0 62 0;
#X connect 52 0 76 0;
#X connect 52 0 88 0;
```

```
#X connect 53 0 51 0;
#X connect 53 1 73 0;
#X connect 54 0 55 1;
#X connect 55 0 102 0;
#X connect 56 0 55 0;
#X connect 57 0 56 0;
#X connect 58 0 57 0;
#X connect 59 0 54 0;
#X connect 60 0 59 0;
#X connect 61 0 51 1;
#X connect 62 0 63 0;
#X connect 63 0 62 0;
#X connect 63 1 74 0;
#X connect 64 0 65 1;
#X connect 65 0 101 0;
#X connect 66 0 65 0;
#X connect 67 0 66 0;
#X connect 68 0 67 0;
#X connect 69 0 75 0;
#X connect 70 0 62 1;
#X connect 71 0 69 0;
#X connect 72 0 60 0;
#X connect 73 0 61 0;
#X connect 73 0 72 0;
#X connect 74 0 70 0;
#X connect 74 0 71 0;
#X connect 75 0 64 0;
#X connect 76 0 77 0;
#X connect 77 0 76 0;
#X connect 77 1 86 0;
#X connect 78 0 79 1;
#X connect 79 0 101 0;
#X connect 80 0 79 0;
#X connect 81 0 80 0;
#X connect 82 0 81 0;
#X connect 83 0 87 0;
#X connect 84 0 76 1;
#X connect 85 0 83 0;
#X connect 86 0 84 0;
#X connect 86 0 85 0;
#X connect 87 0 78 0;
#X connect 88 0 89 0;
#X connect 89 0 88 0;
#X connect 89 1 99 0;
#X connect 90 0 91 1;
#X connect 91 0 102 0;
#X connect 92 0 91 0;
#X connect 93 0 92 0;
#X connect 94 0 93 0;
#X connect 95 0 90 0;
#X connect 96 0 95 0;
#X connect 97 0 88 1;
#X connect 98 0 96 0;
#X connect 99 0 97 0;
#X connect 99 0 98 0;
#X connect 100 0 99 1;
#X connect 100 0 73 1;
```

```
#X connect 100 0 74 1;
#X connect 100 0 86 1;
#X connect 101 0 49 1;
#X connect 102 0 49 0;
#X connect 103 0 61 1;
#X connect 103 0 97 1;
#X connect 103 0 70 1;
#X connect 103 0 84 1;
#X connect 104 0 10 1;
#X connect 104 0 46 1;
#X connect 104 0 19 1;
#X connect 104 0 33 1;
#X connect 104 0 103 0;
```