tilde

Claude Heiland-Allen

2011–2019

Contents

1 compress~.pd

```
#N canvas 0 0 309 587 10;
#X obj 58 71 hip~ 5;
#X obj 121 71 hip~ 5;
#X obj 58 97 *~;
#X obj 120 98 *~;
#X obj 87 158 +~;
#X obj 87 281 rmstodb~;
#X obj 86 388 dbtorms~;
#X obj 27 474 *~;
#X obj 177 474 *~;
#X obj 120 123 lop~ 10;
#X obj 57 124 lop~ 10;
#X obj 87 221 lop~ 25;
#X obj 25 272 /~;
#X obj 178 270 /~;
#X obj 87 179 sqrt~;
#X obj 86 442 *~ 0.25;
#X obj 88 202 +~ 0.01;
#X obj 27 514 tanh~;
#X obj 177 514 tanh~;
#X obj 110 28 adc~;
#X obj 105 546 dac~;
#X obj 85 361 +~ 100;
#X obj 86 314 /~ 16;
#X connect 0 0 2 0;
```

```
#X connect 0 0 2 1;
#X connect 1 0 3 0;
#X connect 1 0 3 1;
#X connect 2 0 10 0;
#X connect 3 0 9 0;
#X connect 4 0 14 0;
#X connect 5 0 22 0;
#X connect 6 0 15 0;
#X connect 7 0 17 0;
#X connect 8 0 18 0;
#X connect 9 0 4 1;
#X connect 10 0 4 0;
#X connect 11 0 12 1;
#X connect 11 0 13 1;
#X connect 11 0 5 0;
#X connect 12 0 7 0;
#X connect 13 0 8 0;
#X connect 14 0 16 0;
#X connect 15 0 7 1;
#X connect 15 0 8 1;
#X connect 16 0 11 0;
#X connect 17 0 20 0;
#X connect 18 0 20 1;
#X connect 19 0 0 0;
#X connect 19 0 12 0;
#X connect 19 1 1 0;
#X connect 19 1 13 0;
#X connect 21 0 6 0;
#X connect 22 0 21 0;
```

2 .gitignore

```
dist
dist-newstyle
.cabal-sandbox
cabal.sandbox.config
tilde
*.hi
*.o
```

3 GOSUB10-004-src-0xA–expr_.diff

```
diff -ruw GOSUB10-004-src-0xA--expr_.orig/0903-4.pd GOSUB10-004-src-0xA--expr_.↙
    ↘ new/0903-4.pd
--- GOSUB10-004-src-0xA--expr_.orig/0903-4.pd    2011-03-27 14:16:09.000000000 ↙
    ↘ +0100
+++ GOSUB10-004-src-0xA--expr_.new/0903-4.pd     2011-08-28 11:46:15.000000000 ↙
    ↘ +0100
@@ -52,8 +52,8 @@
 #X obj 62 119 expr~ if($v1==0 \, 0 \, $v2);
 #X obj 245 298 expr~ fmod($v1*8 \, 1);
 #X obj 543 292 sig~ 0.01;
-#X obj 61 36 r~ bass1-a;
-#X obj 152 38 r~ bass1-b;
+#X obj 61 36 r~ bass1_a;
+#X obj 152 38 r~ bass1_b;
 #X obj 62 475 outlet~;
```

```
 #X obj 62 236 phasor~;
 #X obj 62 264 expr~ if($v1>$v2 \, 1 \, -1);
@@ -118,8 +118,8 @@
 #X obj 13 255 expr~ if($v1==0 \, $v3 \, if($v1==1 \, $v2 \, if($v1==2
 \, 7 \, 12)));
 #X obj 14 625 +~ 48;
-#X obj 393 342 s~ bass1-b;
-#X obj 76 304 s~ bass1-a;
+#X obj 393 342 s~ bass1_b;
+#X obj 76 304 s~ bass1_a;
 #X obj 13 133 r~ clock;
 #X obj 271 103 r~ clock;
 #X obj 13 780 outlet~;
@@ -300,7 +300,7 @@
 #X obj 187 171 sig~ 0;
 #X floatatom 186 141 5 0 0 0 - - -;
 #X obj 27 171 expr~ fmod($v1+$v2 \, 1);
-#X obj 162 265 r~ bass1-b;
+#X obj 162 265 r~ bass1_b;
 #X obj 7 -85 r~ clock;
 #X obj 8 109 expr~ fmod($v1*$v2 \, 1);
 #X obj 9 369 lop~ 20;
@@ -328,7 +328,7 @@
 #X obj 9 448 *~ 0.3;
 #X obj 508 463 *~ 0.3;
 #X obj 505 159 expr~ fmod($v1+$v2 \, 1);
-#X obj 704 258 r~ bass1-b;
+#X obj 704 258 r~ bass1_b;
 #X obj 781 198 expr~ fmod($v1+$v2 \, 1);
 #X obj 507 363 lop~ 15;
 #X obj 667 -10 expr~ fmod($v1*$v2 \, 1);
diff -ruw GOSUB10-004-src-0xA--expr_.orig/family_mart.pd GOSUB10-004-src-0xA--↙
    ↪ expr_.new/family_mart.pd
--- GOSUB10-004-src-0xA--expr_.orig/family_mart.pd      2011-03-27 ↙
    ↪ 14:16:09.000000000 +0100
+++ GOSUB10-004-src-0xA--expr_.new/family_mart.pd      2011-08-28 ↙
    ↪ 11:57:06.000000000 +0100
@@ -656,7 +656,7 @@
 #X restore 27 41 pd synth;
 #N canvas 656 47 638 266 struct 0;
 #X obj 26 161 expr~ floor($v1*16 \, 0);
-#X obj 26 74 phasor~ 0.25;
+#X obj 26 74 phasor~;
 #X obj 26 53 /~;
 #X obj 25 111 expr~ fmod($v1*$v2 \, 1);
 #X obj 25 14 sig~ 0.25;
diff -ruw GOSUB10-004-src-0xA--expr_.orig/moving_souf.pd GOSUB10-004-src-0xA--↙
    ↪ expr_.new/moving_souf.pd
--- GOSUB10-004-src-0xA--expr_.orig/moving_souf.pd      2011-03-27 ↙
    ↪ 14:16:09.000000000 +0100
+++ GOSUB10-004-src-0xA--expr_.new/moving_souf.pd      2011-08-28 ↙
    ↪ 11:45:17.000000000 +0100
@@ -9,7 +9,7 @@
 #X obj 36 86 expr~ fmod($v1*40 \, 1);
 #X obj 194 81 expr~ floor($v1*40 \, 0)%4;
 #X obj 346 81 expr~ floor($v1*10 \, 0);
-#X obj 487 113 s~ sig4x2-riff;
```

```
+#X obj 487 113 s~ sig4x2_riff;
 #X obj 622 36 r~ sig4x2;
 #X obj 486 82 expr~ floor($v1*$v2 \, 0);
 #X obj 197 194 s~ kchange;
@@ -295,7 +295,7 @@
 #X obj 687 66 expr~ if($v1<3 \, 0 \, 1);
 #X obj 186 320 expr~ if(($v1+$v3)%2 || ($v1==3|| $v1==5) \, fmod($v2*32
 \, 1) \, fmod($v2*1 \, 1));
-#X obj 244 9 r~ sig4x2-riff;
+#X obj 244 9 r~ sig4x2_riff;
 #X obj 81 226 r~ kchange;
 #X obj 36 239 +~;
 #X connect 0 0 23 0;
diff -ruw GOSUB10-004-src-0xA--expr_.orig/nandsynth.pd GOSUB10-004-src-0xA--↙
    ↪ expr_.new/nandsynth.pd
--- GOSUB10-004-src-0xA--expr_.orig/nandsynth.pd         2011-03-27 ↙
    ↪ 14:16:09.000000000 +0100
+++ GOSUB10-004-src-0xA--expr_.new/nandsynth.pd 2011-08-28 12:06:38.000000000 ↙
    ↪ +0100
@@ -1,21 +1,21 @@
 #N canvas 391 487 415 196 10;
 #N canvas 371 136 800 387 synth 0;
-#X obj 26 74 phasor~ 1;
+#X obj 26 74 phasor~;
 #X obj 26 93 expr~ if($v1<0.5 \, 0 \, 1);
-#X obj 170 74 phasor~ 1;
+#X obj 170 74 phasor~;
 #X obj 170 93 expr~ if($v1<0.5 \, 0 \, 1);
 #X obj 24 322 dac~;
-#X obj 352 47 phasor~ 1;
+#X obj 352 47 phasor~;
 #X obj 352 73 expr~ if($v1<0.5 \, 0 \, 1);
-#X obj 496 47 phasor~ 1;
+#X obj 496 47 phasor~;
 #X obj 496 73 expr~ if($v1<0.5 \, 0 \, 1);
-#X obj 141 164 phasor~ 1;
+#X obj 141 164 phasor~;
 #X obj 141 190 expr~ if($v1<0.5 \, 0 \, 1);
-#X obj 285 164 phasor~ 1;
+#X obj 285 164 phasor~;
 #X obj 285 190 expr~ if($v1<0.5 \, 0 \, 1);
-#X obj 439 170 phasor~ 1;
+#X obj 439 170 phasor~;
 #X obj 439 190 expr~ if($v1<0.5 \, 0 \, 1);
-#X obj 583 170 phasor~ 1;
+#X obj 583 170 phasor~;
 #X obj 583 190 expr~ if($v1<0.5 \, 0 \, 1);
 #X obj 27 45 *~;
 #X obj 170 55 *~;
diff -ruw GOSUB10-004-src-0xA--expr_.orig/winter_solstice.pd GOSUB10-004-src-0xA↙
    ↪ --expr_.new/winter_solstice.pd
--- GOSUB10-004-src-0xA--expr_.orig/winter_solstice.pd  2011-03-27 ↙
    ↪ 14:16:09.000000000 +0100
+++ GOSUB10-004-src-0xA--expr_.new/winter_solstice.pd   2011-08-28 ↙
    ↪ 11:43:44.000000000 +0100
@@ -5,17 +5,17 @@
 #X obj 183 127 s~ phrase_freq;
```

```
115  #X obj 174 243 s~ phrase_count;
     #X obj 85 218 s~ phrase_sig;
    -#X obj 85 63 s~ MAX-BAR;
    -#X obj 111 127 r~ MAX-BAR;
    +#X obj 85 63 s~ MAX_BAR;
120 +#X obj 111 127 r~ MAX_BAR;
     #X obj 85 127 /~;
     #X obj 85 169 phasor~;
     #X obj 85 82 sig~ 0.36;
     #X obj 183 108 /~ 16;
125  #X obj 85 194 expr~ fmod($v1*$v2 \, 1);
    -#X obj 215 150 r~ MAX-BAR;
    +#X obj 215 150 r~ MAX_BAR;
     #X obj 215 175 /~ 8;
     #X obj 343 215 s~ note_count;
130 -#X obj 478 167 r~ MAX-BAR;
    +#X obj 478 167 r~ MAX_BAR;
     #X obj 344 188 expr~ floor($v1*$v2 \, 0);
     #X msg 361 121 0;
     #X obj 85 43 sig~ 73;
```

4 make-example-tarball.sh

```
#!/bin/bash
set -e
z="$(pwd)"
h="$(dirname "$(readlink -f "${0}")")"
5  o="GOSUB10-004-src-0xA--expr_"
e="tilde-examples-0.1"
t="$(mktemp -d)"
cd "${t}"
mkdir "${e}"
10 wget "http://www.archive.org/download/GOSUB10-004/${o}.tgz"
tar xzf "${o}.tgz"
patch -d "${o}" < "${h}/${o}.diff"
cd "${h}"
for p in "${t}/${o}"/*.pd
15 do
  runghc tilde.hs "${p}" > "${t}/${e}/$(basename "${p%pd}cc")"
  cp -avit "${t}/${e}/" "${p}"
done
cp -avit "${t}/${e}/" "puredata.h" "puredata.c" "sample.h"
20 cp -avi  "Makefile.examples" "${t}/${e}/Makefile"
cp -avi  "README.examples"   "${t}/${e}/README.tilde"
cp -avit "${t}/${e}/" "${t}/${o}/COPYING" "${t}/${o}/README"
mkdir -p dist
cd "${t}"
25 tar --create --verbose --owner=0 --group=0 --gzip --file="${h}/dist/${e}.tgz" "$↙
    ↳ {e}/"
cd "${z}"
```

5 Makefile.examples

```
EXES=0903-4 family_mart kuroshio moving_souf nandsynth winter_solstice

all: $(EXES)
```

```
clean:
	-rm -f $(EXES)

%: %.cc puredata.c puredata.h sample.h
	g++ -O3 -march=native -s -Wall -pedantic -Wextra -Wno-unused -xc++ -
	    ↪ ljack -o $@ $<

.SUFFIXES:
.PHONY: all clean
```

6 NEWS

```
0.1	2011-08-28	expr~ss yourself
	initial release of 'tilde': Pure-data to C++ compiler
	can compile some lightly-modified real-world patches!
```

7 pitchshift~.pd

```
#N canvas 0 0 450 406 10;
#X obj 185 26 adc~;
#X obj 17 291 cos~;
#X obj 17 318 *~;
#X obj 17 347 +~;
#X obj 116 194 wrap~;
#X obj 117 292 cos~;
#X obj 117 319 *~;
#X obj 116 171 +~ 0.5;
#X obj 17 239 -~ 0.5;
#X obj 17 265 *~ 0.5;
#X obj 117 231 -~ 0.5;
#X obj 117 264 *~ 0.5;
#X obj 60 239 *~ 50;
#X obj 60 265 +~ 5;
#X obj 160 232 *~ 50;
#X obj 160 265 +~ 5;
#X obj 227 318 *~;
#X obj 227 347 +~;
#X obj 327 319 *~;
#X obj 59 291 vd~ l;
#X obj 160 294 vd~ l;
#X obj 270 291 vd~ r;
#X obj 370 294 vd~ r;
#X obj 186 53 delwrite~ l 100;
#X obj 296 53 delwrite~ r 100;
#X obj 117 370 dac~;
#X obj 20 63 hsl 128 15 -48 48 0 0 empty empty transpose -2 -8 0 10
-262144 -1 -1 6350 1;
#X obj 17 83 * 0.05776;
#X obj 17 104 exp;
#X obj 17 125 - 1;
#X obj 17 146 * -1;
#X obj 17 167 / 0.05;
#X obj 17 7 loadbang;
#X msg 17 28 0;
#X obj 17 197 phasor~ 10;
#X connect 0 0 23 0;
#X connect 0 1 24 0;
```

```
#X connect 1 0 2 0;
#X connect 1 0 16 0;
#X connect 2 0 3 0;
#X connect 3 0 25 0;
#X connect 4 0 10 0;
#X connect 4 0 14 0;
#X connect 5 0 6 0;
#X connect 5 0 18 0;
#X connect 6 0 3 1;
#X connect 7 0 4 0;
#X connect 8 0 9 0;
#X connect 9 0 1 0;
#X connect 10 0 11 0;
#X connect 11 0 5 0;
#X connect 12 0 13 0;
#X connect 13 0 19 0;
#X connect 13 0 21 0;
#X connect 14 0 15 0;
#X connect 15 0 20 0;
#X connect 15 0 22 0;
#X connect 16 0 17 0;
#X connect 17 0 25 1;
#X connect 18 0 17 1;
#X connect 19 0 2 1;
#X connect 20 0 6 1;
#X connect 21 0 16 1;
#X connect 22 0 18 1;
#X connect 26 0 27 0;
#X connect 27 0 28 0;
#X connect 28 0 29 0;
#X connect 29 0 30 0;
#X connect 30 0 31 0;
#X connect 32 0 33 0;
#X connect 33 0 26 0;
#X connect 34 0 8 0;
#X connect 34 0 7 0;
#X connect 34 0 12 0;
```

8 puredata.c

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <getopt.h>
#include <jack/jack.h>

static jack_port_t *input_port[2];
static jack_port_t *output_port[2];
static jack_client_t *client;

static int process(jack_nframes_t nframes, void *arg) {
  state *s = (state *) arg;
  jack_default_audio_sample_t *in[2], *out[2];
  in [0] = (jack_default_audio_sample_t *)jack_port_get_buffer(input_port [0], ↙
      ↘ nframes);
  in [1] = (jack_default_audio_sample_t *)jack_port_get_buffer(input_port [1], ↙
      ↘ nframes);
```

```
  out[0] = (jack_default_audio_sample_t *)jack_port_get_buffer(output_port[0], ↙
      ↳ nframes);
  out[1] = (jack_default_audio_sample_t *)jack_port_get_buffer(output_port[1], ↙
      ↳ nframes);
  for (jack_nframes_t i = 0; i < nframes; ++i) {
    s->adc[0] = in[0][i];
    s->adc[1] = in[1][i];
    s->dac[0] = 0;
    s->dac[1] = 0;
    dsp(s);
    out[0][i] = s->dac[0].x;
    out[1][i] = s->dac[1].x;
  }
  return 0;
}

static void jack_shutdown(void *arg) {
  exit(1);
}

int main(int argc, char **argv) {

  /* parse arguments */
  const char *client_name = 0;
  const char *server_name = 0;
  jack_options_t options = JackNullOption;
  jack_status_t status;
  while (1) {
    static struct option long_options[] = {
      { "help",    no_argument,       0, 'h' },
      { "client",  required_argument, 0, 'c' },
      { "server",  required_argument, 0, 's' },
      { 0, 0, 0, 0 }
    };
    int option_index = 0;
    int c = getopt_long(argc, argv, "hc:s:", long_options, &option_index);
    if (c == -1) {
      break;
    }
    switch (c) {
      case 'h':
        fprintf(stderr,
          "Usage: %s [options]\n"
          "Optional:\n"
          "  --client  -c <str>  JACK client name\n"
          "  --server  -s <str>  JACK server name\n"
          "Information:\n"
          "  --help    -h        display this information\n"
          , argv[0]
        ); return 0; break;
      case 'c': client_name = optarg; break;
      case 's': server_name = optarg; break;
      case '?': break;
      default: exit(1); break;
    }
  }
  if (optind < argc) {
```

```
      fprintf(stderr, "excess arguments, try --help\n");
      return 1;
    }
    if (!client_name) {
      client_name = strrchr(argv[0], '/');
      if (!client_name) {
        client_name = argv[0];
      } else {
        client_name++;
      }
    }
    if (server_name) {
      options = (jack_options_t) (options | JackServerName);
    }

    /* connect to JACK */
    client = jack_client_open(client_name, options, &status, server_name);
    if (!client) {
      fprintf(stderr, "jack_client_open() failed, status = 0x%2.0x\n", status);
      if (status & JackServerFailed) {
        fprintf(stderr, "Unable to connect to JACK server\n");
      }
      return 1;
    }
    if (status & JackServerStarted) {
      fprintf(stderr, "JACK server started\n");
    }
    if (status & JackNameNotUnique) {
      client_name = jack_get_client_name(client);
      fprintf(stderr, "unique name `%s' assigned\n", client_name);
    }

    /* initialize */
    state state;
    init(&state);
    jack_set_process_callback(client, process, &state);
    jack_on_shutdown(client, jack_shutdown, 0);

    // printf ("engine sample rate: %d\n", (int) jack_get_sample_rate (client));

    /* create ports */
    input_port [0] = jack_port_register(client, "input_1", ↙
        ↳ JACK_DEFAULT_AUDIO_TYPE, JackPortIsInput, 0);
    input_port [1] = jack_port_register(client, "input_2", ↙
        ↳ JACK_DEFAULT_AUDIO_TYPE, JackPortIsInput, 0);
    output_port[0] = jack_port_register(client, "output_1", ↙
        ↳ JACK_DEFAULT_AUDIO_TYPE, JackPortIsOutput, 0);
    output_port[1] = jack_port_register(client, "output_2", ↙
        ↳ JACK_DEFAULT_AUDIO_TYPE, JackPortIsOutput, 0);

    /* check ports */
    if (!input_port[0] || !input_port[1] || !output_port[0] || !output_port[1]) {
      fprintf(stderr, "no more JACK ports available\n");
      return 1;
    }

    /* activate */
```

```
  if (jack_activate(client)) {
    fprintf(stderr, "cannot activate client");
    return 1;
  }

  /* connect ports */
  jack_connect(client, "system:capture_1", jack_port_name(input_port[0]));
  jack_connect(client, "system:capture_1", jack_port_name(input_port[1]));
  jack_connect(client, jack_port_name(output_port[0]), "system:playback_1");
  jack_connect(client, jack_port_name(output_port[1]), "system:playback_2");

  /* process callbacks */
  while (1) {
    sleep(1);
  }

  /* finish */
  jack_client_close(client);
  return 0;
}
```

9 puredata.h

```
#include <stdlib.h>

#include "sample.h"

#define PI sample(3.141592653589793)
#define SR sample(48000.0)

#define SIGf_state(S,R) sample S;
#define SIGf_init(S,R) S = R;
#define SIGf_dsp(S,R,O) O = S;

#define SIG_state(S) SIGf_state(S,0)
#define SIG_init(S) SIGf_init(S,0)
#define SIG_dsp(S,O) SIGf_dsp(S,0,O)

#define ADDf_state(S,R) sample S;
#define ADDf_init(S,R) S = R;
#define ADDf_dsp(S,R,O,L) O = L + S;

#define SUBf_state(S,R) sample S;
#define SUBf_init(S,R) S = R;
#define SUBf_dsp(S,R,O,L) O = L - S;

#define MULf_state(S,R) sample S;
#define MULf_init(S,R) S = R;
#define MULf_dsp(S,R,O,L) O = L * S;

#define DIVf_state(S,R) sample S;
#define DIVf_init(S,R) S = R;
#define DIVf_dsp(S,R,O,L) O = L / S;

#define ADD_dsp(S,O,L,R) O = L + R;
#define SUB_dsp(S,O,L,R) O = L - R;
#define MUL_dsp(S,O,L,R) O = L * R;
```

```
35  #define DIV_dsp(S,O,L,R) O = L / R;

    #define DBTORMS_dsp(S,O,I) O = dbtorms(I);
    #define RMSTODB_dsp(S,O,I) O = rmstodb(I);
    #define SQRT_dsp(S,O,I) O = sqrt(I);
40  #define TANH_dsp(S,O,I) O = tanh(I);
    #define COS_dsp(S,O,I) O = cos(I * 2 * PI);
    #define WRAP_dsp(S,O,I) O = wrap(I);
    #define ABS_dsp(S,O,I) O = abs(I);
    #define MTOF_dsp(S,O,I) O = mtof(I);
45
    #define HIPf_state(S,HZ) struct { sample last, coef; } S;
    #define HIPf_init(S,HZ) { S.last = 0; S.coef = clip(1 - HZ * 2 * PI / SR, 0.0, ↙
        ↳ 1.0); }
    #define HIPf_dsp(S,HZ,O,I) { sample next = I + S.coef * S.last; O = next - S.↙
        ↳ last; S.last = next; }

50  #define HIP_state(S) HIPf_state(S,0)
    #define HIP_init(S) HIPf_init(S,0)
    #define HIP_dsp(S,O,I) HIPf_dsp(S,0,O,I)

    #define LOPf_state(S,HZ) struct { sample last, coef, feedback; } S;
55  #define LOPf_init(S,HZ) { S.last = 0; S.coef = clip(HZ * 2 * PI / SR, 0.0, 1.0);↙
        ↳  S.feedback = 1 - S.coef; }
    #define LOPf_dsp(S,HZ,O,I) { O = (S.last = S.coef * I + S.feedback * S.last); }

    #define LOP_state(S) LOPf_state(S,0)
    #define LOP_init(S) LOPf_init(S,0)
60  #define LOP_dsp(S,O,I) LOPf_dsp(S,0,O,I)

    #define PHASORf_state(S,HZ) struct { sample last, incr; } S;
    #define PHASORf_init(S,HZ) { S.last = 0; S.incr = HZ / SR; }
    #define PHASORf_dsp(S,HZ,O) { O = (S.last = wrap(S.last + S.incr)); }
65
    #define PHASOR_state(S) struct { sample last; } S;
    #define PHASOR_init(S) { S.last = 0; }
    #define PHASOR_dsp(S,O,I) { O = (S.last = wrap(S.last + I / SR)); }

70  #define OSCf_state(S,HZ) struct { sample last, incr; } S;
    #define OSCf_init(S,HZ) { S.last = 0; S.incr = HZ / SR; }
    #define OSCf_dsp(S,HZ,O) { S.last = wrap(S.last + S.incr); O = cos(2 * PI * S.↙
        ↳ last); }

    #define OSC_state(S) struct { sample last; } S;
75  #define OSC_init(S) { S.last = 0; }
    #define OSC_dsp(S,O,I) { S.last = wrap(S.last + I / SR); O = cos(2 * PI * S.last↙
        ↳ ); }

    static int NOISE_seed = 307;
    #define NOISE_state(S) int S;
80  #define NOISE_init(S) S = (NOISE_seed *= 1319);
    #define NOISE_dsp(S,O) { O = sample((S & 0x7fffffff) - 0x40000000) * sample(1.0 ↙
        ↳ / 0x40000000); S = S * 435898247 + 382842987; }

    #define SAMPHOLD_state(S) struct { sample hold, ctrl; } S;
    #define SAMPHOLD_init(S) { S.hold = 0; S.ctrl = 0; }
85  #define SAMPHOLD_dsp(S,O,L,R) { if (R < S.ctrl) { S.hold = L; } S.ctrl = R; O = ↙
```

```
    ↳ S.hold; }

#define TABLE_state(S,NAME,SIZE) struct { int length; sample *buffer; } table_##↙
    ↳ NAME;
#define TABLE_init(S,NAME,SIZE) { state->table_##NAME.length = SIZE; state->↙
    ↳ table_##NAME.buffer = (sample *) calloc(state->table_##NAME.length, sizeof↙
    ↳ (sample)); }

#define TABOSC4f_state(S,NAME,HZ) struct { sample last, incr; } S;
#define TABOSC4f_init(S,NAME,HZ) { S.last = 0; S.incr = HZ / SR; }
#define TABOSC4f_dsp(S,NAME,HZ,O) { \
  int L = state->table_##NAME.length; \
  sample *B = state->table_##NAME.buffer; \
  sample rindex = S.last * (L - 3) + 1; \
  sample f = wrap(rindex); \
  int ri = clip(int(floor(rindex), 1, L - 3)); \
  int i0 = ri - 1; \
  int i1 = ri    ; \
  int i2 = ri + 1; \
  int i3 = ri + 2; \
  sample s0 = B[i0]; \
  sample s1 = B[i1]; \
  sample s2 = B[i2]; \
  sample s3 = B[i3]; \
  O = interpolate4(f, s0, s1, s2, s3); \
  S.last = wrap(S.last + S.incr); \
}

#define TABOSC4_state(S,NAME) struct { sample last; } S;
#define TABOSC4_init(S,NAME) { S.last = 0; }
#define TABOSC4_dsp(S,NAME,O,I) { \
  int L = state->table_##NAME.length; \
  sample *B = state->table_##NAME.buffer; \
  sample rindex = S.last * (L - 3) + 1; \
  sample f = wrap(rindex); \
  int ri = clip(int(floor(rindex), 1, L - 3)); \
  int i0 = ri - 1; \
  int i1 = ri    ; \
  int i2 = ri + 1; \
  int i3 = ri + 2; \
  sample s0 = B[i0]; \
  sample s1 = B[i1]; \
  sample s2 = B[i2]; \
  sample s3 = B[i3]; \
  O = interpolate4(f, s0, s1, s2, s3); \
  S.last = wrap(S.last + I / SR); \
}

#define TABWRITE_state(S,NAME) struct { int windex, writing; } S;
#define TABWRITE_init(S,NAME) { S.windex = 0; S.writing = 0; }
#define TABWRITE_dsp(S,NAME,I) { \
  if (S.writing) { \
    if (S.windex < state->table_##NAME.length) { \
      state->table_##NAME.buffer[S.windex++] = I; \
    } else { \
      S.writing = 0; \
    } \
```

```
  } \
}
#define TABWRITE_0_bang(S,NAME) { S.windex = 0; S.writing = 1; }

#define DELWRITE_state(S,NAME,MS) struct { int length, windex; sample *buffer; } delwrite_##NAME;
#define DELWRITE_init(S,NAME,MS) { state->delwrite_##NAME.length = MS * SR / 1000; state->delwrite_##NAME.windex = 0; state->delwrite_##NAME.buffer = (sample *) calloc(state->delwrite_##NAME.length, sizeof(sample)); }
#define DELWRITE_dsp(S,NAME,MS,I) { state->delwrite_##NAME.buffer[state->delwrite_##NAME.windex] = I; state->delwrite_##NAME.windex += 1; state->delwrite_##NAME.windex %= state->delwrite_##NAME.length; }

#define DELREAD_state(S,NAME,MS) sample S;
#define DELREAD_init(S,NAME,MS) { S = MS; }
#define DELREAD_dsp(S,NAME,MS,O) { \
  int L = state->delwrite_##NAME.length; \
  int W = state->delwrite_##NAME.windex + L; \
  sample *B = state->delwrite_##NAME.buffer; \
  sample rindex = S * SR / 1000; \
  sample f = wrap(rindex); \
  int ri = clip(int(floor(rindex)), 1, L - 4); \
  int i0 = (W - (ri - 1)) % L; \
  int i1 = (W - (ri    )) % L; \
  int i2 = (W - (ri + 1)) % L; \
  int i3 = (W - (ri + 2)) % L; \
  sample s0 = B[i0]; \
  sample s1 = B[i1]; \
  sample s2 = B[i2]; \
  sample s3 = B[i3]; \
  O = interpolate4(f, s0, s1, s2, s3); \
}

#define VD_state(S,NAME)
#define VD_init(S,NAME)
#define VD_dsp(S,NAME,O,I) { \
  int L = state->delwrite_##NAME.length; \
  int W = state->delwrite_##NAME.windex + L; \
  sample *B = state->delwrite_##NAME.buffer; \
  sample rindex = I * SR / 1000; \
  sample f = wrap(rindex); \
  int ri = clip(int(floor(rindex).x), 1, L - 4); \
  int i0 = (W - (ri - 1)) % L; \
  int i1 = (W - (ri    )) % L; \
  int i2 = (W - (ri + 1)) % L; \
  int i3 = (W - (ri + 2)) % L; \
  sample s0 = B[i0]; \
  sample s1 = B[i1]; \
  sample s2 = B[i2]; \
  sample s3 = B[i3]; \
  O = interpolate4(f, s0, s1, s2, s3); \
}

#define SEND_state(S,NAME) sample send_##NAME;
#define SEND_init(S,NAME) { state->send_##NAME = 0; }
#define SEND_dsp(S,NAME,I) { state->send_##NAME = I; }
```

```
#define RECEIVE_state(S,NAME)
#define RECEIVE_init(S,NAME)
#define RECEIVE_dsp(S,NAME,O) { O = state->send_##NAME; }

#define ADC_state(S) sample *S[2];
#define ADC_init(S) { S[0] = &(state->adc[0]); S[1] = &(state->adc[1]); }
#define ADC_dsp(S,L,R) { L = *(S[0]); R = *(S[1]); }

#define DAC_state(S) sample *S[2];
#define DAC_init(S) { S[0] = &(state->dac[0]); S[1] = &(state->dac[1]); }
#define DAC_dsp(S,L,R) { *(S[0]) += L; *(S[1]) += R; }
```

10 puredata.tilde

```
sig~ f SIGf f ~
sig~ _ SIG f ~
+~ f ADDf ~f ~
+~ _ ADD ~~ ~
-~ f SUBf ~f ~
-~ _ SUB ~~ ~
*~ f MULf ~f ~
*~ _ MUL ~~ ~
/~ f DIVf ~f ~
/~ _ DIV ~~ ~
hip~ f HIPf ~f ~
hip~ _ HIP ~f ~
lop~ f LOPf ~f ~
lop~ _ LOP ~f ~
dbtorms~ _ DBTORMS ~ ~
rmstodb~ _ RMSTODB ~ ~
sqrt~ _ SQRT ~ ~
tanh~ _ TANH ~ ~
cos~ _ COS ~ ~
wrap~ _ WRAP ~ ~
abs~ _ ABS ~ ~
mtof~ _ MTOF ~ ~
adc~ _ ADC _ ~~
dac~ _ DAC ~~ _
phasor~ f PHASORf ff ~
phasor~ _ PHASOR ~f ~
osc~ f OSCf ff ~
osc~ _ OSC ~f ~
samphold~ _ SAMPHOLD ~~ ~
delwrite~ sf DELWRITE ~ _
delread~ sf DELREAD f ~
vd~ s VD ~ ~
send~ s SEND ~ _
s~ s SEND ~ _
receive~ s RECEIVE _ ~
r~ s RECEIVE _ ~
noise~ _ NOISE _ ~
table sf TABLE _ _
tabosc4~ sf TABOSC4f f ~
tabosc4~ s TABOSC4 ~ ~
tabwrite~ s TABWRITE ~ _
expr~ * EXPR ~~~~~~~~~~ ~~~~~~~~~~
inlet~ * INLET _ ~
```

```
outlet~ * OUTLET~ _
```

11 README

```
tilde -- a compiler from Pure-data patches via C++ to JACK applications
Copyright (C) 2011 Claude Heiland-Allen <claude@mathr.co.uk>

legal:
  the compiler is GPL, and the generated code currently makes use of GPL
  portions too, as well as code snippets from Pure-data itself.  whether
  that means you must offer to distribute your patch when you distribute
  your binary compiled from it, well, i'm no lawyer but it would be nice
  of you to share :)

usage:
  $ runghc tilde.hs "${PATCH}~.pd" |
    g++ -xc++ - -ljack -o "${PATCH}" -O3 -march=native &&
    "./${PATCH}"   # hook up some sounds

example patches:
  compress~.pd -- dynamic range compression
  pitchshift~.pd -- pitch shifting with delay lines

roadmap:
  port more signal ugens
  pd backend as well as jack standalone
  (eventually) message system
  ((later)) gui controls
  (((never))) dynamic patching, dlopen, etc...

todo:
  ugens:
    writesf~
    vline~
  msg:
    message
    bang~
    snapshot~
    samplerate~
    pow
    receive r
    send s
    trigger t
    change
    delay
    float f
    +
    -
    *
    /
    line
    loadbang
    metro
    mod
    pack
    select
    swap
```

```
    table
  subpatch:
    inlet
    outlet
  abstractions:
    inlet~
    outlet~
    inlet
    outlet
  gui:
    bng
    hsl
    vsl
    floatatom
    savepanel
```

12 README.examples

```
proof of concept of 'tilde': a compiler from Pure-data to C++ JACK standalone
the .cc files are generated from .pd sources, see README for more about them
see http://code.mathr.co.uk/tilde for compiler sources and more info
--
Claude Heiland-Allen 2011
http://mathr.co.uk
claude@mathr.co.uk
```

13 sample.h

```
#include <algorithm>
#include <cmath>

using namespace std;

// typedef double sample;
struct sample {
  double x;

  inline sample() { }

  inline sample(sample const &y) : x(y.x) { if (!isfinite(x)) { x = 0; } }
  inline sample(double const &y) : x(y) { if (!isfinite(x)) { x = 0; } }
  inline sample(float const &y) : x(y) { if (!isfinite(x)) { x = 0; } }
  inline sample(int const &y) : x(y) { }

  inline sample& operator=(sample const &y) { x = y.x; return *this; }

  inline sample& operator+=(sample const &y) { x += y.x; return *this; }
  inline sample& operator-=(sample const &y) { x -= y.x; return *this; }
  inline sample& operator*=(sample const &y) { x *= y.x; return *this; }
  inline sample& operator/=(sample const &y) { if (y.x != 0) { x /= y.x; } else
    ↳ { x = 0; } return *this; }

};

inline sample operator+(sample const &l, sample const &r) { return l.x + r.x; }
inline sample operator-(sample const &l, sample const &r) { return l.x - r.x; }
inline sample operator*(sample const &l, sample const &r) { return l.x * r.x; }
```

```
inline sample operator/(sample const &l, sample const &r) { return l.x / r.x; }

inline sample operator-(sample const &l) { return -l.x; }

inline sample operator%(sample const &l, sample const &r) { int y = r.x; if (y ↙
    ↳ != 0) { return int(l.x) % y; } else { return 0; } }

inline sample operator!(sample const &l) { return !l.x; }
inline sample operator&&(sample const &l, sample const &r) { return l.x && r.x; ↙
    ↳ }
inline sample operator||(sample const &l, sample const &r) { return l.x || r.x; ↙
    ↳ }

inline bool operator< (sample const &l, sample const &r) { return l.x <  r.x; }
inline bool operator<=(sample const &l, sample const &r) { return l.x <= r.x; }
inline bool operator!=(sample const &l, sample const &r) { return l.x != r.x; }
inline bool operator==(sample const &l, sample const &r) { return l.x == r.x; }
inline bool operator> (sample const &l, sample const &r) { return l.x >  r.x; }
inline bool operator>=(sample const &l, sample const &r) { return l.x >= r.x; }

inline sample abs(sample const &l) { return abs(l.x); }
inline sample floor(sample const &l) { return floor(l.x); }
inline sample fmod(sample const &l, sample const &r) { return fmod(l.x, r.x); }

inline sample exp(sample const &l) { return exp(l.x); }
inline sample pow10(sample const &l) { return pow10(l.x); }
inline sample pow(sample const &l, sample const &r) { return pow(l.x, r.x); }

inline sample log(sample const &l) { return log(l.x); }
inline sample log10(sample const &l) { return log10(l.x); }

inline sample sin(sample const &l) { return sin(l.x); }
inline sample cos(sample const &l) { return cos(l.x); }


static inline sample clip(sample x, sample lo, sample hi) {
  return min(max(x, lo), hi);
}

static inline int clip(int x, int lo, int hi) {
  return min(max(x, lo), hi);
}

static inline sample wrap(sample x) {
  return x - floor(x);
}

static inline sample rmstodb(sample x) {
  if (x <= 0) {
    return 0;
  } else {
    return max(sample(0), 100 + 20 * log10(x));
  }
}

static inline sample dbtorms(sample x) {
  if (x <= 0) {
```

```
    return 0;
  } else {
    return pow10((min(x, sample(485)) - 100) / 20);
  }
}

static inline sample mtof(sample x) {
  if (x <= -1500) {
    return 0;
  } else {
    return 8.17579891564 * exp(0.0577622650 * min(x, sample(1499)));
  }
}

static inline sample interpolate4(sample const &x, sample const &a, sample const &
    ↪ &b, sample const &c, sample const &d) {
  sample a1 = 0.5 * (c - a);
  sample a2 = a - 2.5 * b + 2 * c - 0.5 * d;
  sample a3 = 0.5 * (d - a) + 1.5 * (b - c);
        return ((a3 * x + a2) * x + a1) * x + b;
}

static inline sample eif(sample const &b, sample const &t, sample const &f) {
  return b != 0 ? t : f;
}

static inline sample efloor(sample const &x, int y) {
  return floor(x);
}

static inline sample erandom(sample const &l, sample const &r) {
  int i1 = l.x;
  int i2 = r.x;
  return i1 + (((i2 - i1) * (rand() & 0x7fff)) >> 15);
}
```

14 tanh~.pd

```
#N canvas 0 0 450 300 10;
#X obj 23 16 inlet~;
#X obj 23 37 expr~ tanh($v1);
#X obj 23 58 outlet~;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
```

15 tilde.hs

```
{-# LANGUAGE GeneralizedNewtypeDeriving #-}
module Main (main) where

import Data.List(foldl', intercalate, sort, genericLength)
import Safe (atMay)
import System.Environment(getArgs)
import Data.Maybe (isJust)
import Data.Monoid (mconcat)
import qualified Data.Map as M
import qualified Data.Set as S
```

```
import qualified Data.Graph.Inductive as G
import Data.Graph.Inductive.PatriciaTree (Gr)

import Debug.Trace (trace)

type Node = G.LNode Object
type Edge = G.LEdge Wire
type Graph = (String, Gr Object Wire, ([ObjectID], [ObjectID]))

data Object = Object String | Graph Graph
type Wire = (OutletID, InletID)

type Path = [ObjectID]

type WireName = String

type Code = (String, String, String)

state :: Path -> String
state = intercalate "_" . ("state":) . map (show . unObj)

state' :: Path -> String
state' = ("state->" ++) . state

inlet :: Path -> InletID -> WireName
inlet p i = intercalate "_" . ("wire":) . (++["i", show . unIn $ i]) . map (show . unObj) $ p

outlet :: Path -> OutletID -> WireName
outlet p o = intercalate "_" . ("wire":) . (++["o", show . unOut $ o]) . map (show . unObj) $ p

graph :: PdPatch -> Graph
graph pd = if not . S.null $ S.unions[S.fromList[f,t]|(f,t,_)<-es] `S.difference` S.fromList (map fst ns) then error . show $ (map fst ns, es) else
  (name pd, G.mkGraph ns es, (map snd . sort . inlets $ pd, map snd . sort . outlets $ pd))
  where
    ns = zipWith node [0..] . objects $ pd
    es = map edge . connects $ pd


node :: G.Node -> PdObject -> Node
node n (PdObject s) = (n, Object s)
node n (PdSubpatch pd) = (n, Graph . graph $ pd)

edge :: Connect -> Edge
edge (fobj, fout, tobj, tin) = (fromIntegral fobj, fromIntegral tobj, (fout, tin))

dspGraph :: KnownObjects -> Gr Object Wire -> Gr Object Wire
dspGraph objs g = G.mkGraph (filter (dspNode objs g) (G.labNodes g)) (filter (dspEdge objs g) (G.labEdges g))

dspNode :: KnownObjects -> Gr Object Wire -> Node -> Bool
dspNode objs g (n, Object s) = if any (dspOutlet objs g)
  (  [ (m, o) | (m, (o, _)) <- G.lpre g n ]
```

```
  ++ [ (n, o) | (_, (o, _)) <- G.lsuc g n ] ) then True else trace ("warning: ↙
      ↳ discarded: " ++ s) False
dspNode _ _ (_, Graph _) = True

dspEdge :: KnownObjects -> Gr Object Wire -> Edge -> Bool
dspEdge objs g (m, n, (o, i))
  | dspOutlet objs g (m, o) && dspInlet objs g (n, i) = True
  | dspOutlet objs g (m, o) = trace "warning: signal -> non-signal" False
  | dspInlet objs g (n, i) = trace "warning: non-signal -> signal" False
  | otherwise = False

dspOutlet objs g (n, o) = case G.lab g (fromIntegral n) of
  Just (Object s) -> case findObject objs s of
    Just (_, _, os) -> case os `atMay` fromIntegral o of
      Just SignalIO -> True
      Just _ -> False
    Nothing -> False
  Just (Graph (_, g', (_, os))) -> case os `atMay` fromIntegral o of
    Just m -> case G.lab g' (fromIntegral m) of
      Just (Object s) -> case words s of
        ("outlet~":_) -> True
        ("outlet":_) -> False

dspInlet objs g (n, i) = case G.lab g (fromIntegral n) of
  Just (Object s) -> case findObject objs s of
    Just (_, is, _) -> case is `atMay` fromIntegral i of
      Just SignalIO -> True
      Just _ -> False
    Nothing -> False
  Just (Graph (_, g', (is, _))) -> case is `atMay` fromIntegral i of
    Just m -> case G.lab g' (fromIntegral m) of
      Just (Object s) -> case words s of
        ("inlet~":_) -> True
        ("inlet":_) -> False

isDSPInlet g n = case G.lab g n of
  Just (Object s) -> case words s of
    ("inlet~":_) -> True
    ("inlet":_) -> False

isDSPOutlet g n = case G.lab g n of
  Just (Object s) -> case words s of
    ("outlet~":_) -> True
    ("outlet":_) -> False

compileDSP :: KnownObjects -> Path -> Graph -> Code
compileDSP objs path (sn, g, (_, _)) = mconcat . map mconcat $
  [ [ ("","","const sample " ++ inlet path' n_in ++ " = " ++ (intercalate " + " ↙
      ↳ . ("0":)) [ outlet (path ++ [fromIntegral o]) o_out | (o, o_out) <- from↙
      ↳ ] ++ ";\n")
    | (n_in, from) <- M.toAscList froms
    ] ++
    [ ("","","sample " ++ outlet path' n_out ++ ";\n") | n_out <- outs ] ++
    ( case n_name of
      Object s ->
        [ ("/* [" ++ s ++ "] */\n", "/* [" ++ s ++ "] */\n", "/* [" ++ s ++ "] ↙
            ↳ */\n")
```

```
115           , compileDSPObject objs path' s
                (M.fromList[(n_in, inlet path' n_in) | n_in <- M.keys froms])
                (M.fromList[(n_out, outlet path' n_out) | n_out <- outs])
              ]
        Graph (sn', g', (is', os')) ->
120           [ ("", "", "{ /* subpatch " ++ sn' ++ " */\n" ++ concat [ "const sample ↙
                  ↳ inlet_" ++ show (unObj k) ++ " = " ++ inlet path' n_in ++ ";\n" | ↙
                  ↳ (k, n_in) <- is'`zip`[0..], isDSPInlet g' (fromIntegral k) ])
              , compileDSP objs path' (sn', g', (is', os'))
              , ("", "", concat [ outlet path' n_out ++ " = outlet_" ++ show (unObj k)↙
                  ↳  ++ ";\n" | (k, n_out) <- os'`zip`[0..], isDSPOutlet g' (↙
                  ↳ fromIntegral k) ] ++ "}\n")
              ]
      )
125   | let gd = dspGraph objs g
      , n <- G.topsort gd
      , let path' = path ++ [fromIntegral n]
      , let Just n_name = G.lab gd n
      , let froms = M.fromListWith (++) [ (n_in, [(o, o_out)]) | (o, (o_out, n_in)) ↙
        ↳ <- G.lpre gd n ]
130   , let outs = (S.toList . S.fromList) [ n_out | (_, (n_out, _)) <- G.lsuc gd n ↙
        ↳ ]
    ]

compileDSPObject :: KnownObjects -> Path -> String -> M.Map InletID String -> M.↙
    ↳ Map OutletID String -> Code
compileDSPObject objs path s is os
135   | obj == "inlet~" = ("", "", case 0 `M.lookup` os of -- FIXME
        Nothing -> "/* inlet not found in outlet list */\n"
        Just o -> o ++ " = inlet_" ++ show (unObj $ last path) ++ ";\n"
      )
--    | obj == "outlet"  = ("", "", "sample outlet_" ++ show (last path) ++ " = " ↙
    ↳ ++ maybe "0" id (0 `M.lookup` is) ++ ";\n") -- FIXME
140   | obj == "outlet~" = ("", "", "const sample outlet_" ++ show (unObj $ last ↙
        ↳ path) ++ " = " ++ maybe "0" id (0 `M.lookup` is) ++ ";\n")
      | obj == "expr~" =
        let outs = map (`M.lookup` os) [ 0 .. 9 ]
            es = map (filter (/= '$')) . unsemi . filter (/= '\\') . unwords $ ↙
                ↳ args
        in  ( ""
145           , ""
              , "{\n" ++
                concat [ "const sample v" ++ show (unIn (k + 1)) ++ " = " ++ maybe ↙
                    ↳ "0" id (k `M.lookup` is) ++ ";\n" | k <- [ 0 .. 9 ] ] ++
                "#define if(b,t,f) eif(b,t,f)\n" ++
                "#define floor(x,y) efloor(x,y)\n" ++
150             "#define random(x,y) erandom(x,y)\n" ++
                concat [ o ++ " = " ++ e ++ ";\n" | (mo, e) <- outs `zip` es, isJust↙
                    ↳  mo, let Just o = mo ] ++
                "#undef random\n" ++
                "#undef floor\n" ++
                "#undef if\n" ++
155             "}\n"
              )
      | otherwise = case findObject objs s of
        Nothing -> ("/* not supported */\n", "/* not supported */\n", "/* not ↙
            ↳ supported */\n")
```

```
      Just (o, ni, no) ->
        let ins  = map (maybe "0" id . (`M.lookup` is)) [ k | (k, t) <- [0..] `zip` ni, t == SignalIO ]
            outs = map (maybe "disconnected" id . (`M.lookup` os)) [ k | (k, t) <- [0..] `zip` no, t == SignalIO ]
        in  ( "#ifdef " ++ o ++ "_state\n" ++ o ++ "_state(" ++ intercalate "," (state path : args) ++ ")\n#endif\n"
            , "#ifdef " ++ o ++ "_init\n" ++ o ++ "_init(" ++ intercalate "," (state' path : args) ++ ")\n#endif\n"
            , o ++ "_dsp(" ++ intercalate "," ([state' path] ++ args ++ outs ++ ins) ++ ")\n"
            )
  where
    obj = words s !! 0
    args = drop 1 (words s)

inspect :: String -> (String, [ArgT])
inspect s = (words s !! 0, map checkType . drop 1 . words $ s)

findObject :: KnownObjects -> String -> Maybe (String, [IoletT], [IoletT])
findObject objs s = findObject' args0
  where
    (o, args0) = inspect s
    findObject' as = case (o, as) `M.lookup` objs of
      Nothing -> case (o, as ++ [GimmeArg]) `M.lookup` objs of
        Nothing -> case as of
          [] -> Nothing
          _ -> findObject' (init as)
        r -> r
      r -> r

unsemi :: String -> [String]
unsemi xs = let (ys, zs) = span (/= ';') xs
            in  ys : case zs of
              [] -> []
              (';':zs') -> unsemi zs'
              _ -> error "unsemi"

checkType :: String -> ArgT
checkType s = case (reads :: ReadS Float) s of
  [(_,"")] -> FloatArg
  _ -> SymbolArg

flatten :: Code -> String
flatten (s, i, d) = "#include \"puredata.h\"\ntypedef struct {\nsample adc[2];\nsample dac[2];\n" ++ s ++ "} state;\n" ++ "static inline void init(state *state) {\n" ++ i ++ "}\n" ++ "static inline void dsp(state *state) {\nsample disconnected;\n" ++ d ++ "}\n#include \"puredata.c\"\n"

data IoletT = SignalIO | FloatIO deriving (Eq, Ord, Show)
data ArgT = FloatArg | SymbolArg | GimmeArg deriving (Eq, Ord)

parseArgT :: String -> [ArgT]
parseArgT [] = []
parseArgT "_" = []
parseArgT "*" = [GimmeArg]
parseArgT ('f':a) = FloatArg : parseArgT a
```

```
parseArgT ('s':a) = SymbolArg : parseArgT a
parseArgT _ = error "parseArgT"

parseIoletT :: String -> [IoletT]
parseIoletT [] = []
parseIoletT "_" = []
parseIoletT ('~':a) = SignalIO : parseIoletT a
parseIoletT ('f':a) = FloatIO : parseIoletT a
parseIoletT _ = error "parseIoletT"

type KnownObjects = M.Map (String, [ArgT]) (String, [IoletT], [IoletT])

parseSPD :: String -> KnownObjects
parseSPD = M.fromList . map ((\[pd,a,c,i,o] -> ((pd, parseArgT a), (c, ↙
    ↳ parseIoletT i, parseIoletT o))) . words) . lines

main' :: String -> IO ()
main' patch = do
  objs <- parseSPD `fmap` readFile "puredata.tilde"
  putStrLn . flatten . compileDSP objs [] . graph . readPatch patch =<< readFile↙
      ↳ patch

newtype ObjectID = Obj{unObj:: Int} deriving (Eq, Ord, Enum, Num, Real, Integral↙
    ↳ , Show)
newtype InletID  = In {unIn :: Int} deriving (Eq, Ord, Enum, Num, Real, Integral↙
    ↳ , Show)
newtype OutletID = Out{unOut:: Int} deriving (Eq, Ord, Enum, Num, Real, Integral↙
    ↳ , Show)
type Connect  = (ObjectID, OutletID, ObjectID, InletID)
type InletX   = Int
type OutletX  = Int

data PdObject
  = PdObject   String
  | PdSubpatch PdPatch

data PdPatch = PdPatch {
  name     :: String,
  objects  :: [PdObject],
  connects :: [Connect],
  inlets   :: [(InletX, ObjectID)],
  outlets  :: [(OutletX, ObjectID)]
}

readPatch :: String -> String -> PdPatch
readPatch s = head . foldl' readLine [emptyPatch s] . tail . joinAgain . ↙
    ↳ lineSemi

lineSemi :: String -> [String]
lineSemi "" = []
lineSemi s =
  let (l, s') = break (';' ==) . filter ('\n' /=) $ s
  in l : case s' of
    []     -> []
    (_:s'') -> lineSemi s''

joinAgain :: [String] -> [String]
```

```
joinAgain [] = []
joinAgain [x] = [x]
joinAgain (x:y:zs)
  | x == []         = joinAgain (y:zs)
  | last x == '\\'  = joinAgain ((init x++";"++y):zs)
  | otherwise       = x : joinAgain (y:zs)

emptyPatch :: String -> PdPatch
emptyPatch s = PdPatch { name = s, objects = [], connects = [], inlets = [], ↙
    ↳ outlets = [] }

readLine :: [PdPatch] -> String -> [PdPatch]
readLine (p:ps) l =
  let atoms = words l
  in case atoms!!0 of
    "#N" -> case atoms!!1 of
      "canvas"    -> let p' = emptyPatch "pd" in
                     p':p:ps
      _ -> error "readLine #N"
    "#X" -> case atoms!!1 of
      "restore"   -> let (p':ps') = ps in
                     p' { objects = objects p' ++ [PdSubpatch $ p { name = ↙
                         ↳ unwords (drop 4 atoms) }] } : ps'
      "coords"    -> p : ps
      "obj"       -> case atoms !! 4 of
        "inlet"   -> p { objects = objects p ++ [PdObject $ unwords (drop 4 ↙
            ↳ atoms)], inlets = inlets p ++ [(read $ atoms !! 2, genericLength $↙
            ↳  objects p)] } : ps
        "inlet~"  -> p { objects = objects p ++ [PdObject $ unwords (drop 4 ↙
            ↳ atoms)], inlets = inlets p ++ [(read $ atoms !! 2, genericLength $↙
            ↳  objects p)] } : ps
        "outlet"  -> p { objects = objects p ++ [PdObject $ unwords (drop 4 ↙
            ↳ atoms)], outlets = outlets p ++ [(read $ atoms !! 2, genericLength↙
            ↳  $ objects p)] } : ps
        "outlet~" -> p { objects = objects p ++ [PdObject $ unwords (drop 4 ↙
            ↳ atoms)], outlets = outlets p ++ [(read $ atoms !! 2, genericLength↙
            ↳  $ objects p)] } : ps
        _         -> p { objects = objects p ++ [PdObject $ unwords (drop 4 ↙
            ↳ atoms)] } :  ps
      "msg"       -> p { objects = objects p ++ [PdObject $ unwords ("message" :↙
          ↳  drop 4 atoms)] } : ps
      "floatatom" -> p { objects = objects p ++ [PdObject $ unwords ("floatatom"↙
          ↳  : drop 4 atoms)] } : ps
      "text"      -> p { objects = objects p ++ [PdObject $ unwords ("comment" :↙
          ↳  drop 4 atoms)] } : ps
      "connect"   -> p { connects = connects p ++ [(Obj $ read (atoms!!2), Out $↙
          ↳  read (atoms!!3), Obj $ read (atoms!!4), In $ read (atoms!!5))] } : ↙
          ↳ ps
      _ -> error "readLine #X"
    _ -> error "readLine #N/#X"
readLine _ _ = error "readLine"

main :: IO ()
main = main' . head =<< getArgs
```