Emulating a combo organ using Faust

Sampo Savolainen

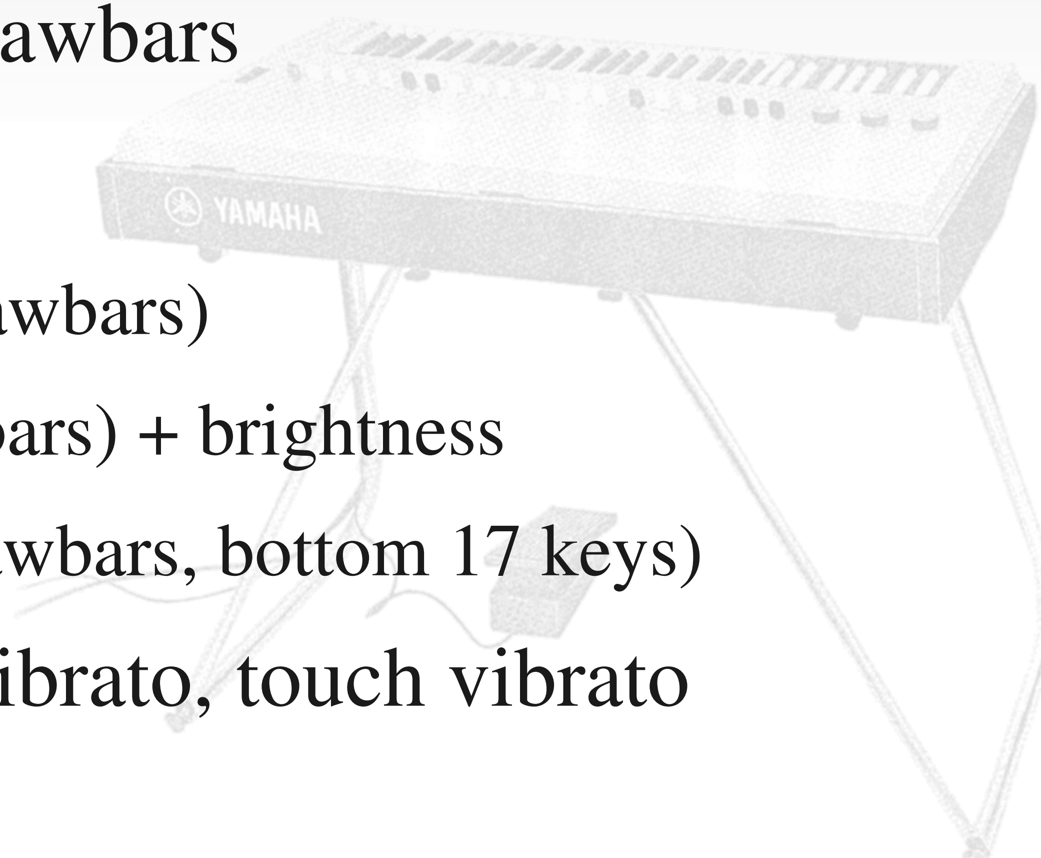
LAC 2010

Utrecht

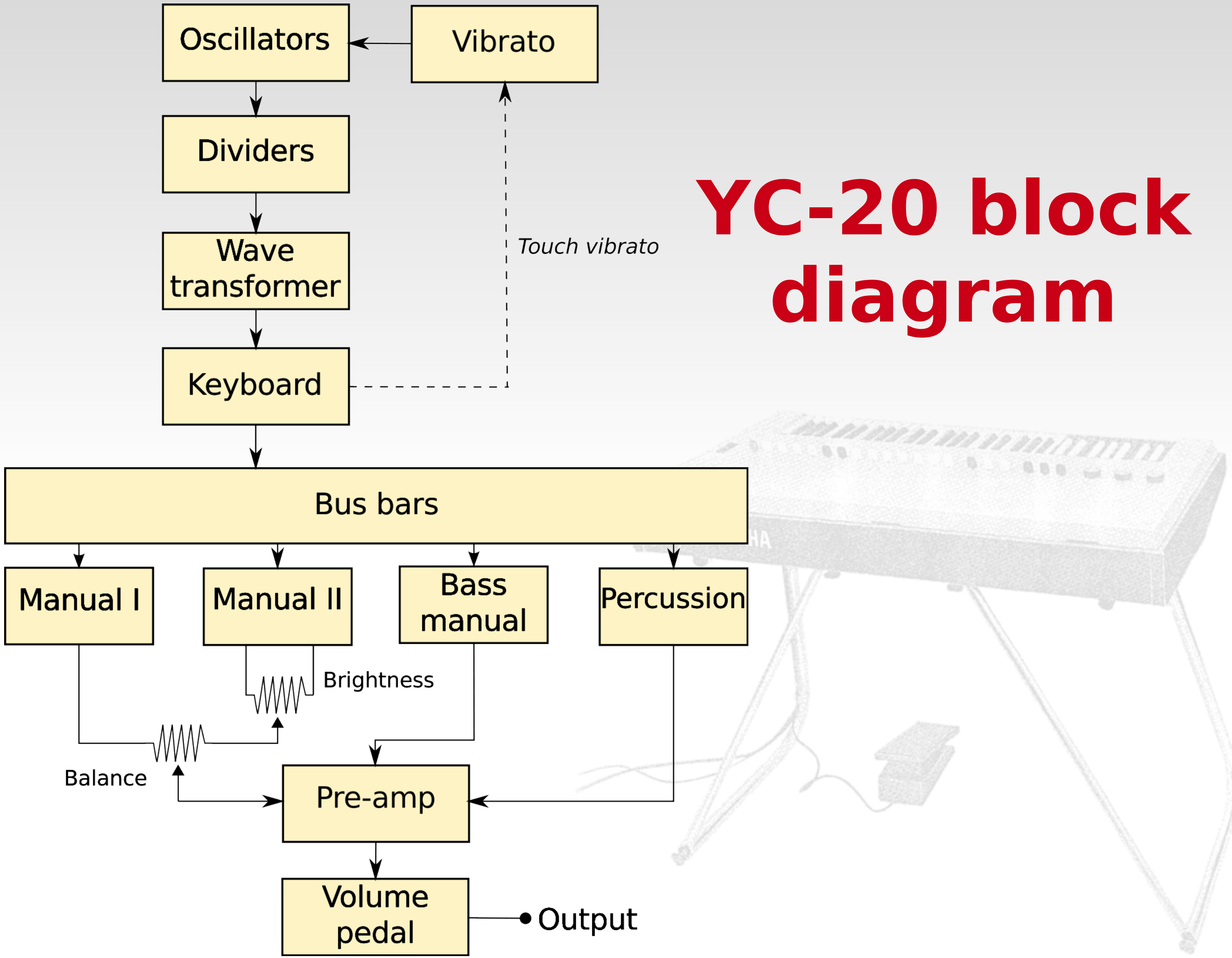

PITCH
VOLUME
BASS VOLUME
TOUCH VIB.
VIBRATO
VIB. SPEED
MAN. BASS
BASS
BALANCE
BRIGHT
PERCUSSIVE
YAMAHA

Yamaha YC-20

- Designed circa 1969
- One manual, 61 keys
- ”Levers” not actual drawbars
- 3 sections
  - Section I (5 + 2 drawbars)
  - Section II (4 drawbars) + brightness
  - Bass manual (2 drawbars, bottom 17 keys)
- Percussive drawbar, vibrato, touch vibrato



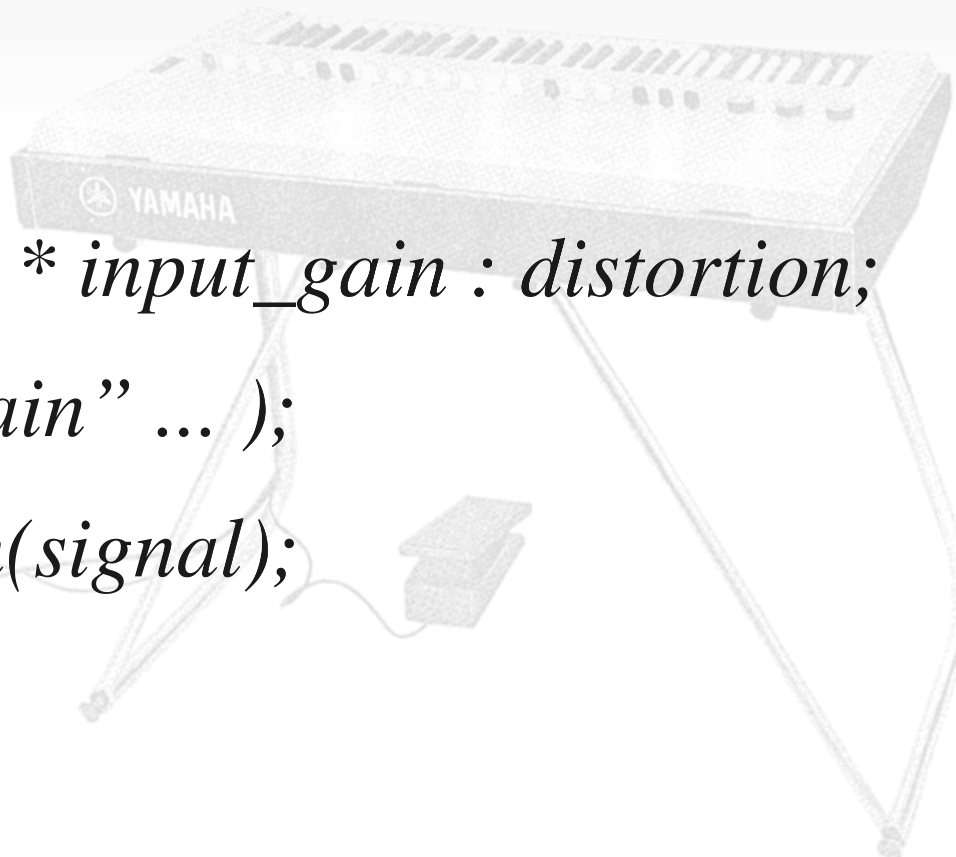
YC-20 block diagram



Faust

- Functional AUdio STream
- A functional programming language designed for signal processing

```
process(signal) = signal * input_gain : distortion;
input_gain = vslider(”gain” ... );
distortion(signal) = tanh(signal);
```

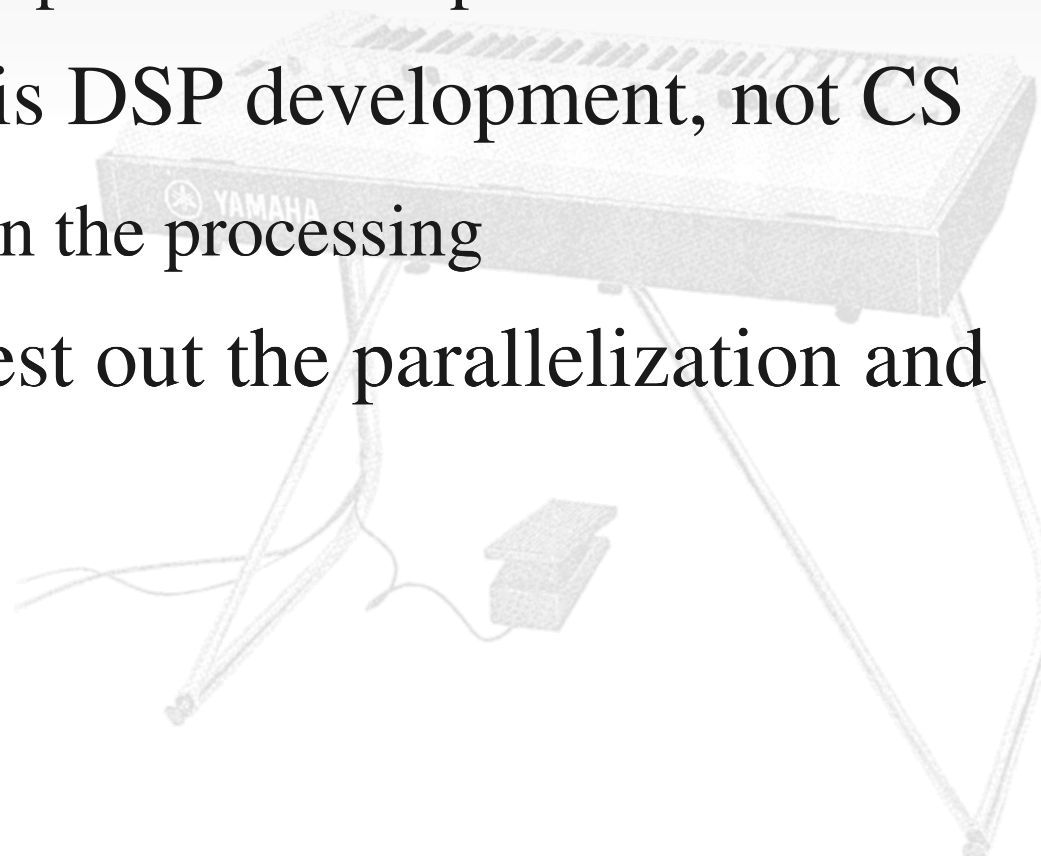


Faust

- The Faust compiler produces C++ code
  - Jack client, LADSPA / VST plugin, etc.
- Faust can optimize the generated code using different methods:
  - Automatic vectorization (see LAC 2009)
  - OpenMP parallelization (see LAC 2009)
  - Work stealing scheduler (tomorrow at 12:15..)

Why use Faust?

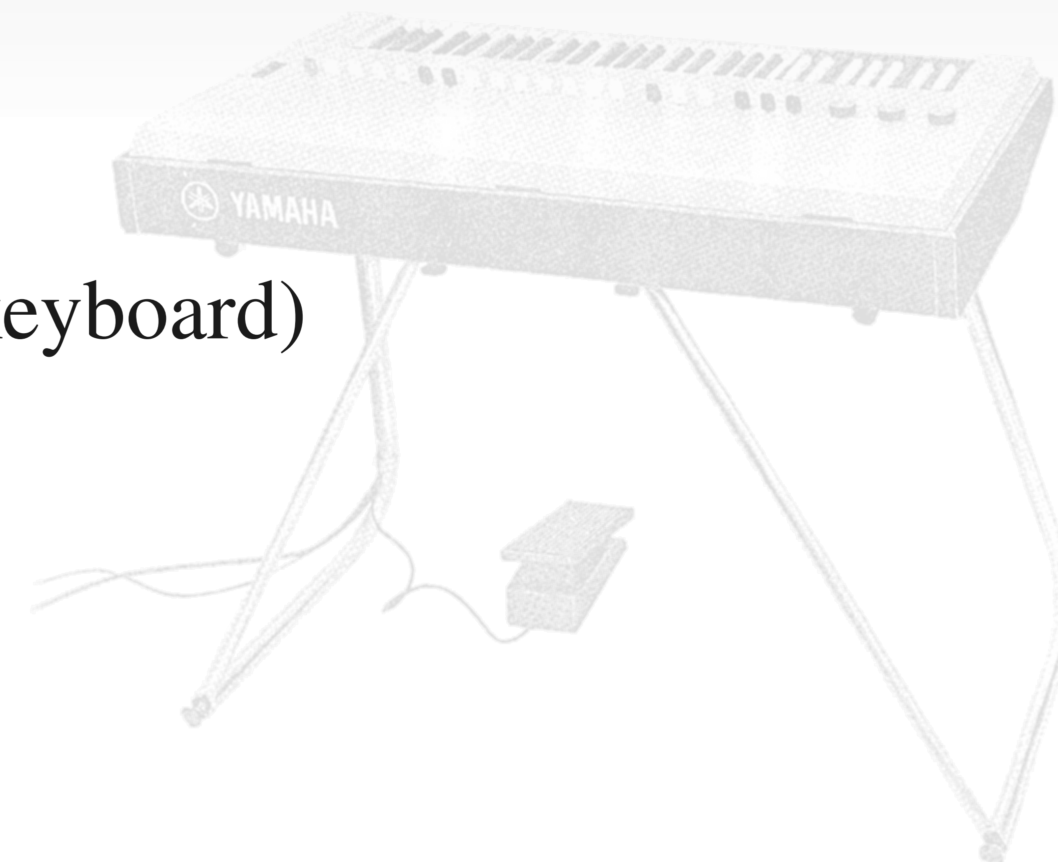
- The divide-down architecture is 'always-on'
    - Instead of routability and controllability, this requires a large number of parallel fixed processes
- An organ synthesizer is DSP development, not CS
    - Helps keep focus on the processing
- To try out Faust and test out the parallelization and performance

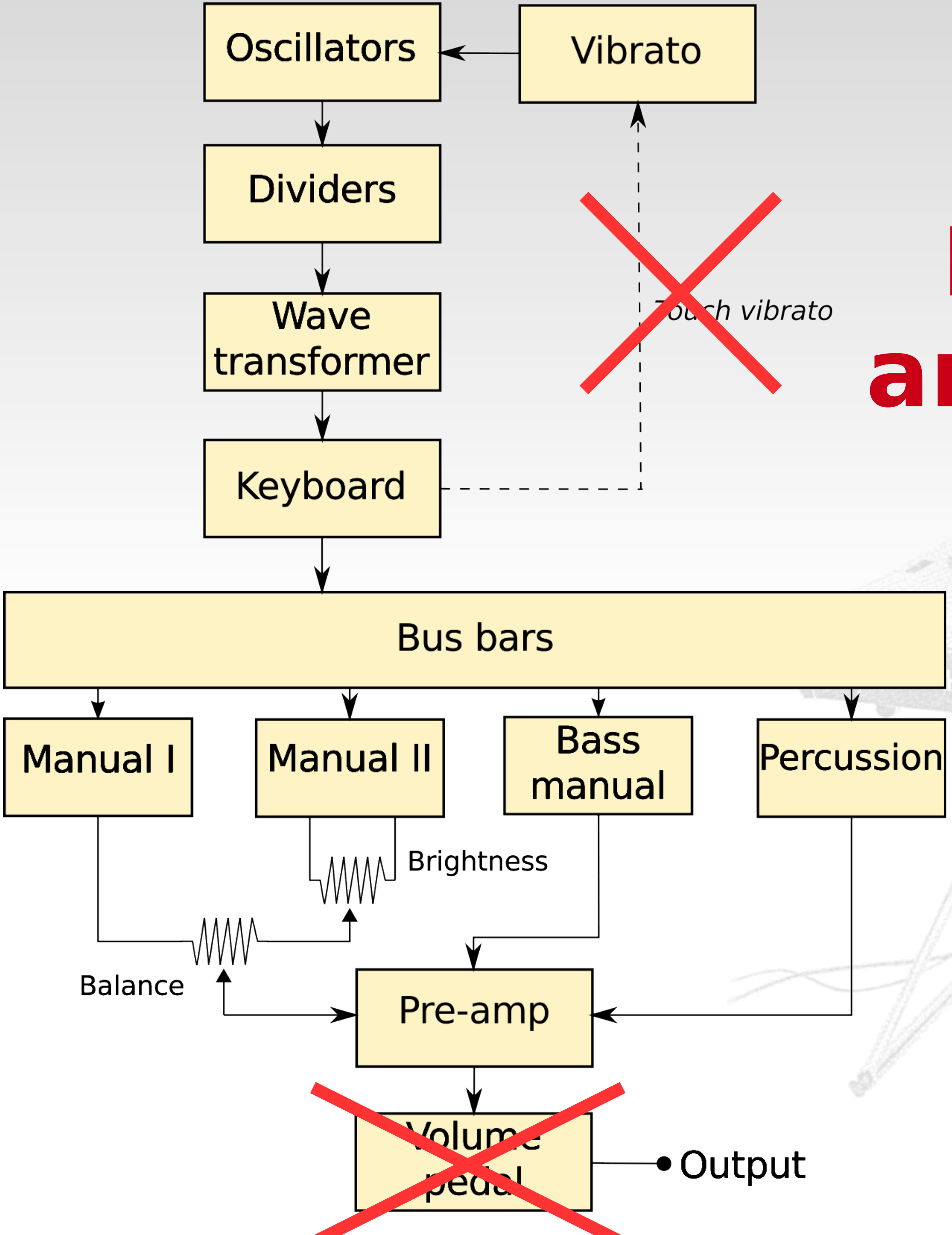


The organ emulation

Not a polysynth but a generator + matrix mixer!

- 96 oscillators
- 204 RC filters
- A matrix mixer (the keyboard)
- Percussion envelope
- Vibrato LFO
- Mixing section



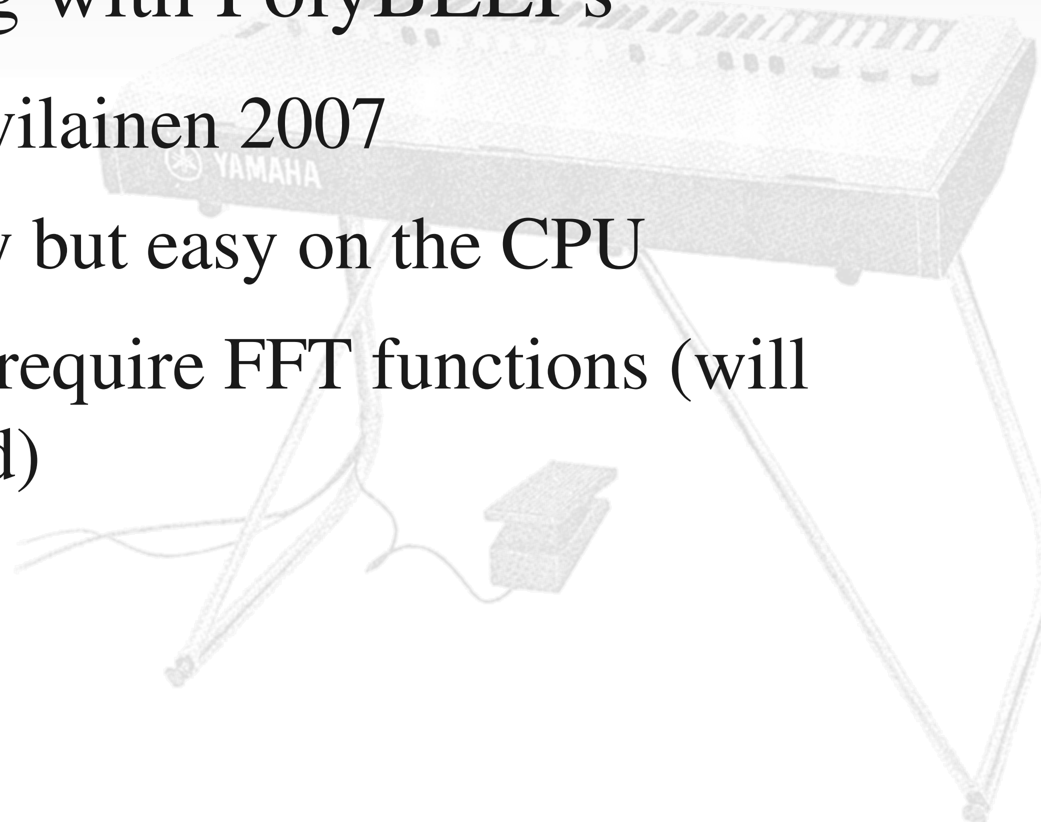


Emulated architecture

- No horizontal aftertouch in MIDI keyboards
- Pre-amp is just a mixer
- Volume pedal can be done externally

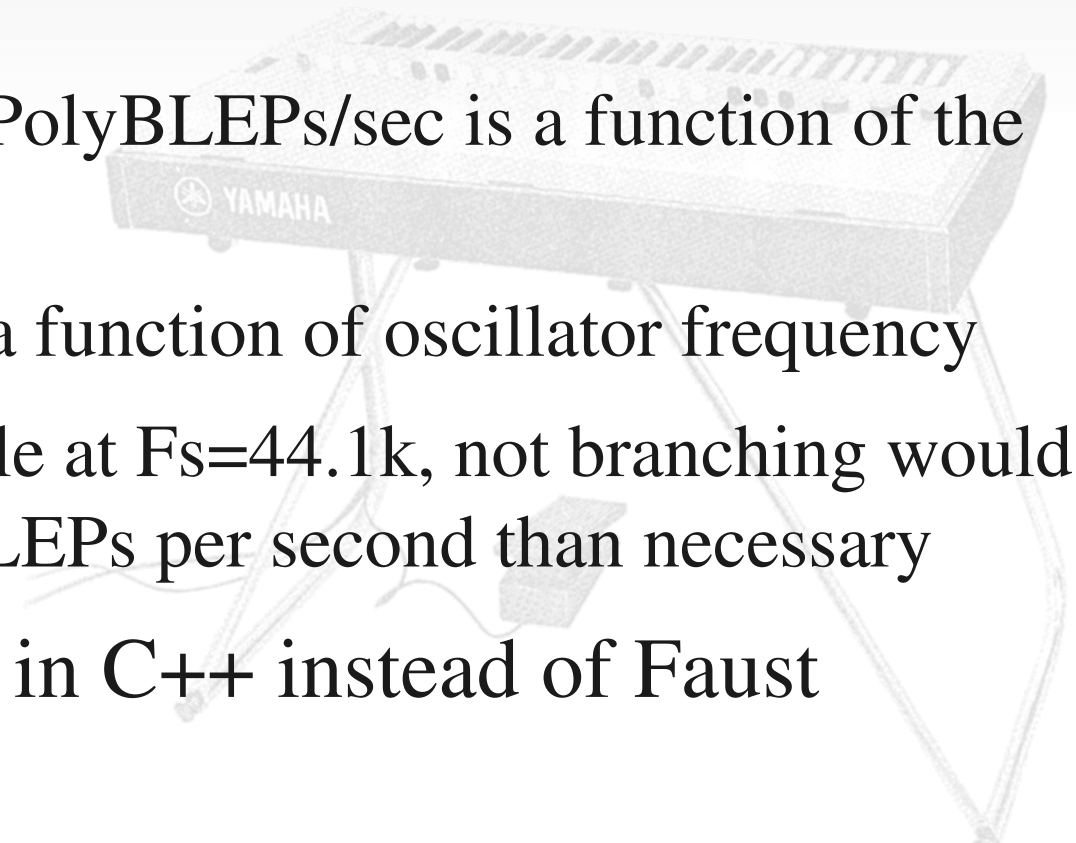
Oscillators

- Main oscillators produce sawtooth waves
- Dividers are flip-flops: they produce rectangles
- Oscillator anti-aliasing with PolyBLEPs
  - Välimäki and Huovilainen 2007
  - Not optimal quality but easy on the CPU
  - True BLEP would require FFT functions (will probably be fixed)

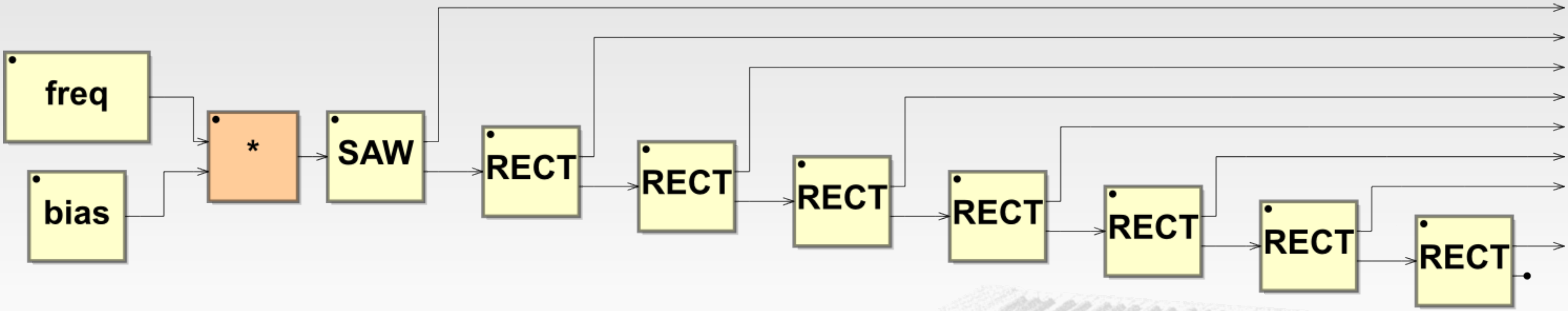


PolyBLEP and branching

- Only two samples per discontinuity are reshaped
- Faust does not branch: all branches of an 'if' statement are always calculated
  - Thus the amount of PolyBLEPs/sec is a function of the used sample rate
  - Required amount is a function of oscillator frequency
  - For a 500Hz rectangle at Fs=44.1k, not branching would mean 352x PolyBLEPs per second than necessary
- Had to be implemented in C++ instead of Faust



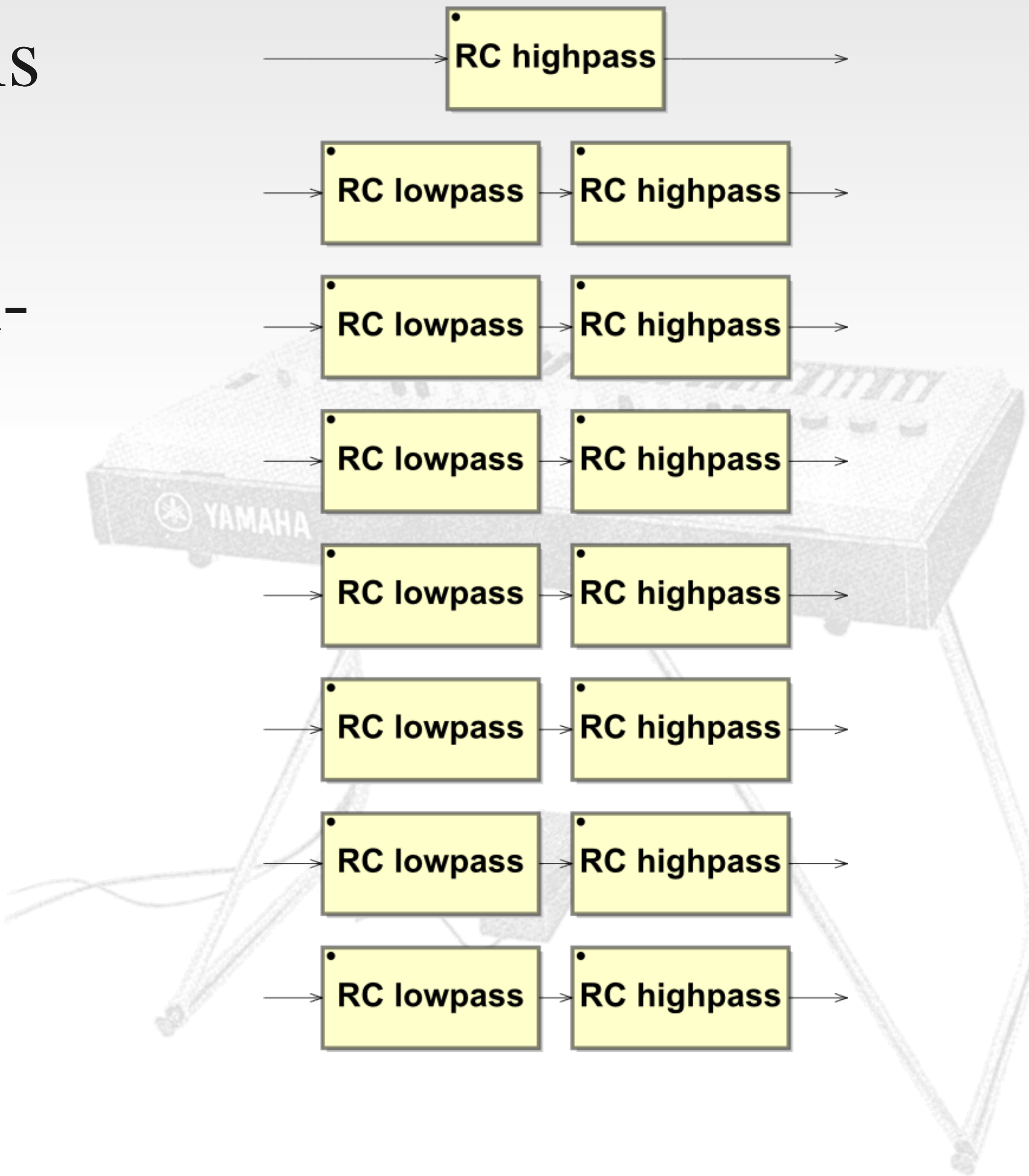
Single osc + dividers



- Common bias applies vibrato and pitch control
- Each oscillator produces both the voice and phase information
- Dividers (marked RECT) contain a phase divisor and a rectangle oscillator

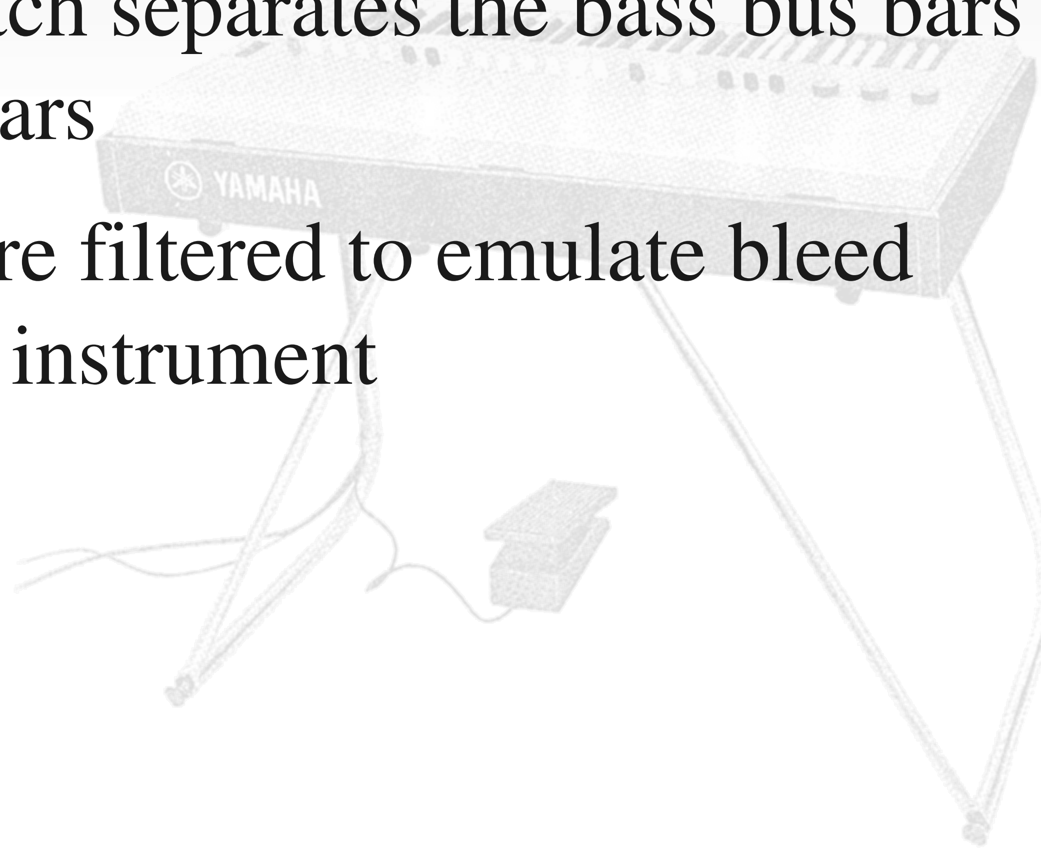
Wave transformer

- Each generated voice is filtered individually
- Main oscillator is high-passed
- Divider outputs are both low and high-passed



The keyboard

- Each key is a seven contact switch (7PST)
- Switches connect voices to bus bars
- Enabling the bass switch separates the bass bus bars from the main bus bars
- Unconnected voices are filtered to emulate bleed exhibited in the real instrument



A single bus bar

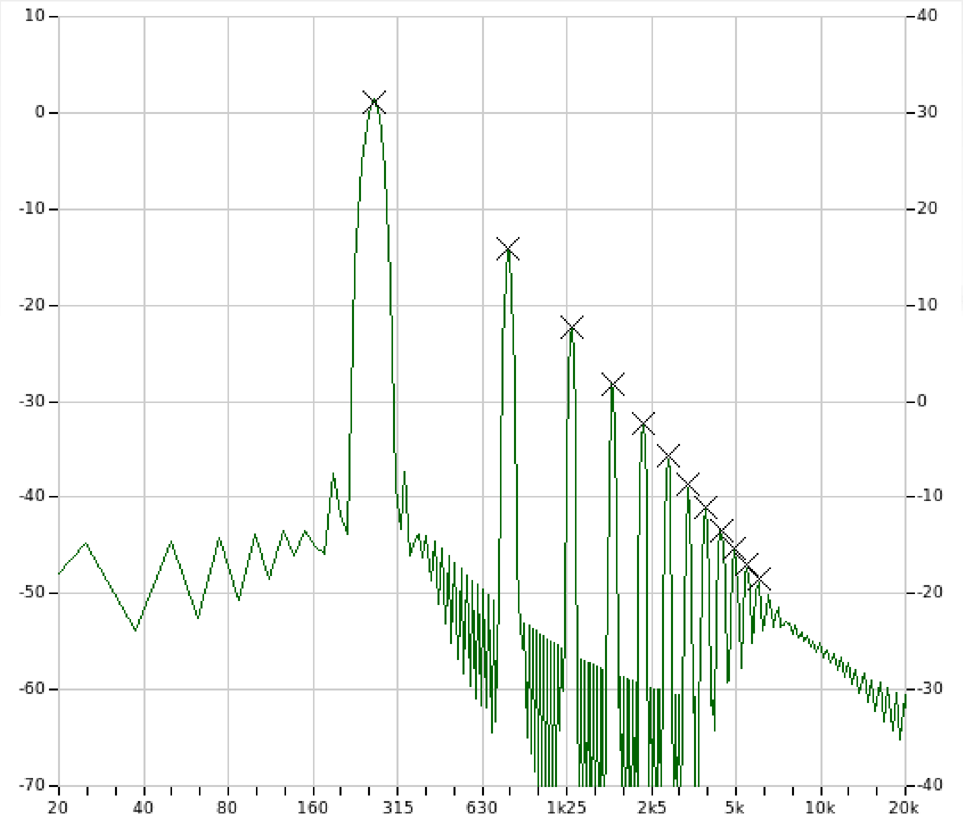
```
bus_1 =  (key_c0*c5 + key_C0*C5 + key_d0*d5 + key_D0*D5 + key_e0*e5 + key_f0*f5 +
          key_F0*F5 + key_g0*g5 + key_G0*G5 + key_a0*a5 + key_A0*A5 + key_b0*b5 +
          key_c1*c6 + key_C1*C6 + key_d1*d6 + key_D1*D6 + key_e1*e6)
       * (1.0 - bass_engaged)
       + key_f1*f6 + key_F1*F6 + key_g1*g6 + key_G1*G6 + key_a1*a6 + key_A1*A6 +
         key_b1*b6 + key_c2*c7 + key_C2*C7 + key_d2*d7 + key_D2*D7 + key_e2*e7 +
         key_f2*f7 + key_F2*F7 + key_g2*g7 + key_G2*G7 + key_a2*a7 + key_A2*A7 +
         key_b2*b7 + key_c3*c8 + key_C3*C8 + key_d3*d8 + key_D3*D8 + key_e3*e8 +
         key_f3*f8 + key_F3*F8 + key_g3*g8 + key_G3*G8 + key_a3*a8 + key_A3*A8 +
         key_b3*b8 + key_c4*c8 + key_C4*C8 + key_d4*d8 + key_D4*D8 + key_e4*e8 +
         key_f4*f8 + key_F4*F8 + key_g4*g8 + key_G4*G8 + key_a4*a8 + key_A4*A8 +
         key_b4*b8 + key_c5*c8;

bus_1_all =  (c5 + C5 + d5 + D5 + e5 + f5 + F5 + g5 + G5 + a5 + A5 + b5
            + c6 + C6 + d6 + D6 + e6) * (1.0 - bass_engaged)
                                     + f6 + F6 + g6 + G6 + a6 + A6 + b6
            + c7 + C7 + d7 + D7 + e7 + f7 + F7 + g7 + G7 + a7 + A7 + b7
            + c8 + C8 + d8 + D8 + e8 + f8 + F8 + g8 + G8 + a8 + A8 + b8
            + c8 + C8 + d8 + D8 + e8 + f8 + F8 + g8 + G8 + a8 + A8 + b8
            + c8;

bus_1_bleed = bus_1_all - bus_1 : bus_bleed_filter : apply_realism;
```

Voice sections

- Section I is trivial: there is no extra filtering
- Section II has a brightness control
  - Bus bars are separated into high-passed and low-passed streams
  - Brightness controls a mix of the two
- Bass manual drawbars are mixes of multiple bus bars
  - Not emulated perfectly – a simple lowpass is pretty good though

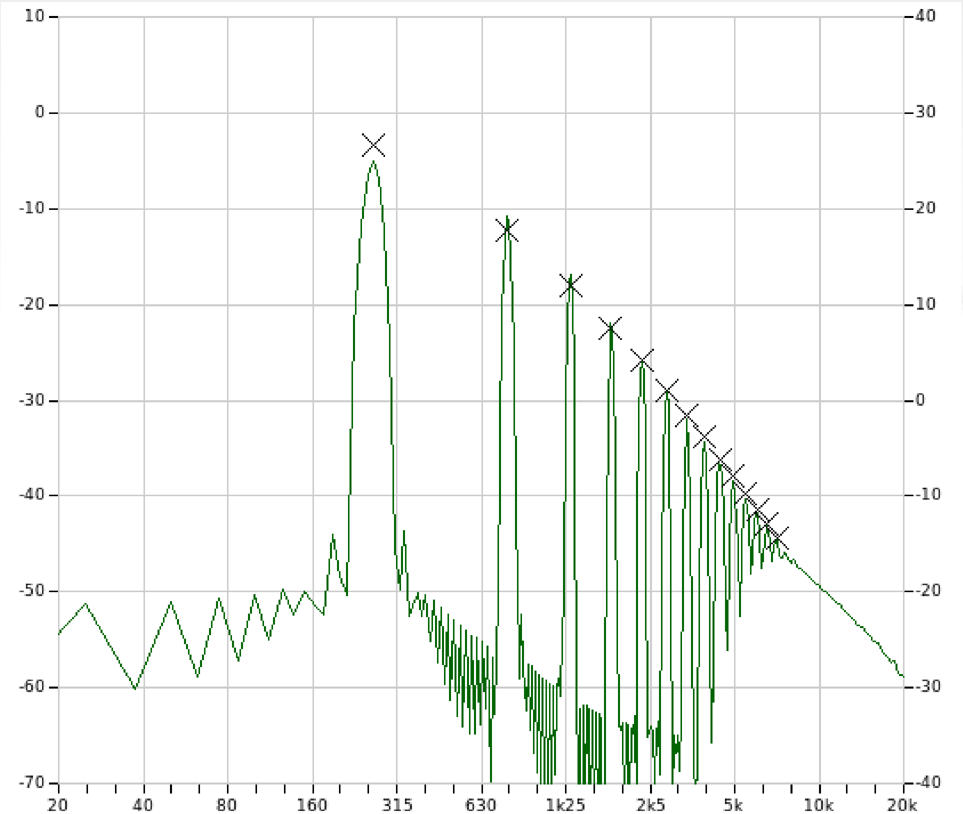
Comparison – Section I

- C2 played on 8' drawbar
- Line marks analysis of the emulation output
- Crosses mark measured harmonic peaks of the real organ
- Section I voices are emulated nearly perfectly



Comparison – Section II

- C2 played on 8' drawbar, full brightness
- Measured at the same overall volume
- Harmonics are within 3dB
- However, other notes (C3 for example) line up perfectly



Performance

- This much processing is inherently slow
- But the design leaves much room for parallelization
    - … which Faust should be good at
- Performance is sensitive to GCC flags

Performance numbers

	Core2 T7400* DSP usage	Xeon** DSP usage
Scalar	53%	52%
Vectorized	35%	41%
Scheduler (2 cores)	25%	29%
Scheduler (3 cores)	-	23%
Scheduler (4 cores)	-	20%

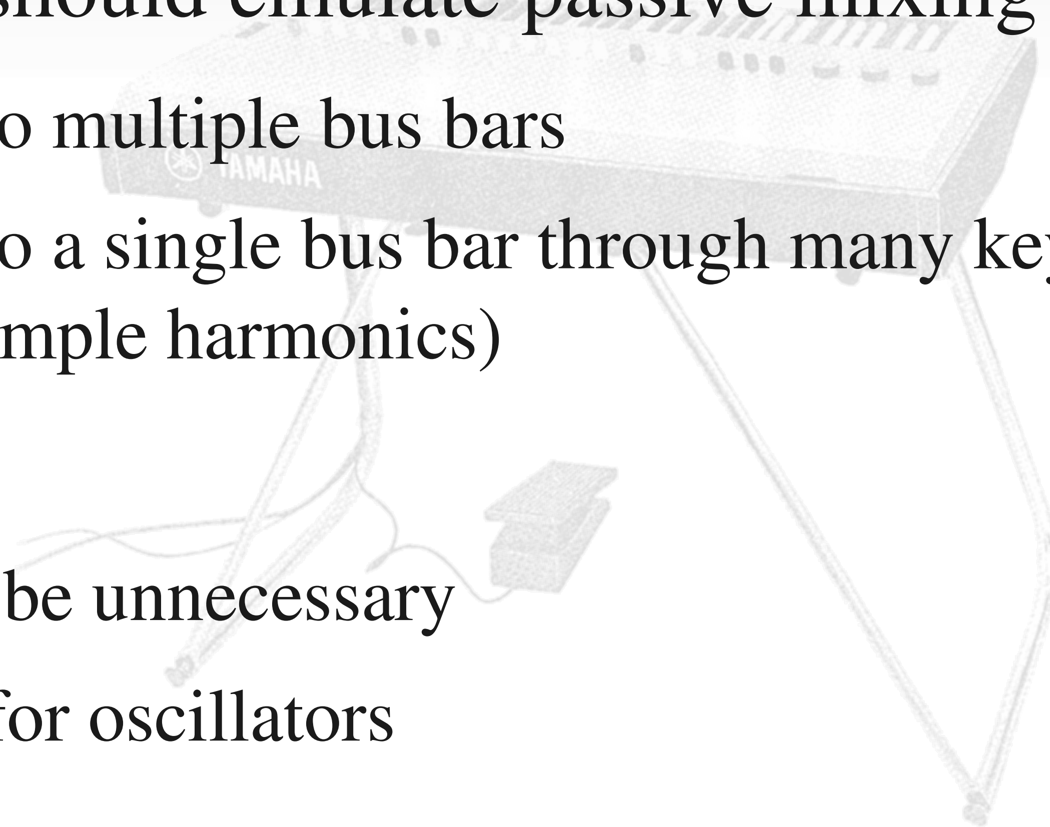
Measured while running at 2.7ms latency, Fs=48kHz, buffer size 128

* 2.16GHz Core2 T7400 running 32-bit Ubuntu 9.10

** 2 x 2Ghz Xeon dual core running 64-bit OS X 10.6.3

Things to improve

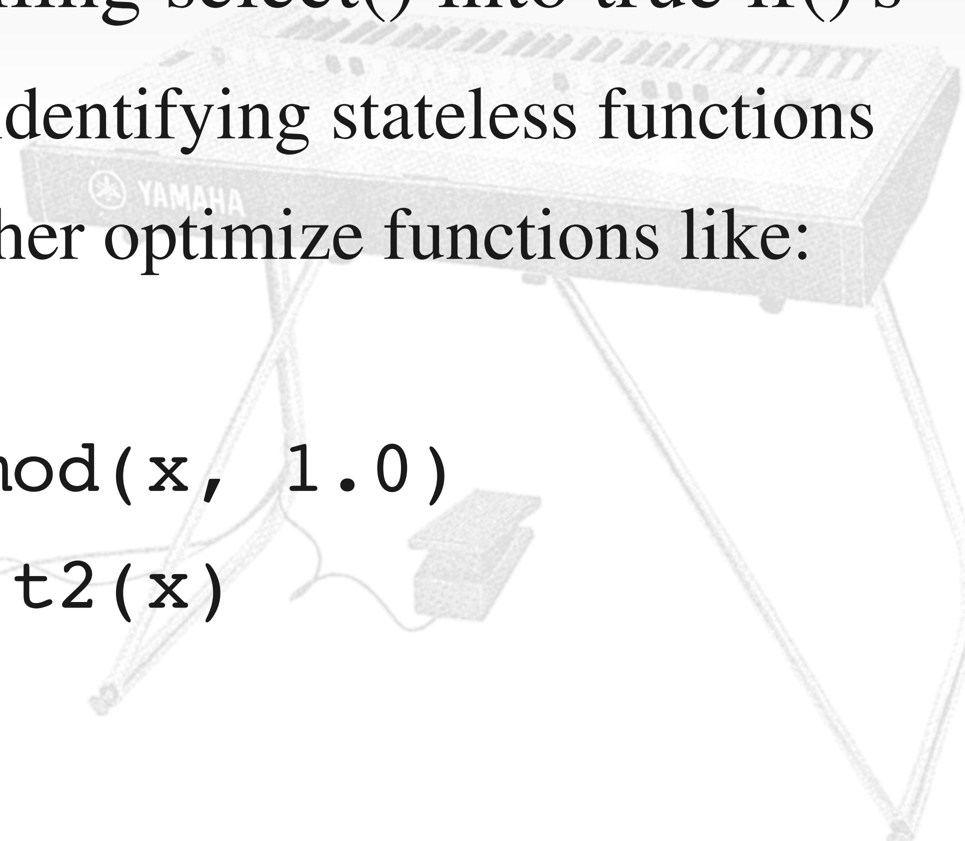
- Section II filters are not perfect
- Bass manual circuit is not truly emulated
- The keyboard matrix should emulate passive mixing
  - Voices connected to multiple bus bars
  - Voices connected to a single bus bar through many key switches (for example harmonics)
- Streamlining
  - Some filters might be unnecessary
  - Use lookup tables for oscillators



Challenges with Faust

- No true branches – PolyBLEP had to be implemented in C++
- Has been solved by compiling select() into true if()'s
  - Could be improved by identifying stateless functions
  - And generalized to further optimize functions like:

```
t1(x) = (x > 0)
t2(x) = x^2 - fmod(x, 1.0)
 f(x) = t1(x) * t2(x)
```



.. Challenges with Faust

- We need arrays!
  - Coding the keyboard mixer was a PITA
  - Generated code can not be optimized
- Compiling a complex project is slow
  - Scalar 11s, vector 5 min, scheduler <u>8 min</u>
  - process() into subroutines? Less register overload...
- Issues with vec and sch modes (solved partly)

.. Challenges with Faust

- Naming should be more strict, the following is ambiguous but perfectly legal:

```
f(a, b) = a + b
with { a = b * b; };
```

- Compiler error messages can be very unhelpful
  - 67kB error messages have been spotted
- Language documentation should be improved

Benefits of using Faust

- Functional programming is an excellent model for signal processing
- Parallelization is the future!
- .. but realtime parallel programming requires a level of expertise uncommon even for seasoned programmers
  - Let alone people whose career focus is in what these programs actually do (the DSP)!

.. Benefits of using Faust

- Faust code is readable

  gain(a) : distortion(f) : attenuation;

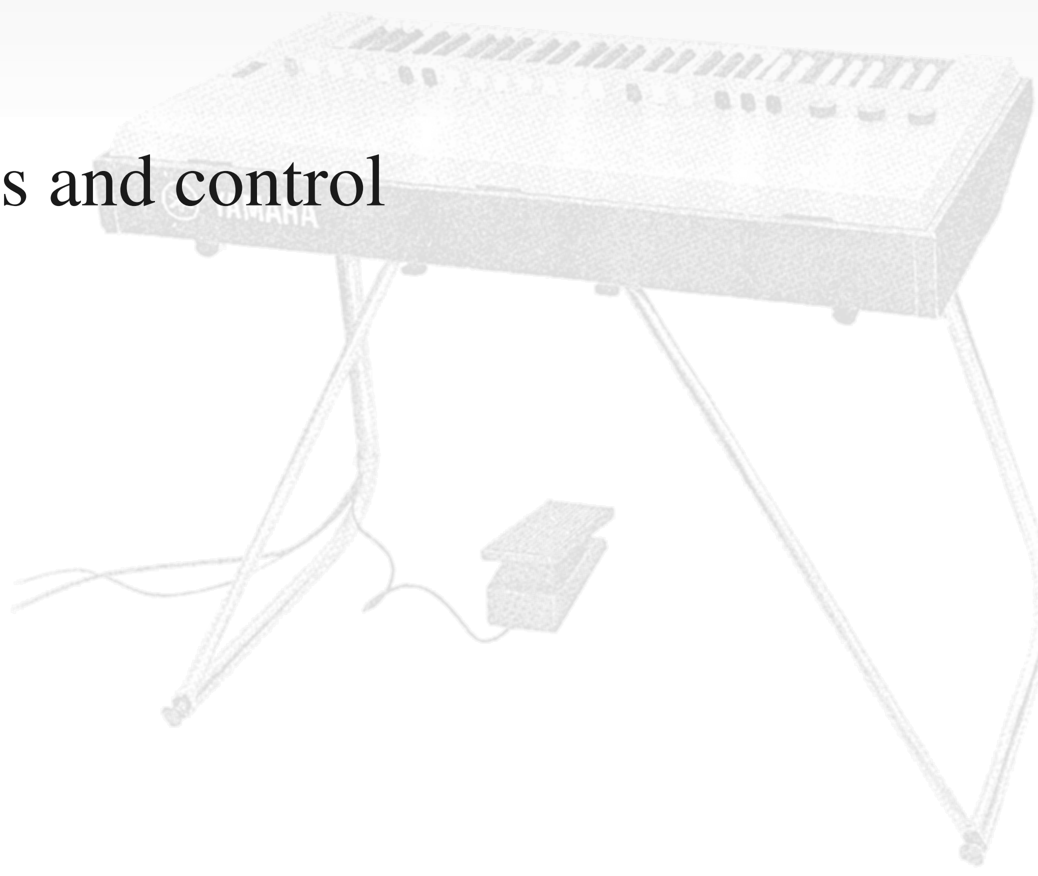
  vs

  attenuation(distortion(f, gain(a, signal)));

- SVG output is a very helpful tool
- Code re-use is a reality
  - Combining C/C++ modules from different sources often requires refactoring or runtime data conversion

… introducing

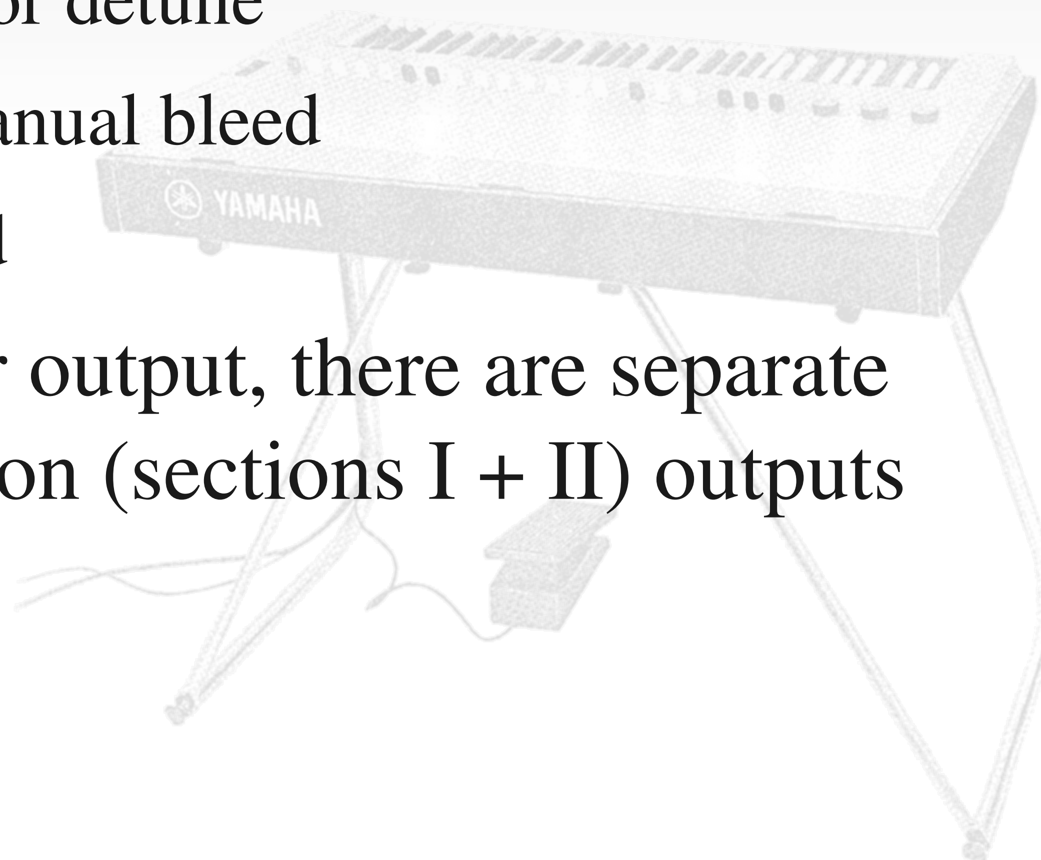
Foo YC-20



- Jack audio and MIDI
  - MIDI for both notes and control
- Gtkmm/Cairo UI
- Realism switch
- 1.0 released … now

Miscellaneous features

- Realism switch
    - Off: nothing extra
    - 2/4: slight oscillator detune
    - 3/4: percussion manual bleed
    - 4/4: drawbar bleed
- Addition to the master output, there are separate bass and treble section (sections I + II) outputs



demo

Thanks for listening

- I would like to thank Petri Junno, Stéphane Letz, Yann Orlarey, Thorsten Wilms, Torben Hohn, Sakari Bergen, Edgar Aichinger

http://code.google.com/p/foo-yc20