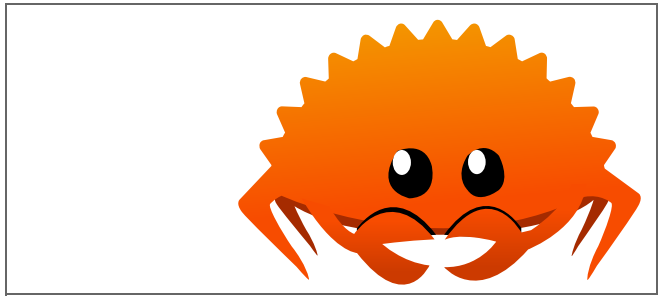
Experiences with Faust + Rust

FAUST



whoami

- @obsoleszenz
- Techno DJ
- Tinkerer
- Exploring&Learning
- Means:
  - I don't know shit

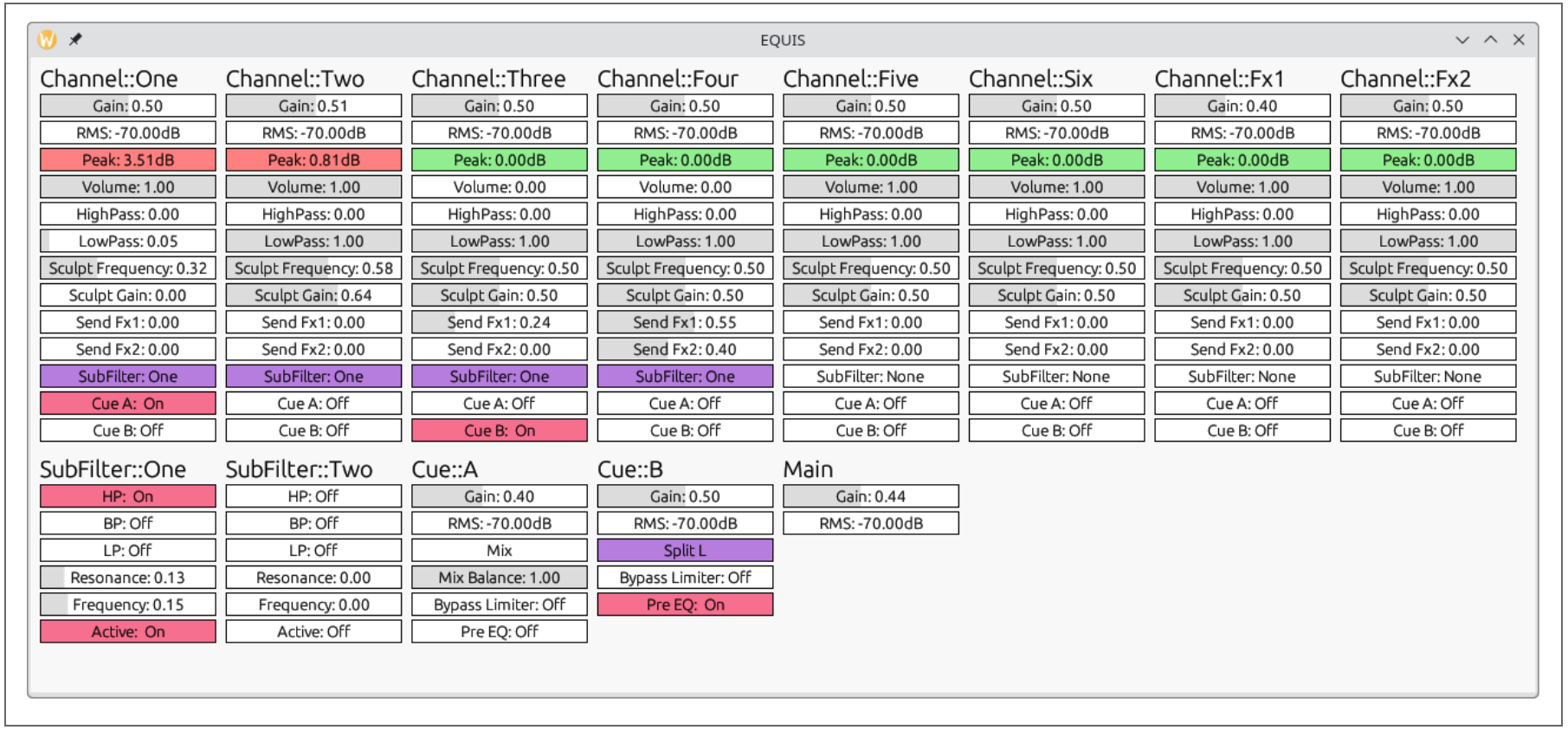

Make music
Develop
a new thing
to make music
JAKE-CLARK.TUMBLR
imgflip.com

My Projects

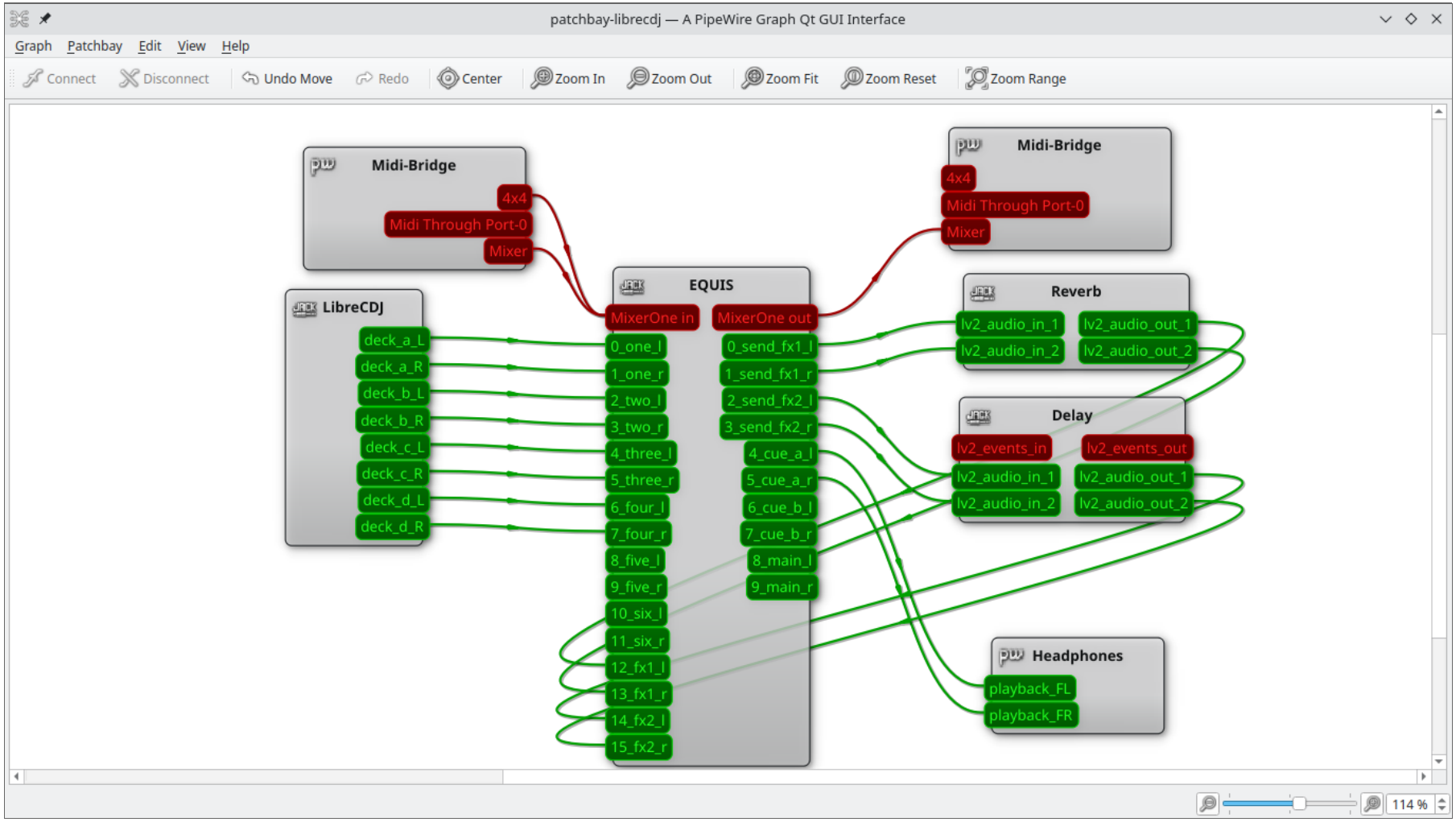
EQUIS

- DJ Mixer as Software
- Filters, filters, filters
- Inspired by Allen&Heath & MODEL1
- Implemented in FAUST + Rust

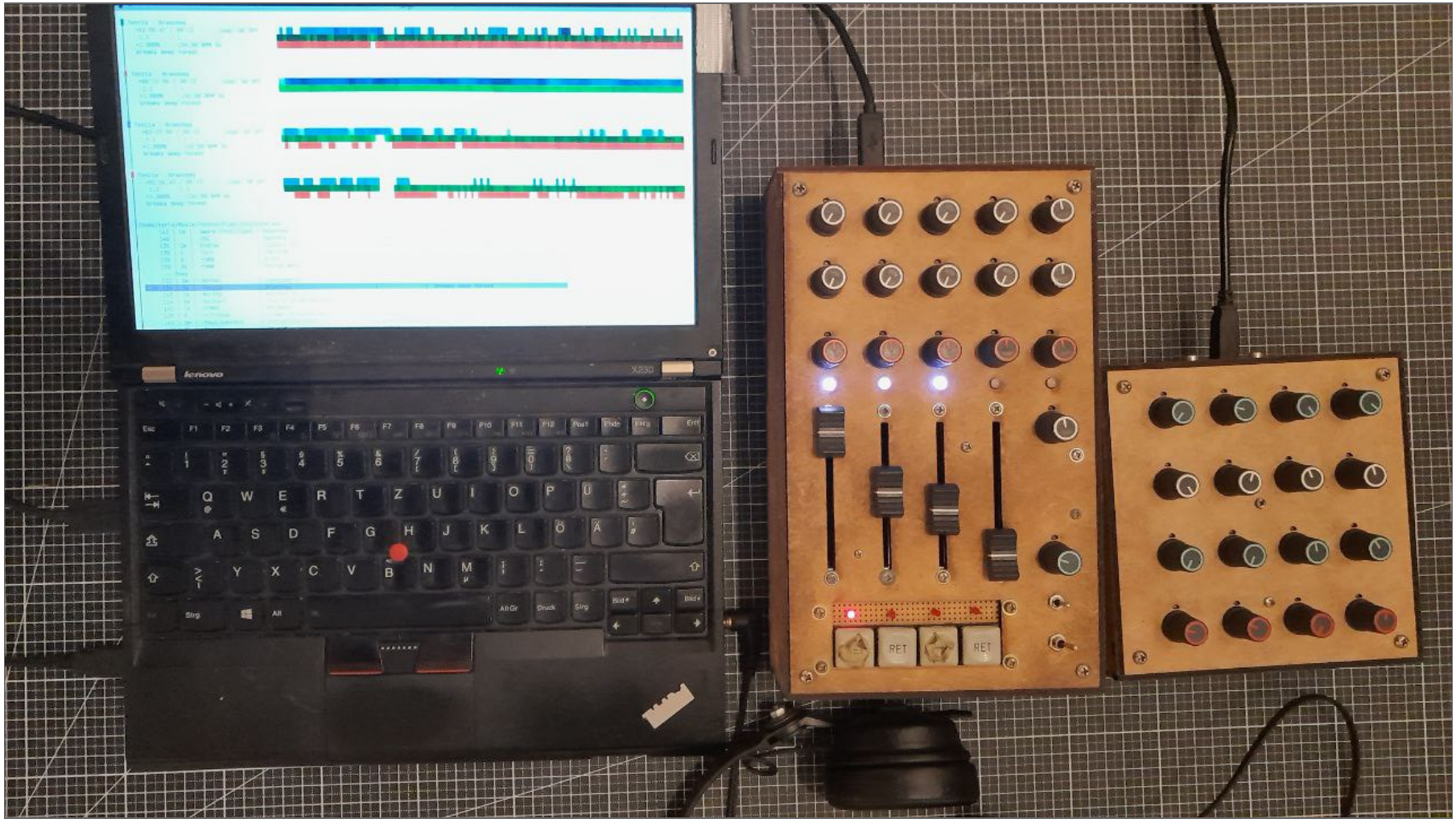
EQUIS


EQUIS
Channel::One
Gain: 0.50
RMS: -70.00dB
Peak: 3.51dB
Volume: 1.00
HighPass: 0.00
LowPass: 0.05
Sculpt Frequency: 0.32
Sculpt Gain: 0.00
Send Fx1: 0.00
Send Fx2: 0.00
SubFilter: One
Cue A: On
Cue B: Off
Channel::Two
Gain: 0.51
RMS: -70.00dB
Peak: 0.81dB
Volume: 1.00
HighPass: 0.00
LowPass: 1.00
Sculpt Frequency: 0.58
Sculpt Gain: 0.64
Send Fx1: 0.00
Send Fx2: 0.00
SubFilter: One
Cue A: Off
Cue B: Off
Channel::Three
Gain: 0.50
RMS: -70.00dB
Peak: 0.00dB
Volume: 0.00
HighPass: 0.00
LowPass: 1.00
Sculpt Frequency: 0.50
Sculpt Gain: 0.50
Send Fx1: 0.24
Send Fx2: 0.00
SubFilter: One
Cue A: Off
Cue B: On
Channel::Four
Gain: 0.50
RMS: -70.00dB
Peak: 0.00dB
Volume: 0.00
HighPass: 0.00
LowPass: 1.00
Sculpt Frequency: 0.50
Sculpt Gain: 0.50
Send Fx1: 0.55
Send Fx2: 0.40
SubFilter: One
Cue A: Off
Cue B: Off
Channel::Five
Gain: 0.50
RMS: -70.00dB
Peak: 0.00dB
Volume: 1.00
HighPass: 0.00
LowPass: 1.00
Sculpt Frequency: 0.50
Sculpt Gain: 0.50
Send Fx1: 0.00
Send Fx2: 0.00
SubFilter: None
Cue A: Off
Cue B: Off
Channel::Six
Gain: 0.50
RMS: -70.00dB
Peak: 0.00dB
Volume: 1.00
HighPass: 0.00
LowPass: 1.00
Sculpt Frequency: 0.50
Sculpt Gain: 0.50
Send Fx1: 0.00
Send Fx2: 0.00
SubFilter: None
Cue A: Off
Cue B: Off
Channel::Fx1
Gain: 0.40
RMS: -70.00dB
Peak: 0.00dB
Volume: 1.00
HighPass: 0.00
LowPass: 1.00
Sculpt Frequency: 0.50
Sculpt Gain: 0.50
Send Fx1: 0.00
Send Fx2: 0.00
SubFilter: None
Cue A: Off
Cue B: Off
Channel::Fx2
Gain: 0.50
RMS: -70.00dB
Peak: 0.00dB
Volume: 1.00
HighPass: 0.00
LowPass: 1.00
Sculpt Frequency: 0.50
Sculpt Gain: 0.50
Send Fx1: 0.00
Send Fx2: 0.00
SubFilter: None
Cue A: Off
Cue B: Off
SubFilter::One
HP: On
BP: Off
LP: Off
Resonance: 0.13
Frequency: 0.15
Active: On
SubFilter::Two
HP: Off
BP: Off
LP: Off
Resonance: 0.00
Frequency: 0.00
Active: Off
Cue::A
Gain: 0.40
RMS: -70.00dB
Mix
Mix Balance: 1.00
Bypass Limiter: Off
Pre EQ: Off
Cue::B
Gain: 0.50
RMS: -70.00dB
Split L
Bypass Limiter: Off
Pre EQ: On
Main
Gain: 0.44
RMS: -70.00dB

EQUIS

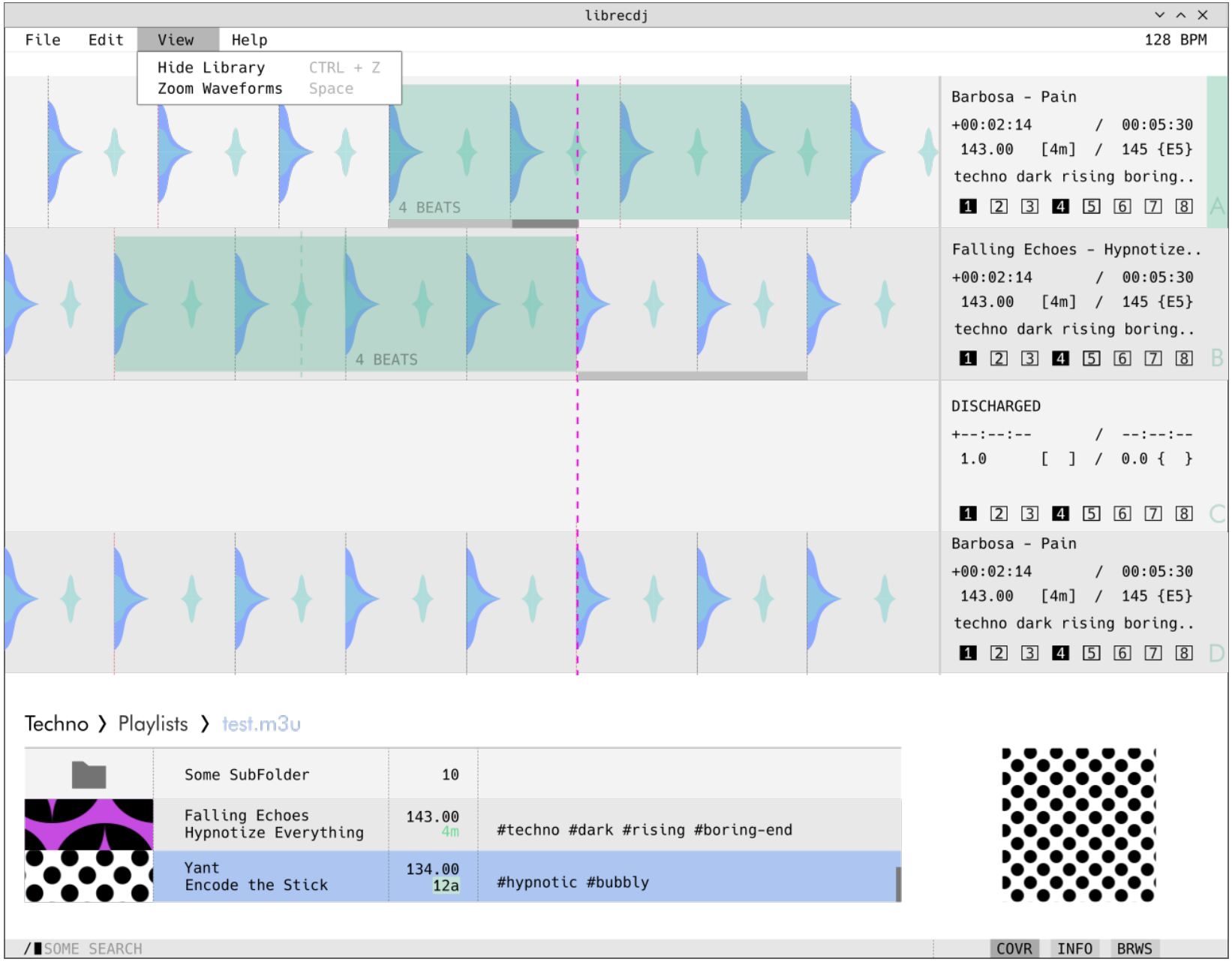

patchbay-librecdj — A PipeWire Graph Qt GUI Interface
Graph
Patchbay
Edit
View
Help
Connect
Disconnect
Undo Move
Redo
Center
Zoom In
Zoom Out
Zoom Fit
Zoom Reset
Zoom Range
Midi-Bridge
4x4
Midi Through Port-0
Mixer
Midi-Bridge
4x4
Midi Through Port-0
Mixer
LibreCDJ
deck_a_L
deck_a_R
deck_b_L
deck_b_R
deck_c_L
deck_c_R
deck_d_L
deck_d_R
EQUIS
MixerOne in
MixerOne out
0_one_l
1_one_r
2_two_l
3_two_r
4_three_l
5_three_r
6_four_l
7_four_r
8_five_l
9_five_r
10_six_l
11_six_r
12_fx1_l
13_fx1_r
14_fx2_l
15_fx2_r
0_send_fx1_l
1_send_fx1_r
2_send_fx2_l
3_send_fx2_r
4_cue_a_l
5_cue_a_r
6_cue_b_l
7_cue_b_r
8_main_l
9_main_r
Reverb
lv2_audio_in_1
lv2_audio_in_2
lv2_audio_out_1
lv2_audio_out_2
Delay
lv2_events_in
lv2_events_out
lv2_audio_in_1
lv2_audio_in_2
lv2_audio_out_1
lv2_audio_out_2
Headphones
playback_FL
playback_FR
114 %

EQUIS



LibreCDJ

- Also written in rust
- Free Open Source DJ Software
- Reliable & good sounding
- Club Ready™
- Fun to use
- Alternative to proprietary + expensive solutions


librecdj
File
Edit
View
Help
128 BPM
Hide Library CTRL + Z
Zoom Waveforms Space
4 BEATS
Barbosa - Pain
+00:02:14 / 00:05:30
143.00 [4m] / 145 {E5}
techno dark rising boring..
1 2 3 4 5 6 7 8
A
4 BEATS
Falling Echoes - Hypnotize..
+00:02:14 / 00:05:30
143.00 [4m] / 145 {E5}
techno dark rising boring..
1 2 3 4 5 6 7 8
B
DISCHARGED
+--:--:-- / --:--:--
1.0 [ ] / 0.0 { }
1 2 3 4 5 6 7 8
C
Barbosa - Pain
+00:02:14 / 00:05:30
143.00 [4m] / 145 {E5}
techno dark rising boring..
1 2 3 4 5 6 7 8
D
Techno ❯ Playlists ❯ test.m3u
Some SubFolder 10
Falling Echoes
Hypnotize Everything
143.00
4m
#techno #dark #rising #boring-end
Yant
Encode the Stick
134.00
12a
#hypnotic #bubbly
/SOME SEARCH
COVR
INFO
BRWS

FAUST + Rust related projects

- Maintaining
  - faust-build
  - faust-types
- lowpass-lr4-faust-nih-plug

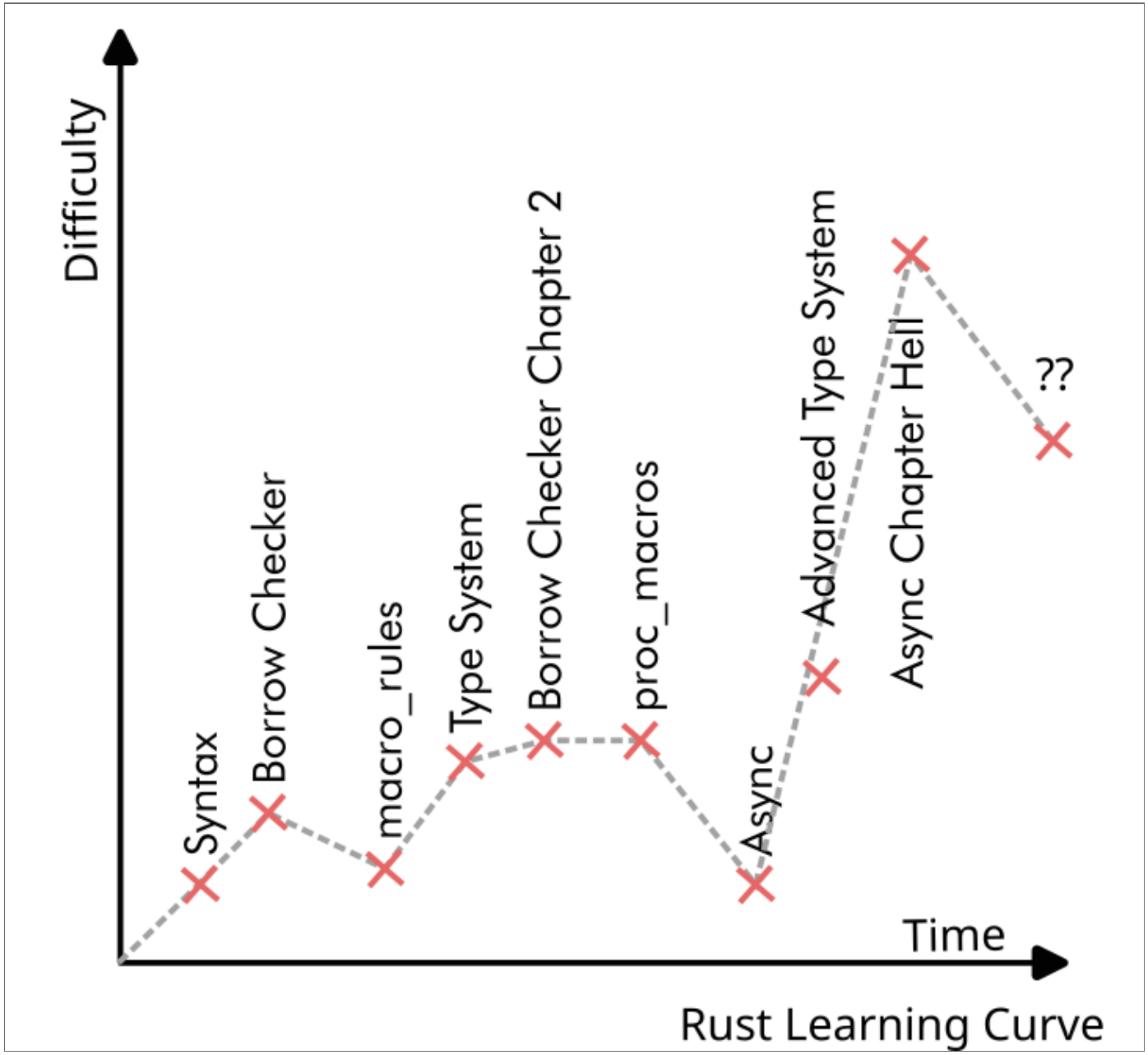
Sounds interesting? Join us!

We are looking for likeminded hackers, designers, musicians + friends
Find us on codeberg/github or send me an e-mail

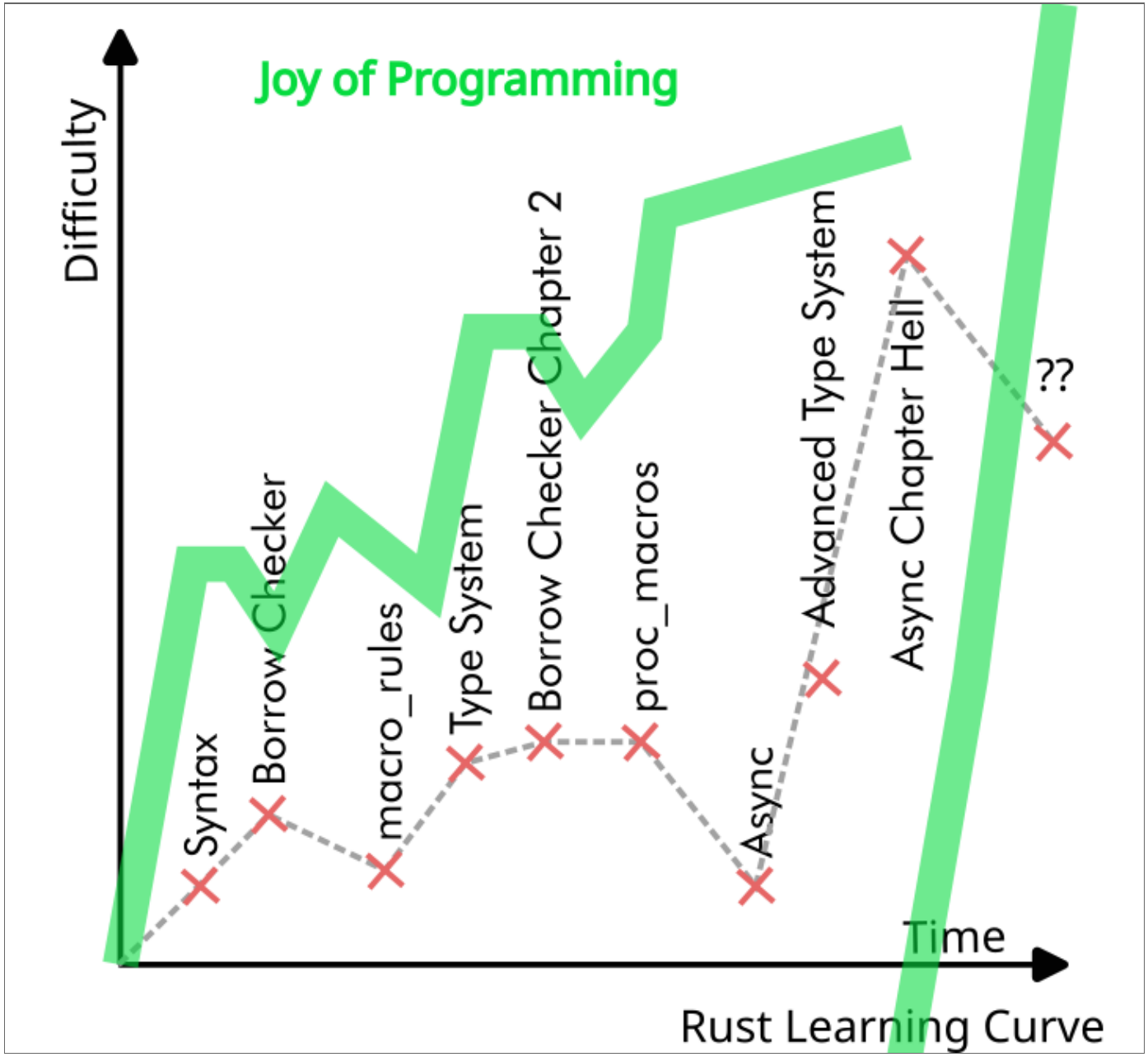
Links

- codeberg.org/obsoleszenz
- soundcloud.com/obsoleszenz
- @obsoleszenz@nerdculture.de
- obsoleszenz@riseup.net

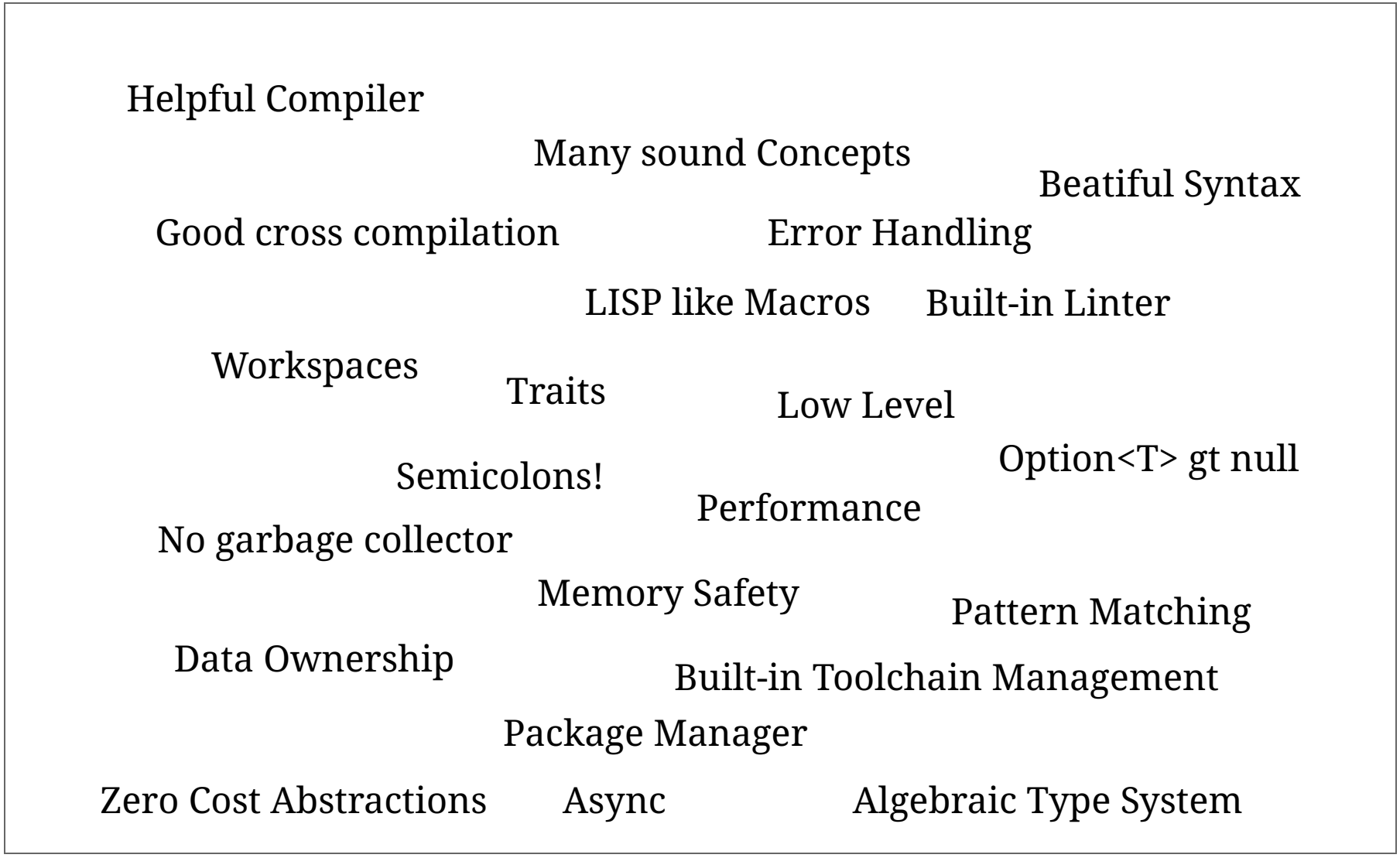
Why NOT rust?


Difficulty
Syntax
Borrow Checker
macro_rules
Type System
Borrow Checker Chapter 2
proc_macros
Async
Advanced Type System
Async Chapter Hell
??
Time
Rust Learning Curve

Why rust?


Joy of Programming
Difficulty
Time
Rust Learning Curve
Syntax
Borrow Checker
macro_rules
Type System
Borrow Checker Chapter 2
proc_macros
Async
Advanced Type System
Async Chapter Hell
??

Why rust?


Helpful Compiler
Many sound Concepts
Beatiful Syntax
Good cross compilation
Error Handling
LISP like Macros
Built-in Linter
Workspaces
Traits
Low Level
Option<T> gt null
Semicolons!
Performance
No garbage collector
Memory Safety
Pattern Matching
Data Ownership
Built-in Toolchain Management
Package Manager
Zero Cost Abstractions
Async
Algebraic Type System

Why rust?

- RUST is not a better C/C++ but a new way of programming

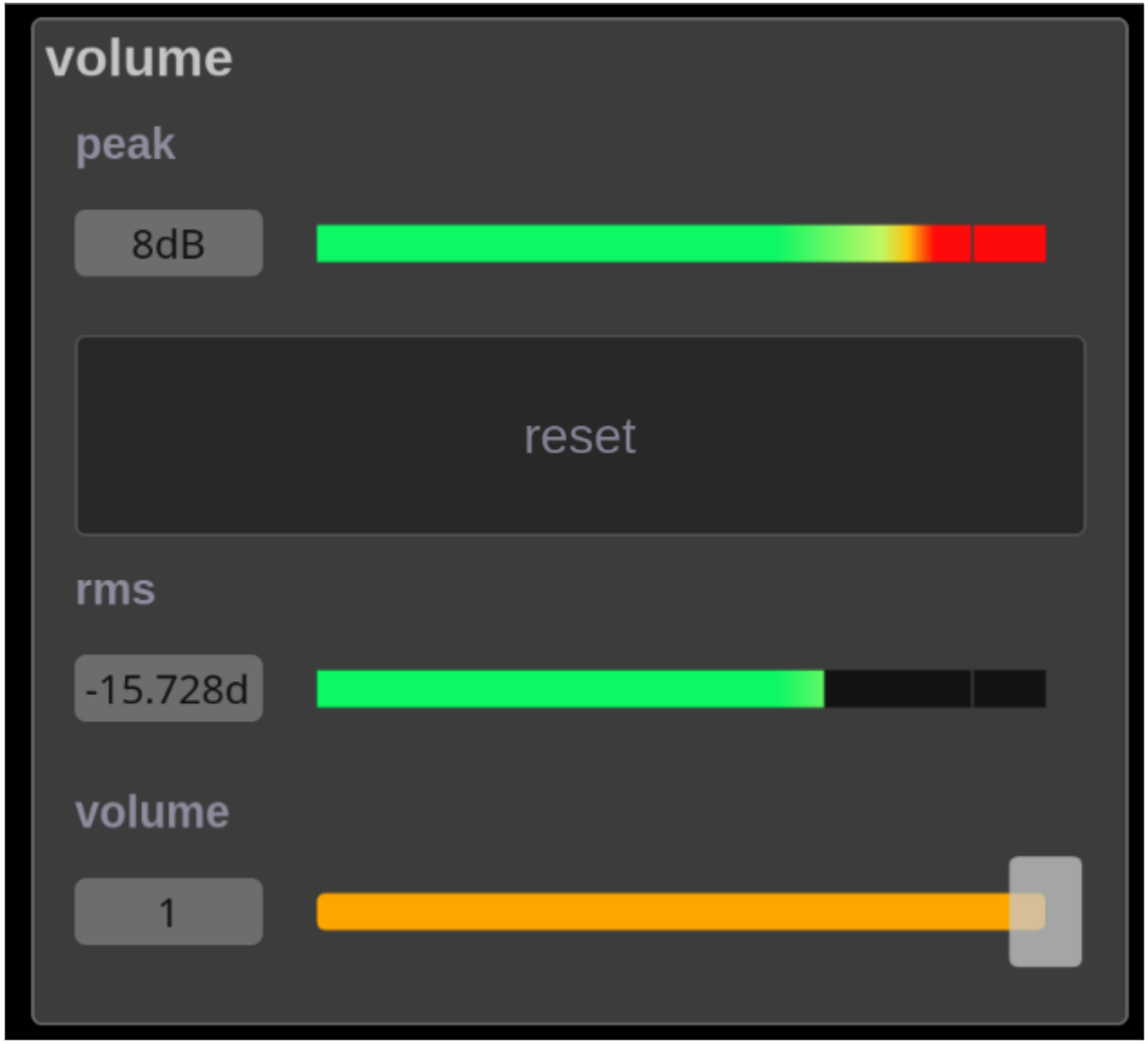
Why Faust?

- Beautiful syntax & concept
- Fast prototyping & exploration
- Reliable in production
- Good standard library & ecosystem

combine(faust, rust)

- Both progressive/functional languages
- Runs almost everwhere
  - Linux/FreeBSD/Mac/Win/Android/ios/arduino/vst/lv2/clap...
- Powerful modern system language + powerful DSP language
- Combining the Ecosystems

Let's build something!



How do I do it?

- Dependencies
  - rust-faust ecosystem
  - jack
  - egui
- **codeberg.org/obsoleszenz/rust-faust-jack-egui-template**

Project Structure

```
├── Cargo.toml
└── crates
    ├── dsp
    │   ├── build.rs
    │   ├── Cargo.toml
    │   ├── dsp
    │   │   └── main.dsp
    │   └── src/*.rs
    ├── backend
    │   ├── Cargo.toml
    │   └── src/*.rs
    └── gui
        ├── Cargo.toml
        └── src/*.rs
```

DSP: Adding dependencies

- ```
  cargo add faust-build faust-types
  ```
- Checkout **https://github.com/Frando/rust-faust**
  ```

DSP: Add some faust

- Add faust code into dsp/dsp/main.dsp

```
import("stdfaust.lib");

p_volume = hslider("volume", 1.0, 0.0, 1.0, 0.01) : si.smoo;
p_peak_reset = button("reset");

holdMax(resetTrigger) =  _ : (_,_:resettableMax(resetTrigger))~_
with {
    resettableMax(resetTrigger) = _,_ <: max,(!,_) :
select2(resetTrigger);
};

stereo2mono = _,_ :> abs : _  / 2 ;

mono_dbmeter(reset) = max(ba.db2linear(MIN_DB)) : ba.linear2db :
min(MAX_DB)  : max ~ -(80.0/ma.SR) : holdMax(ba.pulse(HOLD_PEAK))
    <:
  ,
  holdMax(reset)
    :
  hbargraph("rms[unit:dB]", MIN_DB, MAX_DB),
  hbargraph("peak[unit:dB]", MIN_DB, MAX_DB)
    :>

with {
  MIN_DB = -70;
  MAX_DB = 8;
  // window size for RMS
  WINDOW_SIZE = 100;
  // hold time for peak value in samples
  HOLD_PEAK = ma.SR/8;
};
```

```
dbmeter(reset) = _,_ <: _,_, stereo2mono : _, attach(_,
mono_dbmeter(reset));

process = par(i, 2, _ * p_volume) : dbmeter(p_peak_reset);
```

DSP: faust-build

crates/dsp/build.rs

```
fn main() {
  #[cfg(feature = "faust-rebuild")]
  faust_build::FaustBuilder::new("dsp/main.dsp", "src/faust_dsp.rs")
    .set_struct_name("FaustDSP")
    .build();
}
```

DSP: faust-build

- Define faust-rebuild feature in crates/dsp/Cargo.toml
- ```
  cargo -p dsp --features faust-rebuild
  ```
- This will generate crates/dsp/src/faust_dsp.rs
  ```

DSP: Wrapping

- generated rust code is not idiomatic
- until that's fixed we wrap it!
- Check out crop2000 prs & issues

- ```
  pub struct DSP {
      faust_dsp: FaustDSP,
  }

  impl DSP {
      ...
  }
  ```
  ```

DSP: Wrapping Basics

```
impl DSP {
    pub fn new() -> Self {
        Self {
            faust_dsp: FaustDSP::new(),
        }
    }
    pub fn set_sample_rate(&mut self, sample_rate: usize) {
        self.faust_dsp.init(sample_rate as i32);
    }

    pub fn sample_rate(&self) -> usize {
        self.faust_dsp.get_sample_rate() as usize
    }
}
```

DSP: Wrapping Parameters

```
impl FaustDsp for FaustDSP {
      fn build_user_interface_static(ui_interface: &mut dyn UI<Self::T>) {
        ui_interface.open_vertical_box("Volume");
        ui_interface.declare(Some(ParamIndex(0)), "unit", "dB");
        ui_interface.add_horizontal_bargraph("peak", ParamIndex(0),
-7e+01, 8.0);
        ui_interface.add_button("reset", ParamIndex(1));
        ui_interface.declare(Some(ParamIndex(2)), "unit", "dB");
        ui_interface.add_horizontal_bargraph("rms", ParamIndex(2), -7e+01,
8.0);
        ui_interface.add_horizontal_slider("volume", ParamIndex(3), 1.0,
0.0, 1.0, 0.01);
        ui_interface.close_box();
      }
}
```

DSP: Wrapping Parameters

```rust
use faust_types::ParamIndex;

#[derive(Debug, PartialEq, Eq)]
pub enum Parameter {
    Volume,
    Reset,
}

impl From<Parameter> for ParamIndex {
    fn from(value: Parameter) -> ParamIndex {
        match value {
            Parameter::Volume => ParamIndex(3),
            Parameter::Reset => ParamIndex(1),
        }
    }
}
```

DSP: Wrapping UI Parameters

```rust
pub enum UIParameter {
    Rms,
    Peak,
}

impl From<UIParameter> for ParamIndex {
    fn from(value: UIParameter) -> ParamIndex {
        match value {
            UIParameter::Rms => ParamIndex(2),
            UIParameter::Peak => ParamIndex(0),
        }
    }
}
```

DSP: Wrapping Parameters

- ```
  impl DSP {
      pub fn set_parameter(&mut self, parameter: Parameter, value: f32) {
          self.faust_dsp.set_param(parameter.into(), value);
      }

      pub fn parameter(&mut self, parameter: Parameter) -> f32 {
          self.faust_dsp
              .get_param(parameter.into())
              .expect("Invalid parameter index")
      }

      pub fn ui_parameter(&self, ui_parameter: UIParameter) -> f32 {
          self.faust_dsp
              .get_param(ui_parameter.into())
              .expect("Invalid parameter index")
      }
  }
  ```
  ```

DSP: Problems with Faust compute

- ```
  inputs: &[&[f32]]
  ```

- Compiler cannot verify input/output buffer length
- A slice of slices is annoying to use
- count type is i32 instead of usize
  ```

DSP: Wrapping compute

```
impl DSP {
    pub const INPUTS: usize = 2;
    pub const OUTPUTS: usize = 2;

    pub fn compute(
        &mut self,
        count: usize,
        inputs: &[I; Self::INPUTS],
        outputs: &mut [O; Self::OUTPUTS],
    ) where
        I: AsRef<[f32]>,
        O: AsMut<[f32]>,
    {
        // We need to change the memory layout of inputs/outputs so
        // that FaustDSP likes it.
        // As this is a bit ugly I hope we can move this to the faust
        // generated code at some point.
        let inputs: [&[f32]; Self::INPUTS] = [
            inputs[0].as_ref(),
            inputs[1].as_ref()
        ];

        let mut outputs_iter = outputs.iter_mut();
        let mut outputs: [&mut [f32]; Self::OUTPUTS] = [
            outputs_iter.next().unwrap().as_mut(),
            outputs_iter.next().unwrap().as_mut(),
        ];

        self.faust_dsp.compute(
          count as i32,
          inputs.as_slice(),
          outputs.as_mut_slice()
        );
    }
}
```

DSP: State

```rust
use crate::{parameter::UIParameter, Parameter, DSP};

#[derive(Debug, Clone, PartialEq, Default)]
pub struct DSPState {
    pub rms: f32,
    pub peak: f32,
    pub p_volume: f32,
    pub p_reset: f32,
}

impl DSPState {
    pub fn update(&mut self, dsp: &mut DSP) {
        self.rms = dsp.ui_parameter(UIParameter::Rms);
        self.peak = dsp.ui_parameter(UIParameter::Peak);
        self.p_volume = dsp.parameter(Parameter::Volume);
        self.p_reset = dsp.parameter(Parameter::Reset);
    }
}
```

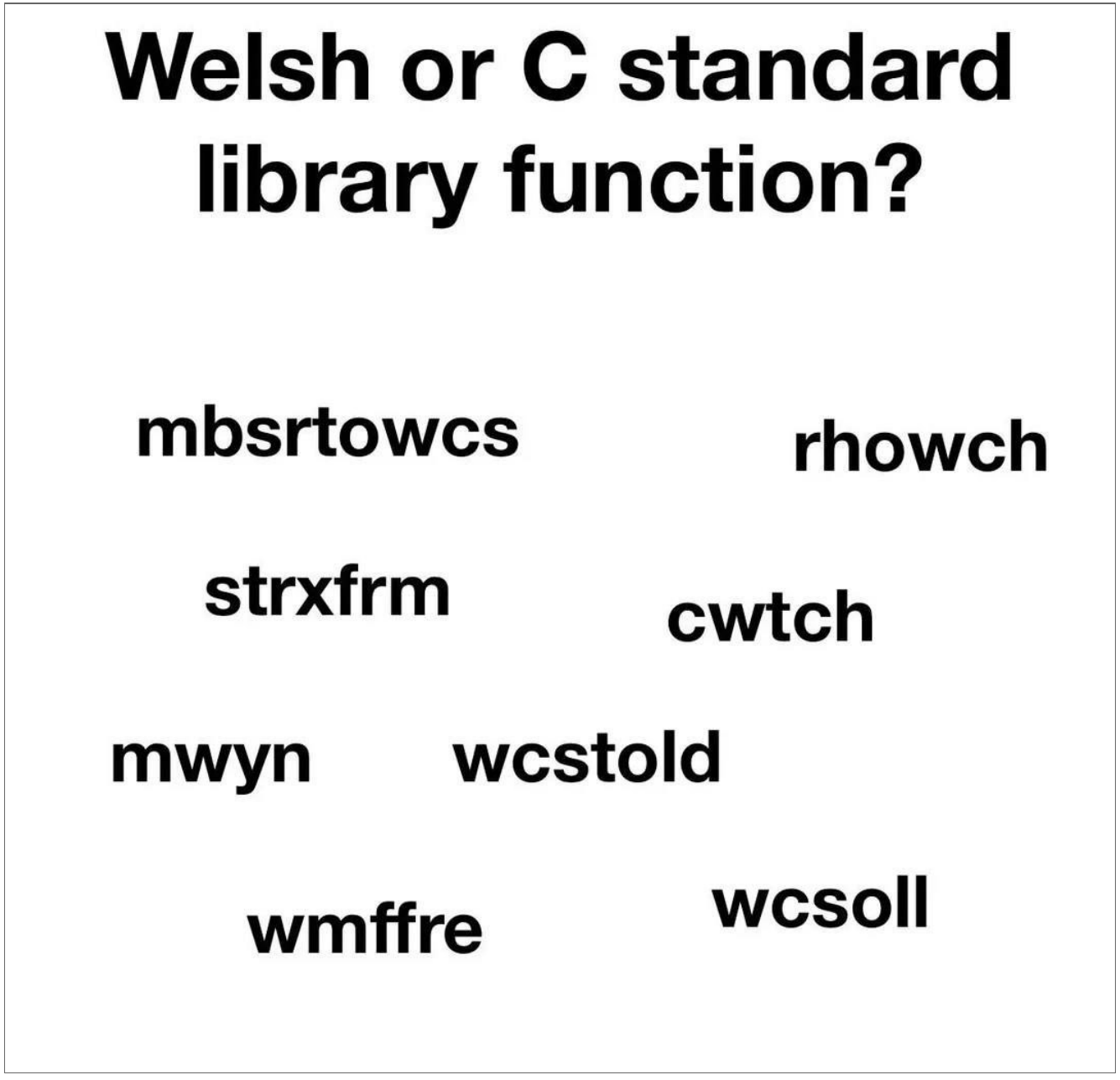
DSP: State

```
impl DSP {
    pub fn update_state(&mut self, dsp_state: &mut DSPState) {
        dsp_state.update(self)
    }
}
```

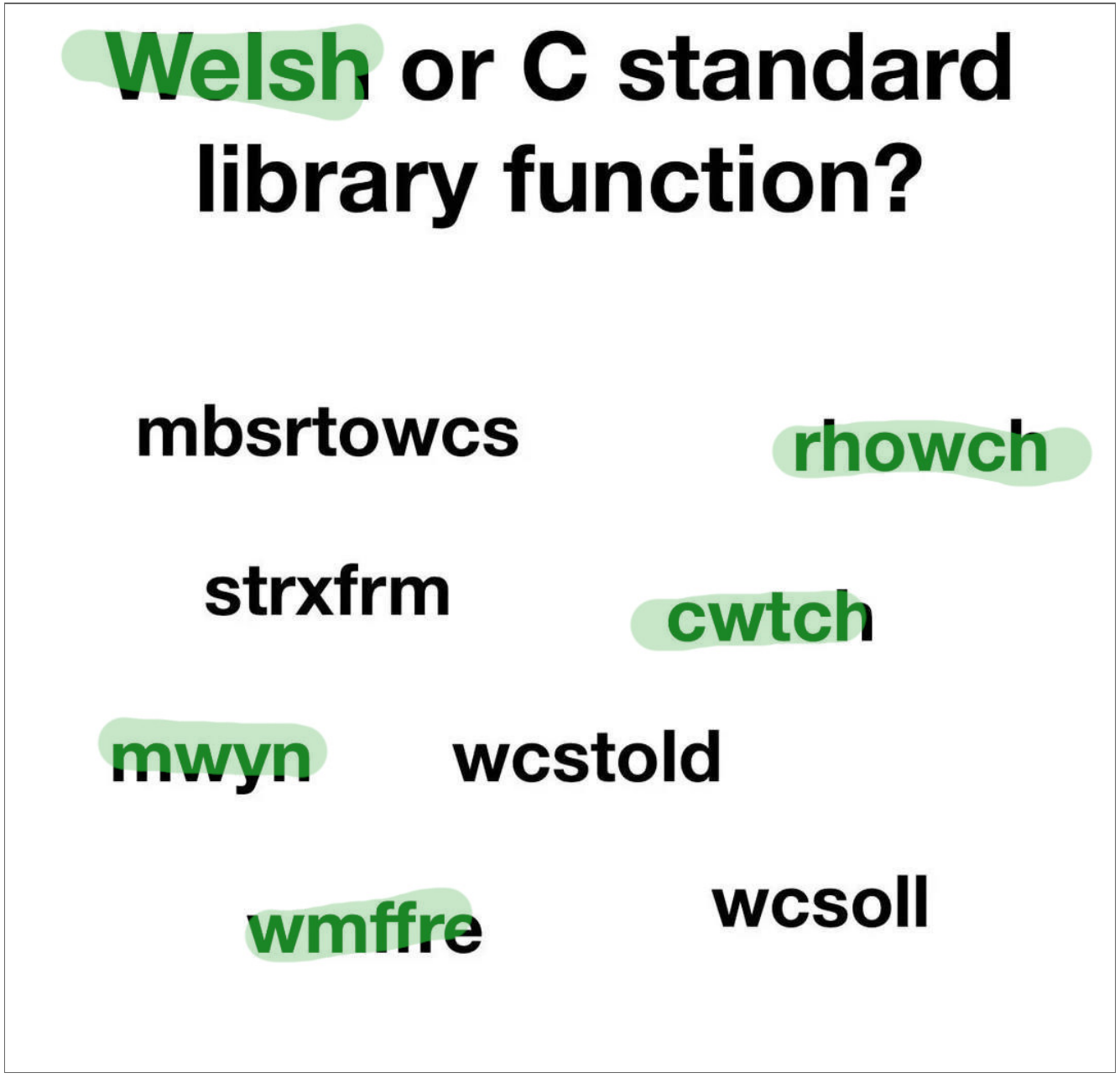
DSP: Usage examples

```rust
let mut dsp = DSP::new();
let dsp.set_sample_rate(44_800);

// Example with arrays
const MAX_BUFFER: usize = 1024;
let inputs = [[0.0_f32; MAX_BUFFER]; DSP::inputs()];
let mut outputs = [[0.0_f32; MAX_BUFFER]; DSP::outputs()];
dsp.compute(128, &inputs, &mut outputs);

// Example with resizable vectors
let max_buffer_size = 1024;
let mut inputs = vec![vec![0_f32; max_buffer_size]; DSP::INPUTS];
let mut outputs = vec![vec![0_f32; max_buffer_size]; DSP::OUTPUTS];
dsp.compute(
  128,
  intpus.as_slice().try_into().unwrap(),
  outputs.as_mut_slice().try_into().unwrap()
);

// Setting parameters
dsp.set_parameter(Parameter::Volume, 1.0);
dsp.set_parameter(Parameter::Reset, 1.0);

// Reading ui parameters
let rms = dsp.get_ui_parameter(UIParameter::Rms);
let peak = dsp.get_ui_parameter(UIParameter::Peak);

// Using DspState
let mut state = DspState::default();
dsp.update_state(&mut dsp_state);
```

Anyone programming in C?



Anyone programming in C?



Backend

- Should initiate jack
- Be the bridge between ui and audio thread
- Receive parameter changes
- Send state updates

Backend: JackBackend

- ```
  cargo add jack rtrb triple_buffer
  ```

- ```
  pub struct JackBackend {
      dsp: DSP,
      jack: jack::Client,
      ports_audio_in: Vec<Port<AudioIn>>,
      ports_audio_out: Vec<Port<AudioOut>>,
      state_writer: StateWriter,
      command_consumer: CommandConsumer,
  }
  ```

Backend: Types

- ```rust
  use rtrb::{Consumer, Producer};
  use triple_buffer::{Input, Output, TripleBuffer};
  pub type StateReader = Output<DSPState>;
  pub type StateWriter = Input<DSPState>;

  pub type Command = (Parameter, f32);
  pub type CommandProducer = Producer<Command>;
  pub type CommandConsumer = Consumer<Command>;
  ```
  ```

Backend: JackBackend::new()

```
impl JackBackend {
    pub fn new() -> (Self, StateReader, CommandProducer) {
        let dsp = DSP::new();

        let (jack, _status) =
            jack::Client::new("IFC24",
jack::ClientOptions::NO_START_SERVER).unwrap();
        let mut ports_audio_in: Vec<Port<AudioIn>> = Vec::new();
        let mut ports_audio_out: Vec<Port<AudioOut>> = Vec::new();

        ports_audio_in.push(
            jack.register_port("in_l", jack::AudioIn)
                .expect("Failed registering in_l port"),
        );
        ports_audio_in.push(
            jack.register_port("in_r", jack::AudioIn)
                .expect("Failed registering in_r port"),
        );
        ports_audio_out.push(
            jack.register_port("out_l", jack::AudioOut)
                .expect("Failed registering out_l port"),
        );
        ports_audio_out.push(
            jack.register_port("out_r", jack::AudioOut)
                .expect("Failed registering out_r port"),
        );

        let (state_writer, state_reader) =
TripleBuffer::new(&DSPState::default()).split();
        let (command_producer, command_consumer) =
rtrb::RingBuffer::new(24);

        (
            Self {
                dsp,
                jack,
                ports_audio_in,
```

```
                ports_audio_out,
                state_writer,
                command_consumer,
            },
            state_reader,
            command_producer,
        )
    }
}
```

Backend: JackBackend::run()

```
impl JackBackend {
    pub fn run(mut self) -> JackJoinHandle {
        let (shutdown_send, shutdown_recv) = sync_channel::<()>(1);
        let join_handle = std::thread::spawn(move || {
            // init dsp with a given sample rate
            let mut sample_rate = self.jack.sample_rate();
            self.dsp.set_sample_rate(sample_rate);

            // init input and output buffers
            let mut max_buffer_size = (self.jack.buffer_size() * 2) as
usize;
            let mut inputs = vec![vec![0_f32; max_buffer_size];
DSP::INPUTS];
            let mut outputs = vec![vec![0_f32; max_buffer_size];
DSP::OUTPUTS];
            info!("start jackbackend with
max_buffer_size={max_buffer_size} and sample_rate={sample_rate}");

            let mut last_ui_state_update = Instant::now()
                .checked_sub(MAX_ELAPSED_UI_WRITE * 2)
                .unwrap();

            // create jack process closure that runs for each buffer
            let process_callback = move |jack_client: &jack::Client,
                                         ps: &jack::ProcessScope|
                  -> jack::Control {
                let mut state_changed = false;

                let len: usize = ps
                    .n_frames()
                    .try_into()
                    .expect("can't cast jack n_frames to usize");

                // Handle jack requesting a bigger buffer as we can handle
                if len > max_buffer_size {
                    error!(
                            "jack wants {} samples but our buffer can only
```

```
hold {}. resizing buffers. this might cause xruns.",
                            len, max_buffer_size
                        );
                    max_buffer_size = len * 2;
                    inputs = vec![vec![0_f32; max_buffer_size];
DSP::INPUTS];
                    outputs = vec![vec![0_f32; max_buffer_size];
DSP::OUTPUTS];
                }

                let jack_sample_rate = jack_client.sample_rate();
                // Handle jack changing the sample rate
                if jack_sample_rate != sample_rate {
                    error!("jack requested sample_rate change to
{jack_sample_rate}");
                    self.dsp.set_sample_rate(jack_sample_rate);
                    sample_rate = jack_sample_rate;
                }

                // copy audio input for all ports from jack to the faust
input buffer
                inputs
                    .iter_mut()
                    .enumerate()
                    .for_each(|(index_port, input)| {
                        let port_audio_in =
self.ports_audio_in[index_port].as_slice(ps);
                        input[0..len].copy_from_slice(port_audio_in);
                    });

                // Apply all commands the backend sent us
                while let Ok(command) = self.command_consumer.pop() {
                    self.dsp.set_parameter(command.0, command.1);
                    state_changed = true;
                }

                // Flush denormals and then run the faust dsp
                no_denormals(|| {
                    self.dsp.compute(
                        len,
                        inputs
                            .as_slice()
                            .try_into()
                            .expect("Invalid amount of inputs channels"),
                        outputs
```

```
                                .as_mut_slice()
                                .try_into()
                                .expect("Invalid amount of output channels"),
                        );
                    });

                    // copy audio output for all ports from faust to the jack
output
                    outputs
                        .iter_mut()
                        .enumerate()
                        .for_each(|(index_port, output)| {
                            let port =
self.ports_audio_out[index_port].as_mut_slice(ps);
                            port.copy_from_slice(&output[0..len]);
                        });

                    // Update state if needed
                    if state_changed || last_ui_state_update.elapsed() >
MAX_ELAPSED_UI_WRITE {
                        let dsp_state = self.state_writer.input_buffer();
                        self.dsp.update_state(dsp_state);
                        self.state_writer.publish();
                        last_ui_state_update = Instant::now();
                    }

                    jack::Control::Continue
                };

                // init jack process handler.
                let process =
jack::ClosureProcessHandler::new(process_callback);

                // activate the jack, which starts the processing.
                let active_client = jack::AsyncClient::new(self.jack, (),
process).unwrap();

                shutdown_recv
                    .recv()
                    .expect("error on shutdown_recv channel");

                active_client
                    .deactivate()
                    .expect("failed deactivating jack client");
```

```
            debug!("stopped jack backend & thread");
        });

        JackJoinHandle {
            join_handle: Some(join_handle),
            shutdown_send,
        }
    }
}
```

Backend: Backend

```rust
use dsp::Parameter;
use tracing::error;

use crate::{
    jack::{CommandProducer, JackJoinHandle, StateReader},
    JackBackend,
};

pub struct Backend {
    jack_join_handle: Option<JackJoinHandle>,
    command_producer: CommandProducer,
}

impl Backend {
    pub fn new() -> (Self, StateReader) {
        let (jack_backend, state_reader, mut command_producer) =
JackBackend::new();
        let jack_join_handle = jack_backend.run();
        (
            Self {
                jack_join_handle: Some(jack_join_handle),
                command_producer,
            },
            state_reader,
        )
    }

    pub fn command(&mut self, parameter: Parameter, value: f32) {
        if let Err(err) = self.command_producer.push((parameter, value)) {
            error!("Failed sending command to backend because of
err={err:?}");
        }
    }
}

impl Drop for Backend {
    fn drop(&mut self) {
```

```
            if let Some(jack_join_handle) = self.jack_join_handle.take() {
                jack_join_handle.shutdown();
            }
        }
}
```

Check-In


My kidnappers returning me after listening to me talk about rust for two hours

Gui: app.rs

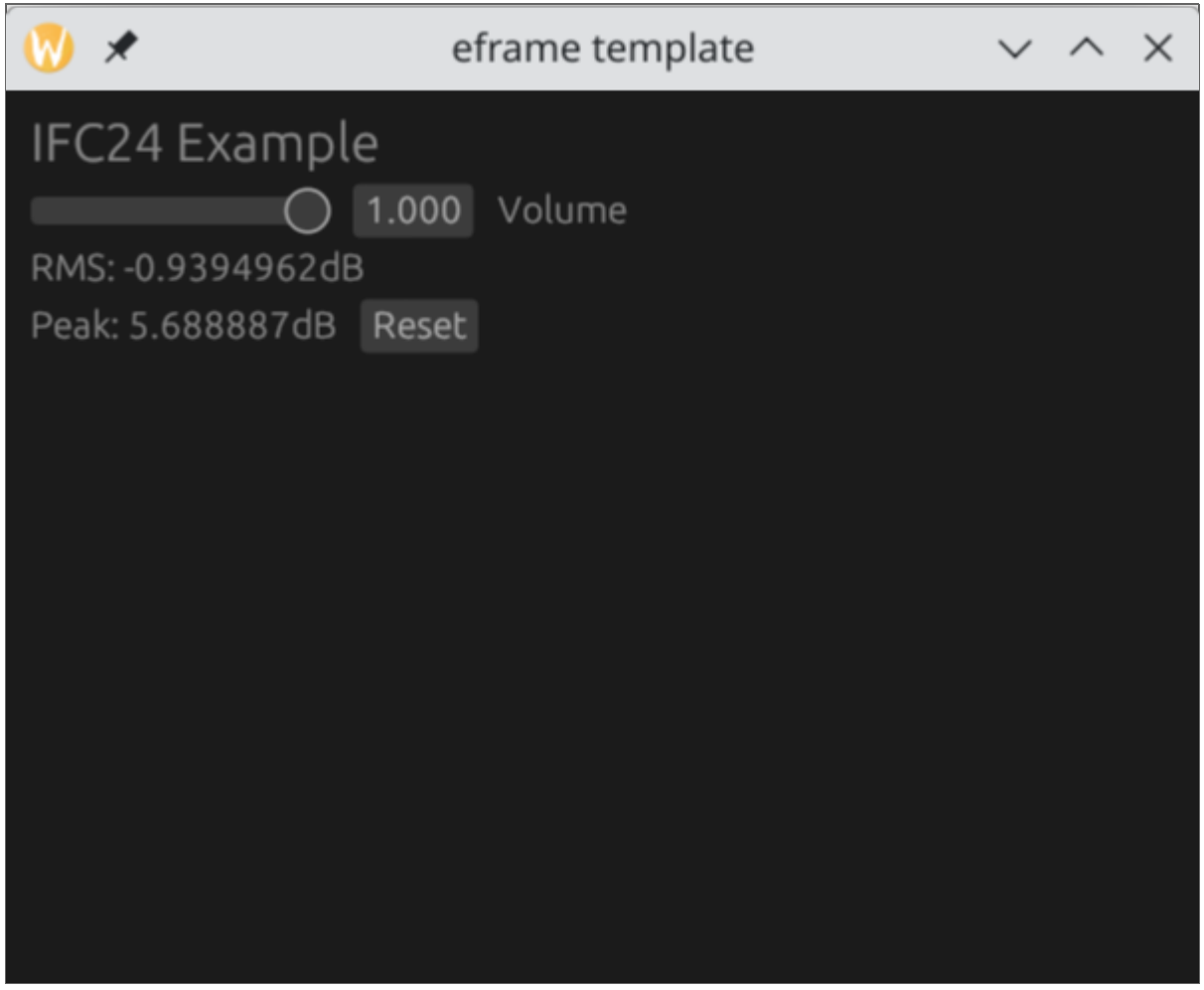
```rust
use std::time::Duration;

use backend::{Backend, Parameter, StateReader};
use eframe::egui::{self, Button, Sense};

const REFRESH_RATE_HZ: f32 = 30.0;

pub struct App {
    backend: Backend,
    state_reader: StateReader,
}

impl App {
    /// Called once before the first frame.
    pub fn new(_cc: &eframe::CreationContext<'_>) -> Self {
        let (backend, state_reader) = Backend::new();

        Self {
            backend,
            state_reader,
        }
    }
}

impl eframe::App for App {
    /// Called each time the UI needs repainting, which may be many times
per second.
    fn update(&mut self, ctx: &egui::Context, _frame: &mut eframe::Frame)
{
        let state = self.state_reader.read();

        egui::CentralPanel::default().show(ctx, |ui| {
            // The central panel the region left after adding TopPanel's
and SidePanel's
            ui.heading("IFC24 Example");

            ui.add(
```

```
                    egui::Slider::from_get_set(0.0..=1.0, |value| {
                        if let Some(value) = value {
                            self.backend.command(Parameter::Volume, value as
f32);
                            return value;
                        }
                        state.p_volume as f64
                    })
                    .text("Volume"),
                );
                ui.add(egui::Label::new(format!("RMS: {}dB", state.rms)));
                ui.horizontal(|ui| {
                    ui.add(egui::Label::new(format!("Peak: {}dB",
state.peak)));
                    let reset_button_response =
ui.add(Button::new("Reset").sense(Sense::drag()));
                    if reset_button_response.drag_started() {
                        self.backend.command(Parameter::Reset, 1.0);
                    }
                    if reset_button_response.drag_released() {
                        self.backend.command(Parameter::Reset, 0.0);
                    }
                });
            });
            ctx.request_repaint_after(Duration::from_secs_f32(1.0 /
REFRESH_RATE_HZ));
        }
}
```

Gui: main.rs

```rust
mod app;

use app::App;
use eframe::egui;

fn main() -> eframe::Result<()> {
    tracing_subscriber::fmt::init();
    let native_options = eframe::NativeOptions {
        viewport: egui::ViewportBuilder::default()
            .with_inner_size([400.0, 300.0])
            .with_min_inner_size([300.0, 220.0]),
        ..Default::default()
    };
    eframe::run_native(
        "eframe template",
        native_options,
        Box::new(|cc| Box::new(App::new(cc))),
    )
}
```

Tadaaaa!


eframe template
IFC24 Example
1.000
Volume
RMS: -0.9394962dB
Peak: 5.688887dB
Reset

Recommendations

- nih_plug
- vizia
- slint
- symphonia

Speaker notes