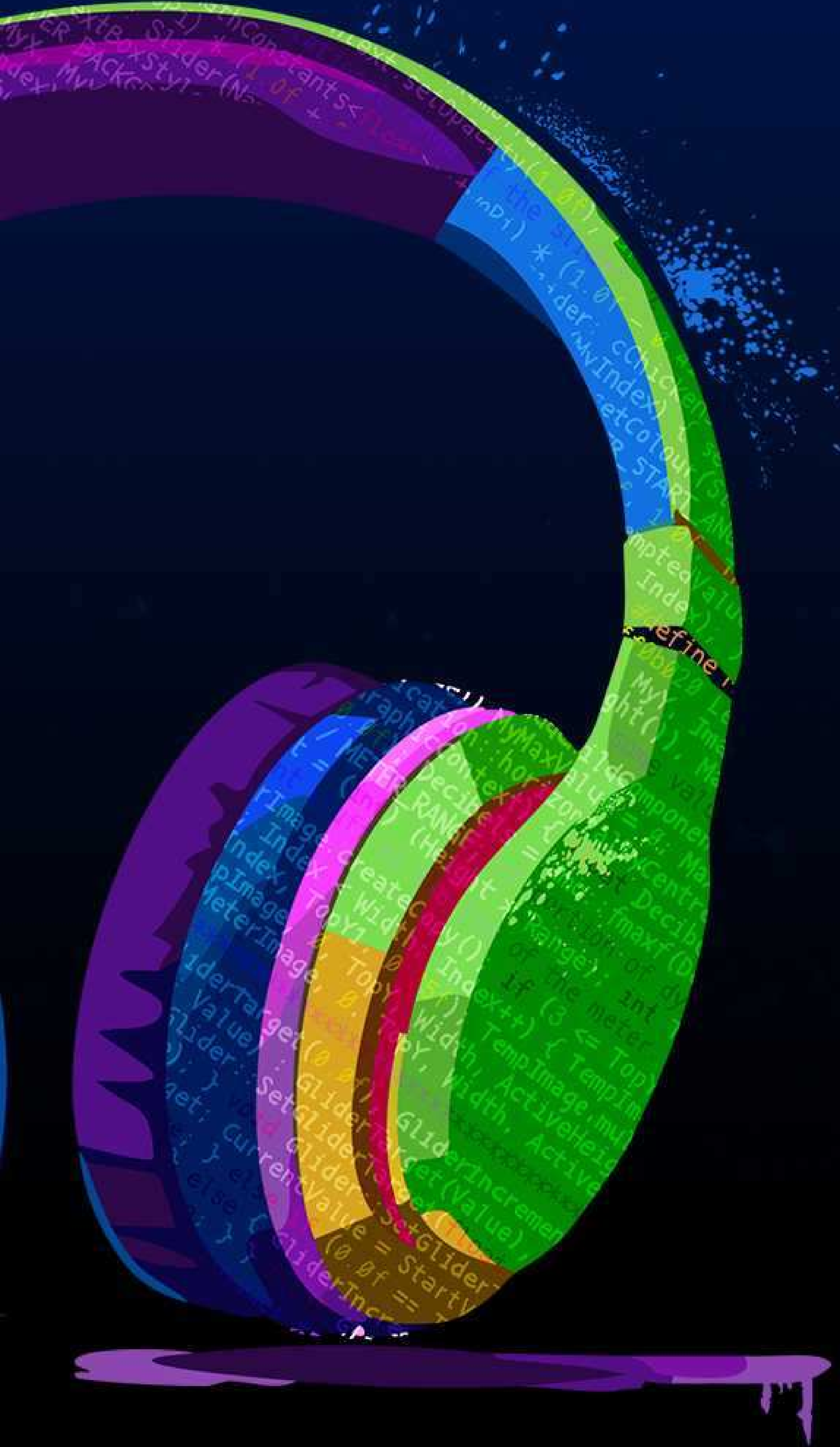


ADC24 Bristol

BUILDING AUDIO APPS WITH RUST

AN OVERVIEW OF TOOLS AND TECHNIQUES

STEPHAN ECKES

About Me

Audio Communication and Technology M.Sc.

- Technical University Berlin

Software Development (Working Student)

- Neumann
- u-he
- Holoplot

About Me

ai|coustics (ai-coustics.com)

- real-time speech enhancement
- responsible for SDK and inference engine
- demo apps and devices

neodsp (github.com/neodsp)

- open source libraries
- ***youtube.com/@neodsp*** → audio, coding & Linux videos

Why Rust?

The Rust ecosystem empowers me, as a solo developer, to create apps that work right from the start, rarely fail, and compile across all platforms.

Slides

Grab your copy

https://github.com/neodsp/adc-talk-24

Audio IO

Platform Specific APIs

- highest level of control
- harder to use
- need to integrate for each platform yourself
- bindings may not be feature complete

ALSA
github.com/diwic/alsa-rs

JACK
github.com/RustAudio/rust-jack

PulseAudio
github.com/jnqnfe/pulse-binding-rust

PipeWire ⭐ official
gitlab.freedesktop.org/pipewire/pipewire-rs

CoreAudio
github.com/RustAudio/coreaudio-rs

WASAPI
github.com/HEnquist/wasapi-rs

ASIO
github.com/RustAudio/cpal

Oboe
github.com/katyo/oboe-rs

Platform Independent APIs

- only higher level functions
- easier to use
- build once - run everywhere
- not all have duplex streams
- not all wrap all system APIs

Portaudio

- duplex streams
- wrapper is stable

RtAudio

- duplex streams

cxx-juce

- duplex streams
- long build process

C-bindings

CPAL

- “pure” Rust
- web + mobile
- no duplex streams

cubeb

- Firefox audio backend
- in transition to pure Rust
- no ASIO
- no selectable frame-size

Rust

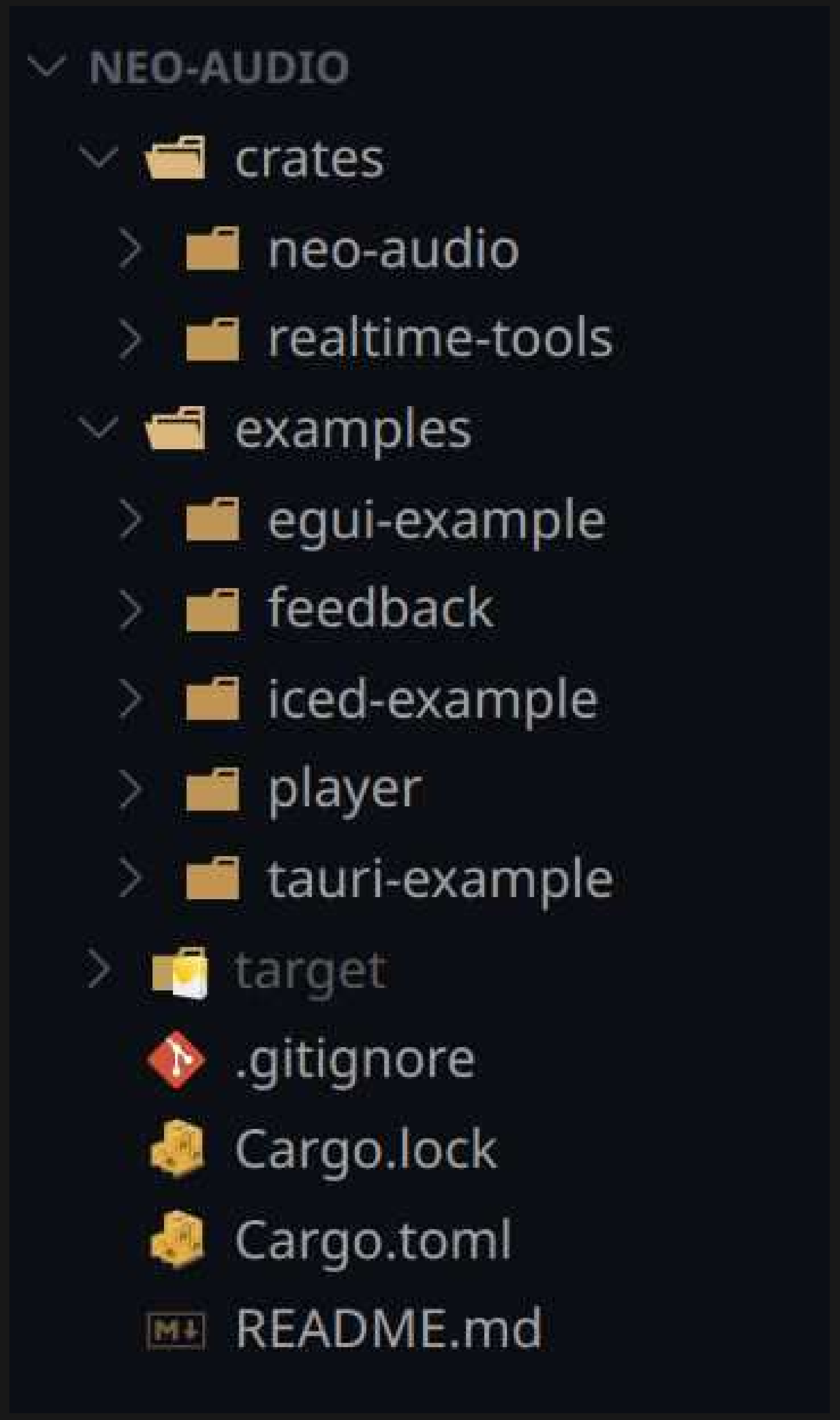
Portaudio

```rust
let settings = pa::DuplexStreamSettings::new(input_params, output_params, SAMPLE_RATE, FRAMES);

let callback = move |pa::DuplexStreamCallbackArgs {
                        in_buffer,
                        out_buffer,
                        ..
                    }| {
    out_buffer.copy_from_slice(in_buffer);
    pa::Continue
};

let mut stream = pa.open_non_blocking_stream(settings, callback)?;
stream.start()?;
stream.stop()?;
```

neo-audio

github.com/neodsp/neo-audio

- my backend-agnostic audio framework
- still early (don't blindly use it in production)
- use it for inspiration
- shows integration into three UI frameworks



```
NEO-AUDIO
  crates
    neo-audio
    realtime-tools
  examples
    egui-example
    feedback
    iced-example
    player
    tauri-example
  target
  .gitignore
  Cargo.lock
  Cargo.toml
  README.md
```

neo-audio

```rust
let mut neo_audio = NeoAudio::<PortAudioBackend>::new()?;
```

```rust
let sender = neo_audio.start_audio(MyProcessor::default())?;
```

```rust
sender.send(MyMessage::Gain(0.5))?;
```

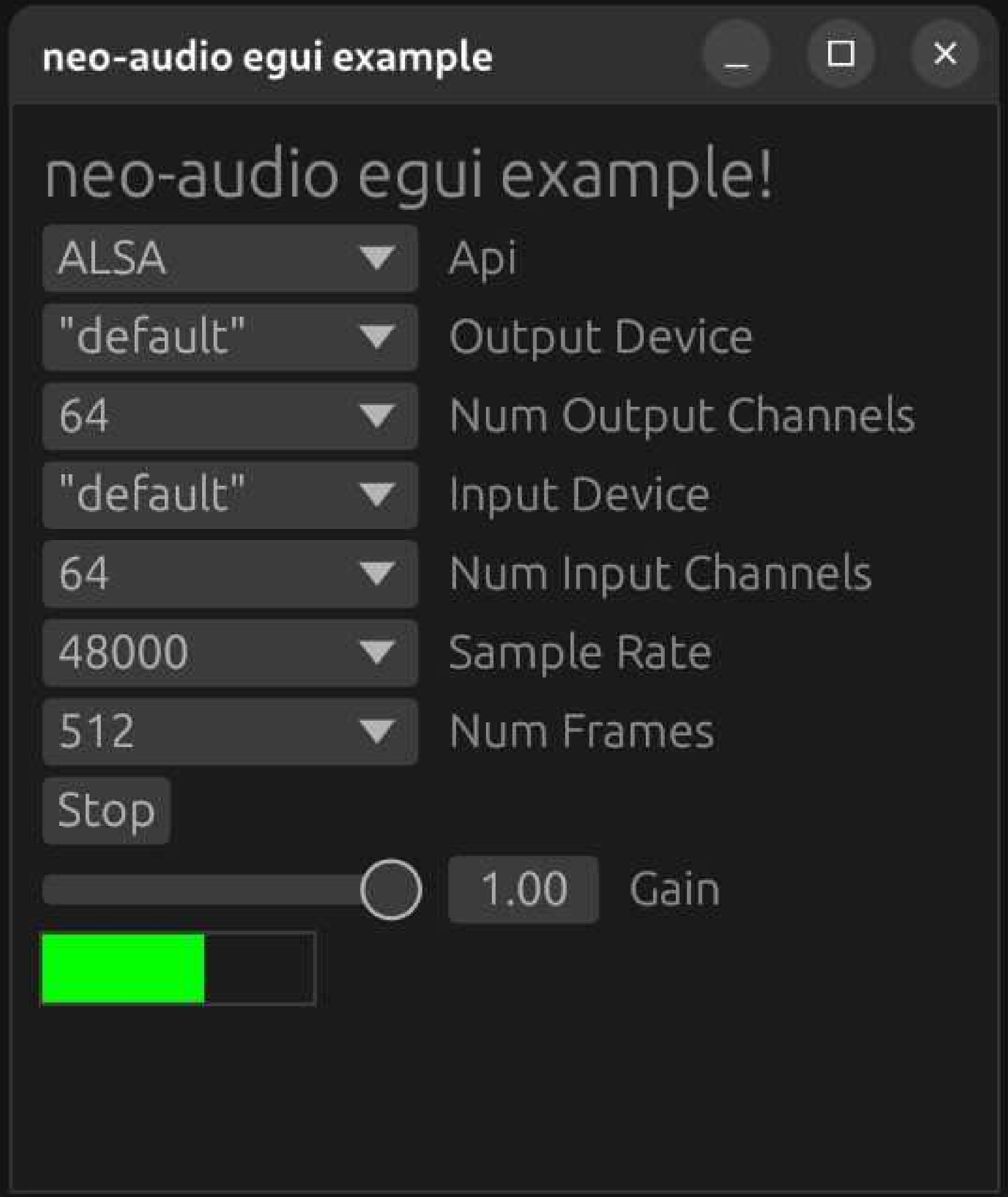
neo-audio

```
let output_devices = neo_audio.backend().available_output_devices();
```

neo-audio

```rust
neo_audio
    .backend_mut()
    .set_output_device(Device::Name("My Soundcard Name".into()))?;

let selected_output_device = neo_audio.backend().output_device();
```

neo-audio

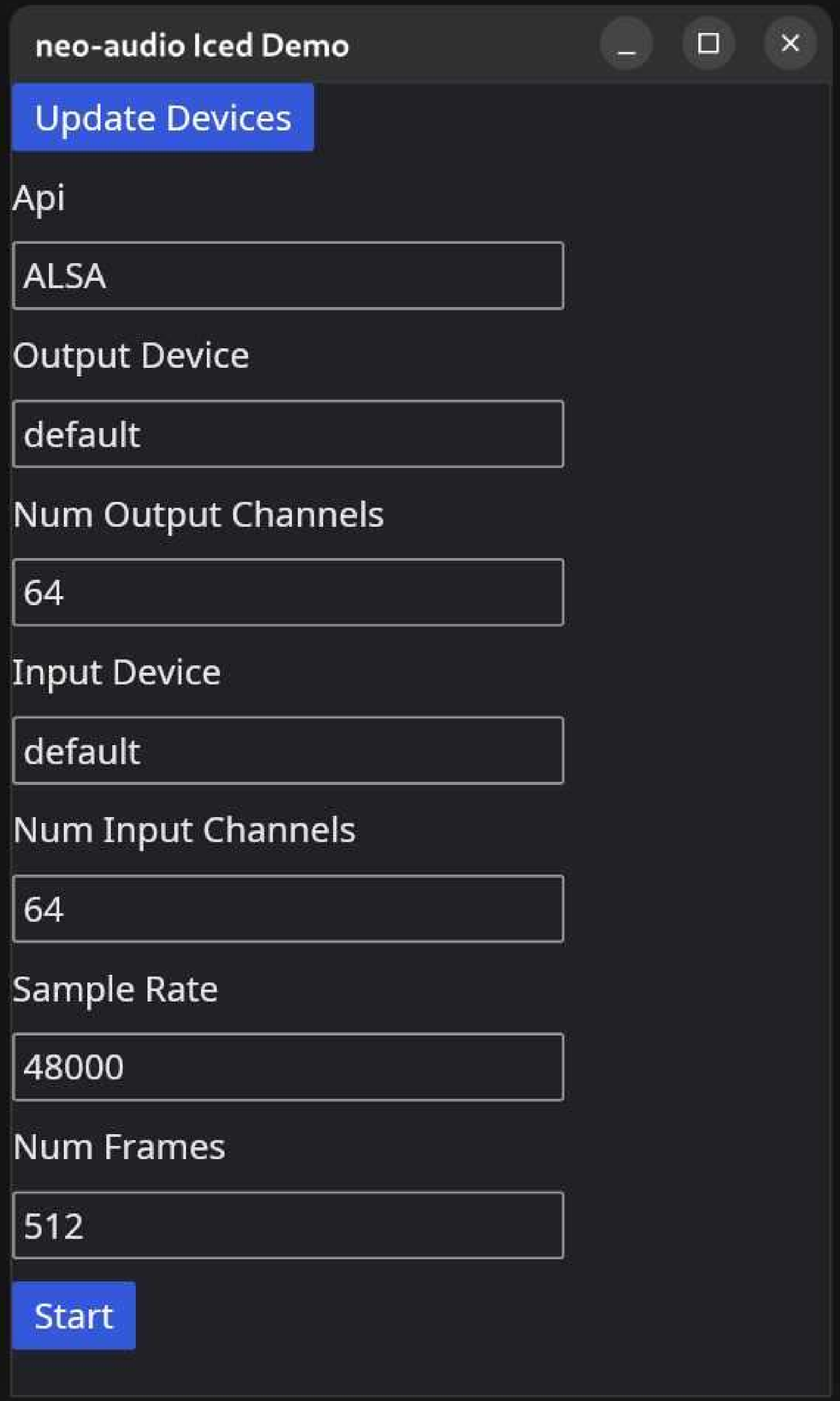

neo-audio egui example

neo-audio egui example!

ALSA	Api
"default"	Output Device
64	Num Output Channels
"default"	Input Device
64	Num Input Channels
48000	Sample Rate
512	Num Frames

Stop

1.00 Gain


neo-audio Iced Demo

Update Devices

Api
ALSA

Output Device
default

Num Output Channels
64

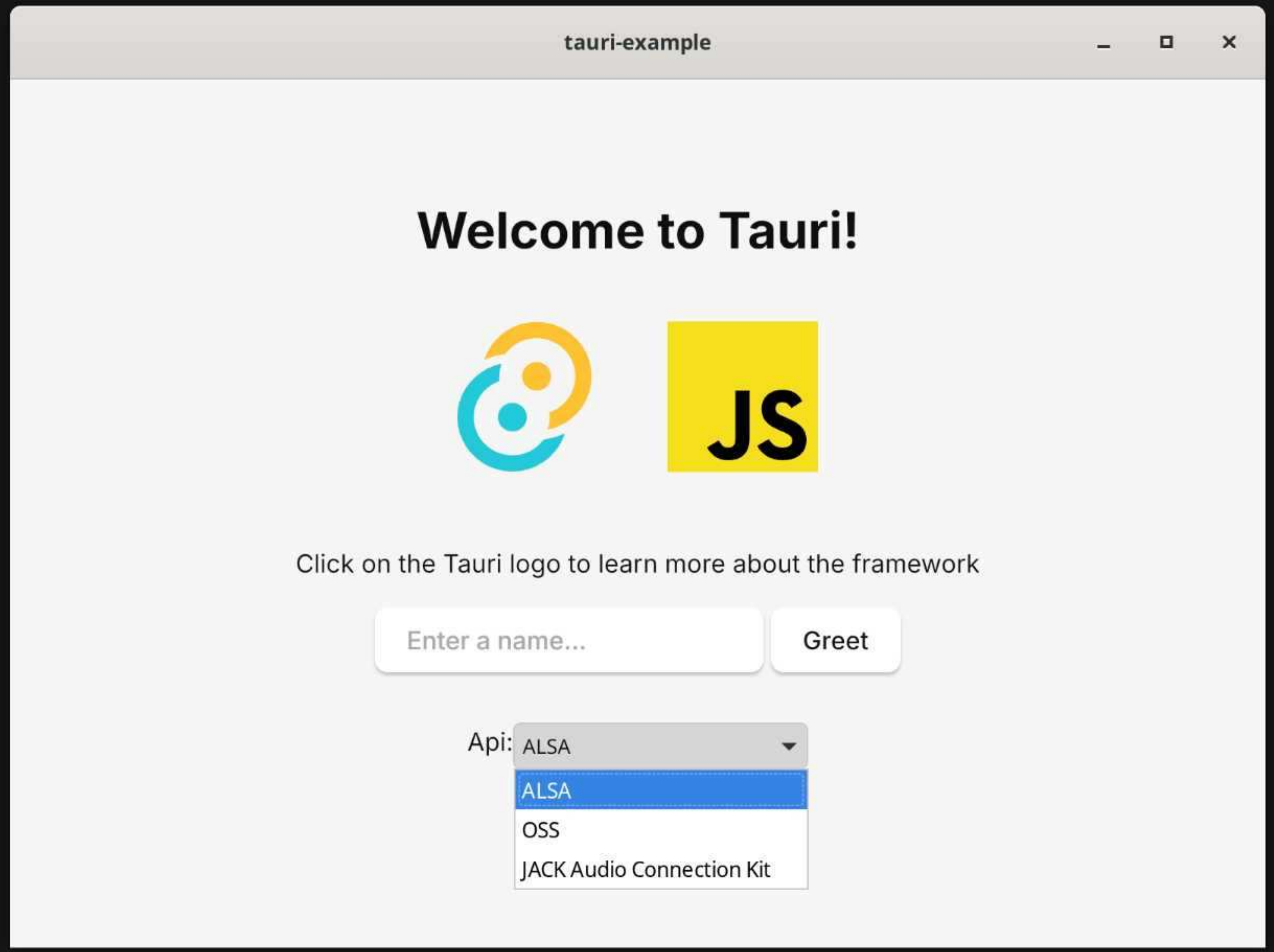
Input Device
default

Num Input Channels
64

Sample Rate
48000

Num Frames
512

Start


tauri-example

Welcome to Tauri!

Click on the Tauri logo to learn more about the framework

Enter a name... Greet

Api: ALSA

ALSA
OSS
JACK Audio Connection Kit

Thread Synchronization

one-directional

Channels

- easiest way to synchronize
- one directional
- non-blocking option
- limits in performance
- ***std::sync::mpsc::channel***
  - multi producer → single consumer
- ***crossbeam::channel::bounded***
  - better performance
  - pre-allocated memory
  - multi producer → multi consumer

```rust
use crossbeam_channel::bounded;

let (sender, receiver) = bounded(1024);

// send in one thread
sender.send(MyMessage::Mute).unwrap();

// receive in another thread
for message in receiver.try_iter() {
    // process all messages
}
```

Ringbuffer / FIFO

- faster than channels, minimal latency
- good for audio data (e.g. level meter)
- only single producer → single consumer
- ***ringbuf***
  - flexible with different types
    (local or shared, static or heap)
  - lock-free
- ***rtrb***
  - real-time ringbuffer
  - faster in some cases but only one type

```rust
use rtrb::RingBuffer;

let (mut prod, mut cons) = RingBuffer::new(1024);

prod.push(0).unwrap();

let value = cons.pop().unwrap();
```

triple-buffer

- only the newest update visible
- good for spectrum / oscilloscope
- only single producer → single consumer
- lock and wait-free
- pre-allocated

```rust
use triple_buffer::triple_buffer;

let (mut buf_input, mut buf_output) = triple_buffer([0.0; 2]);

buf_input.write([1.0, 2.0]);

let latest = buf_output.read();
```

shared memory

Mutex

- easiest
- automatically unlock when guard goes out of scope
- can block the thread or fail to get the resource
- ***std::sync::Mutex***
  - returns *PoisonError* if thread that holds a lock panics
- ***parking_lot::Mutex***
  - better performance
  - does not return *PoisonError*

```rust
use parking_lot::Mutex;

let data = Mutex::new(0);

// blocks the thread
let mut data = data.lock();

// does not block, but can be `None`
let mut maybe_data = data.try_lock();
```

does not need to be mutable 😮

Rich Pointers

- ***std::boxed::Box***
  - just to move an object to the heap
  - if you don’t know the size at compile time
- ***std::rc::Rc***
  - reference counted
  - can be cloned “for free” (single thread)
- ***std::sync::Arc***
  - atomically reference counted
  - can be cloned “for free” (multi thread)
  - necessary to share Atomics or Mutexes

```rust
let data = Arc::new(Mutex::new(0));

let data_clone = data.clone();

// move into another thread here
thread::spawn(move || {
  let mut value = data_clone.lock();
});

let mut value = data.lock();
```

Atomics

- harder to use
- only work for very small data
- usually used for audio parameters
- by far the best performance
- ***std::sync::atomic***
  - only for bool and integer types
- ***atomic_float***
  - atomics for f32 and f64
- ***crossbeam::atomic::AtomicCell***
  - generic over types (use AtomicCell::<T>::is_lock_free())

does not need to be mutable 😯

```rust
use atomic_float::AtomicF32;
use std::sync::atomic::Ordering;

let atomic = AtomicF32::new(0.0);

// in one thread
atomic.store(1.0; Ordering::Release);

// in another thread
let value = atomic.load(Ordering::Aquire);
```

Orderings

- must be provided which each load/store operation
- behave similar to C++
- tells the compiler how many optimizations it is allowed to do by re-ordering the code
- wrong ordering can lead to bugs

```
#[non_exhaustive]
pub enum Ordering {
    Relaxed,
    Release,
    Acquire,
    AcqRel,
    SeqCst,
}
```

Orderings

- ***Relaxed*** gives the most freedom to the compiler
- ***Release*** is for storing a value with some guarantees
- ***Acquire*** is for loading a value with some guarantees
- ***AcqRel*** is for operations that load and store at the same time
- ***SeqCst*** has the strongest guarantees (default in C++)

When atomics can fail

```rust
let num = Arc::new(AtomicUsize::new(0));

// spin up two threads that add one
let ths: Vec<_> = (0..2)
    .map(|_| {
        let num = num.clone();
        thread::spawn(move || {
            let curr = num.load(Acquire);
            num.store(curr + 1, Release);
        })
    })
    .collect();

// wait for both threads to be finished
for th in ths {
    th.join().unwrap();
}

// this assertion can fail!
assert_eq!(2, num.load(Relaxed));
```

```rust
thread::spawn(move || {
    let r1 = y.load(Ordering::Relaxed); // A
    x.store(r1, Ordering::Relaxed); // B
});
thread::spawn(move || {
    let r2 = x.load(Ordering::Relaxed); // C
    y.store(42, Ordering::Relaxed); // D
});
```

It is possible that r1 == r2 == 42 !

Resources for atomics

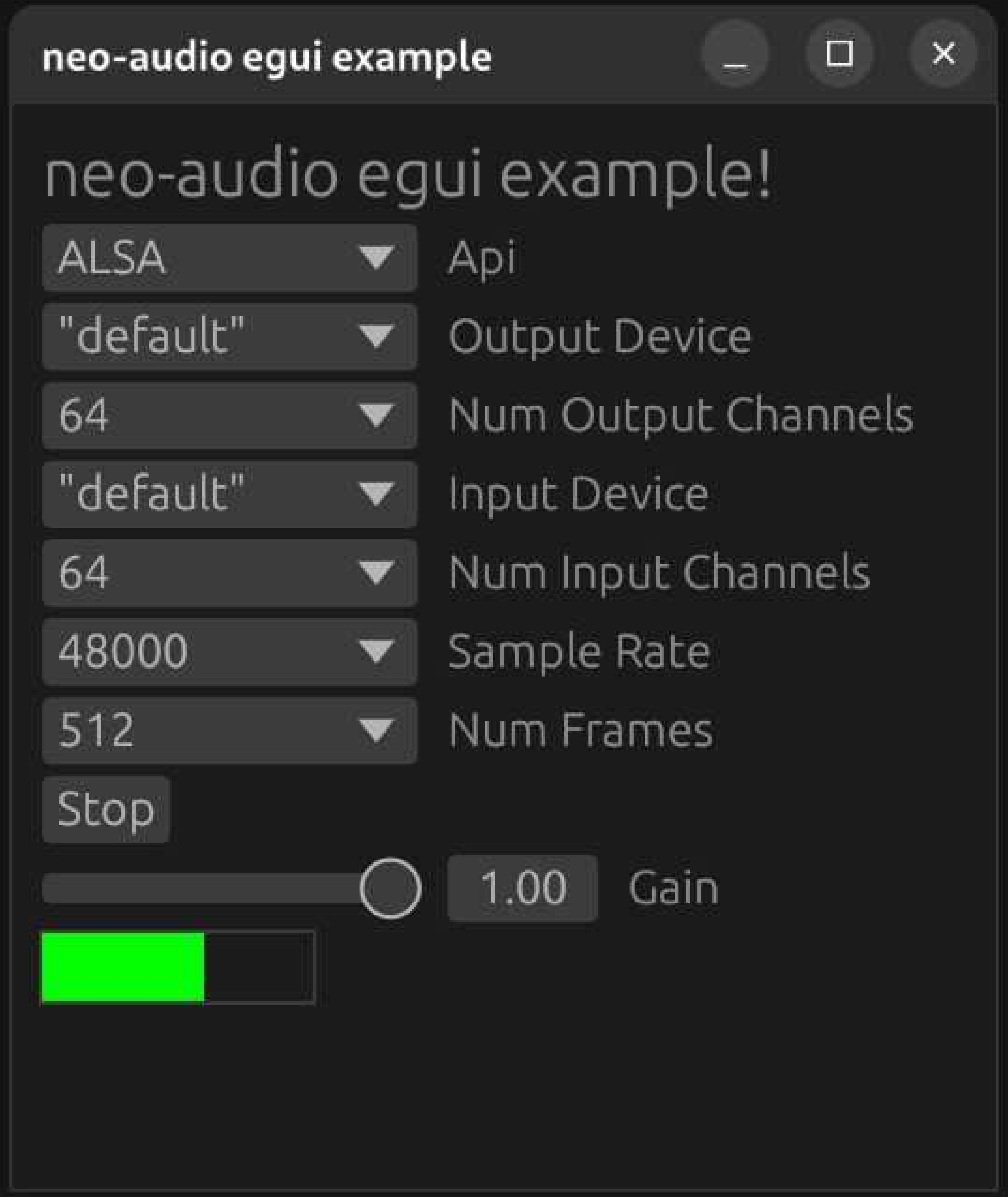
Tokio::Loom Documentation

Crust of Rust: Atomics and Memory Ordering → Jon Gjengset

https://www.youtube.com/watch?v=rMGWeSjctlY

Graphical User Interface

neo-audio

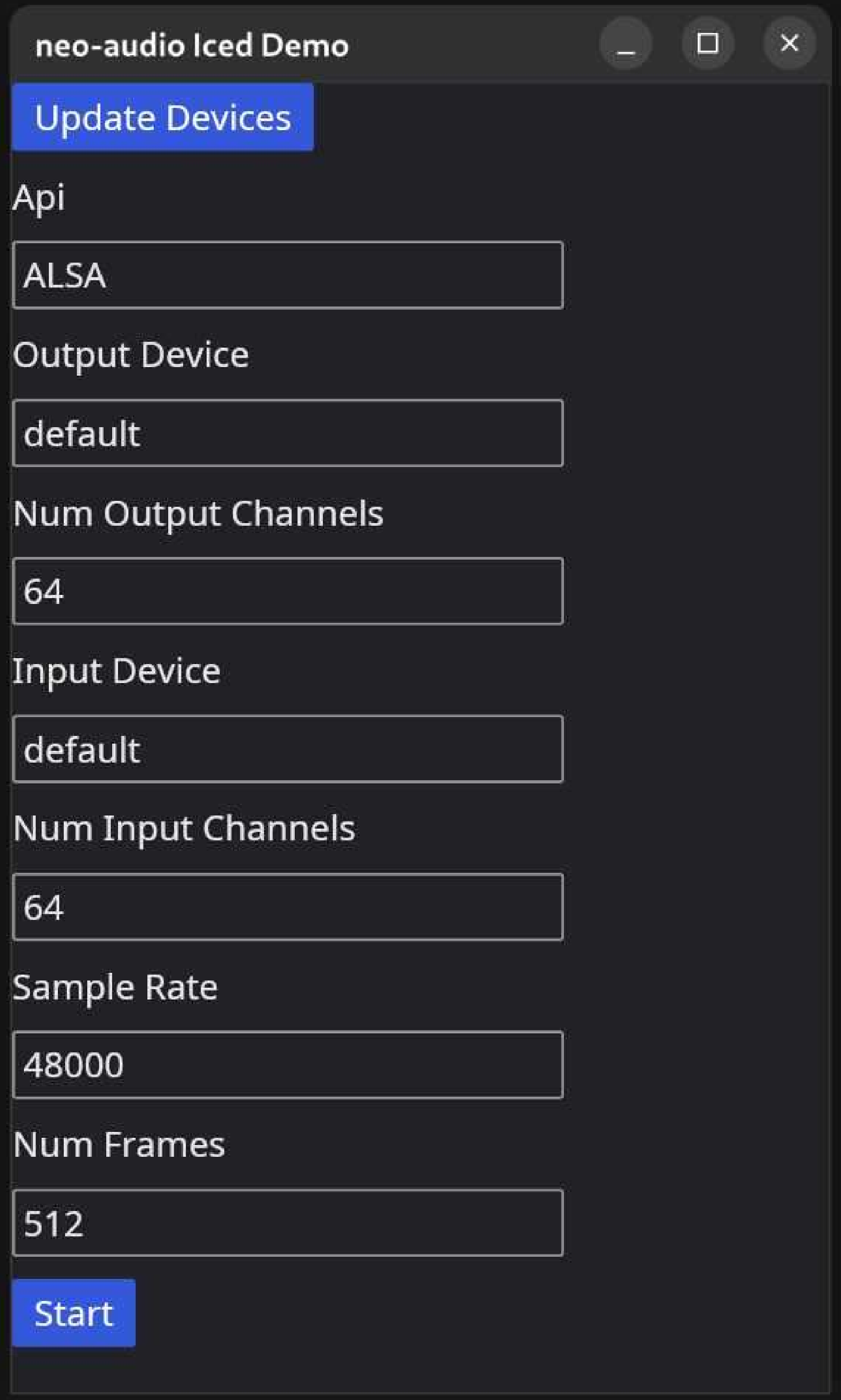


neo-audio egui example

neo-audio egui example!

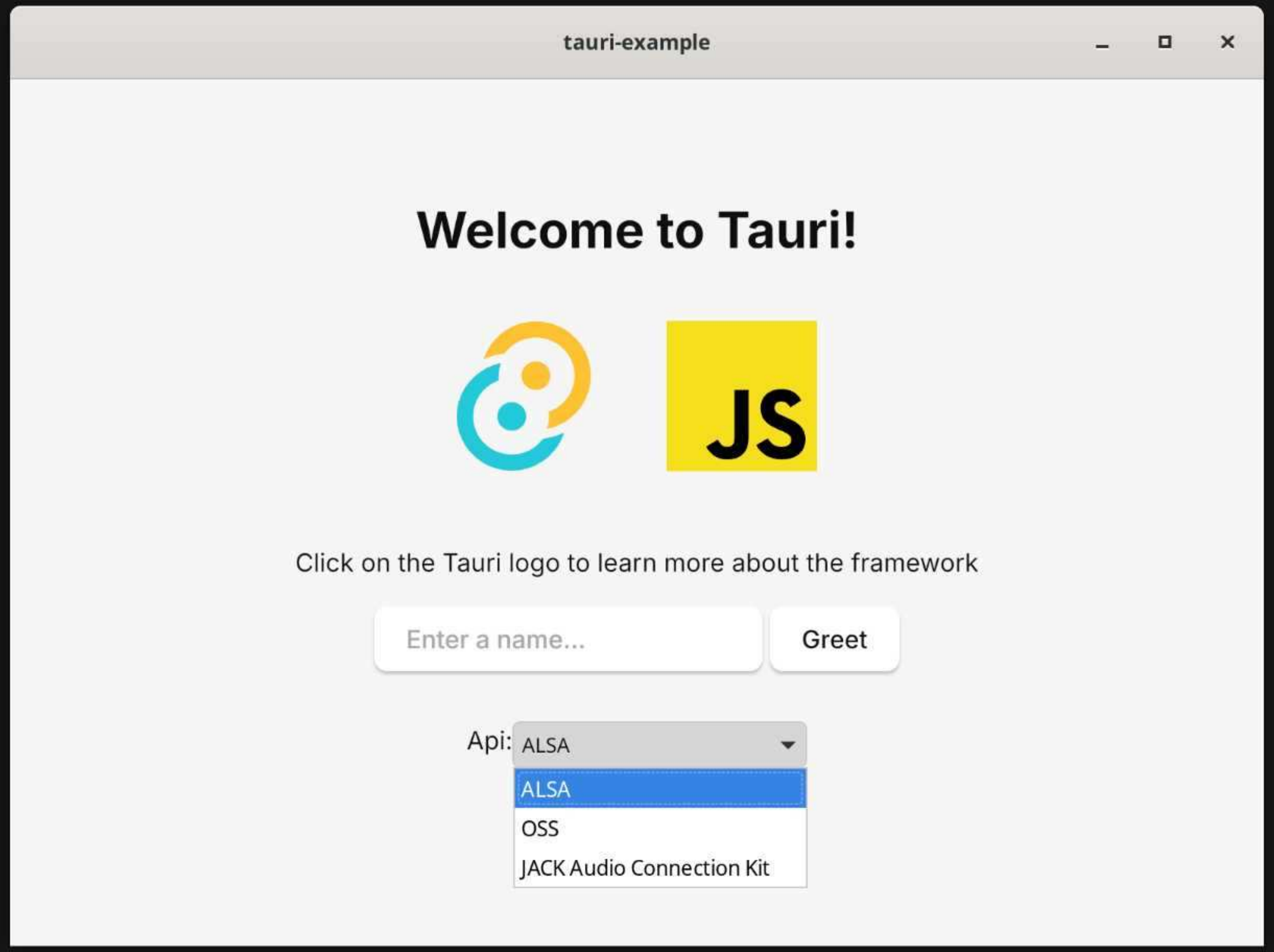
ALSA	Api
"default"	Output Device
64	Num Output Channels
"default"	Input Device
64	Num Input Channels
48000	Sample Rate
512	Num Frames

1.00 Gain



neo-audio Iced Demo

Api	ALSA
Output Device	default
Num Output Channels	64
Input Device	default
Num Input Channels	64
Sample Rate	48000
Num Frames	512



tauri-example

Welcome to Tauri!

Click on the Tauri logo to learn more about the framework

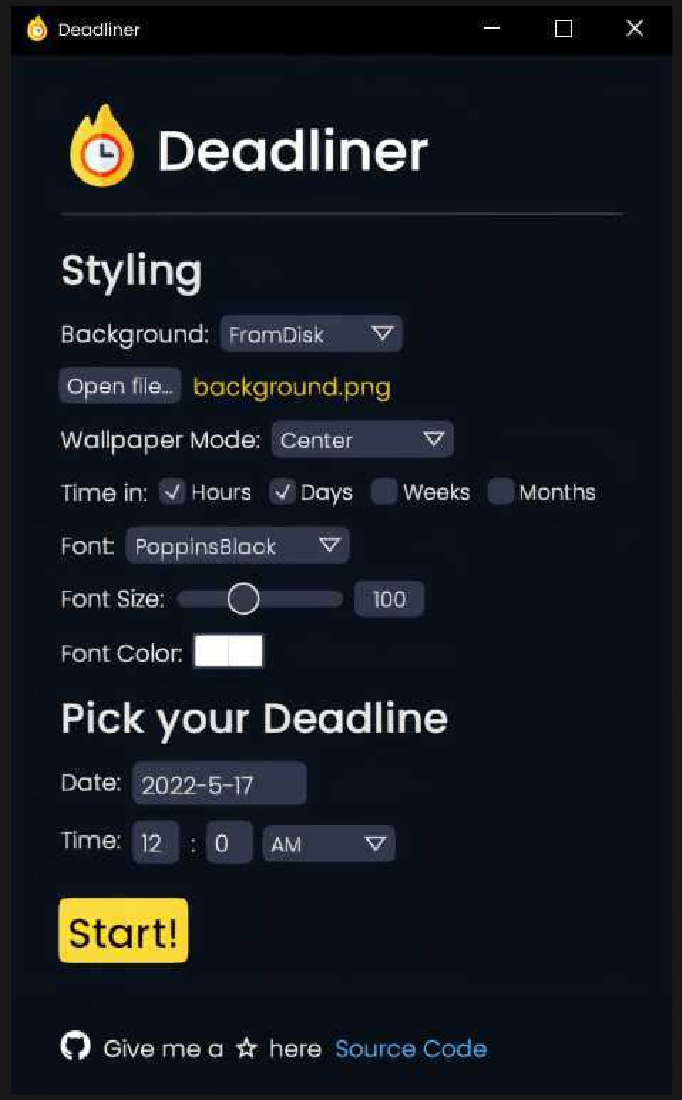
Api: ALSA

We will talk about my two favorite UI libraries, that changed the way how I look at UIs

Native User Interface

egui

- backed by rerun.io
- 22k stars, 453 contributors
- MIT or Apache-2.0 license



Deadliner

Deadliner

Styling

Background: FromDisk
Open file... background.png
Wallpaper Mode: Center
Time in: ✓ Hours ✓ Days ☐ Weeks ☐ Months
Font: PoppinsBlack
Font Size: 100
Font Color:

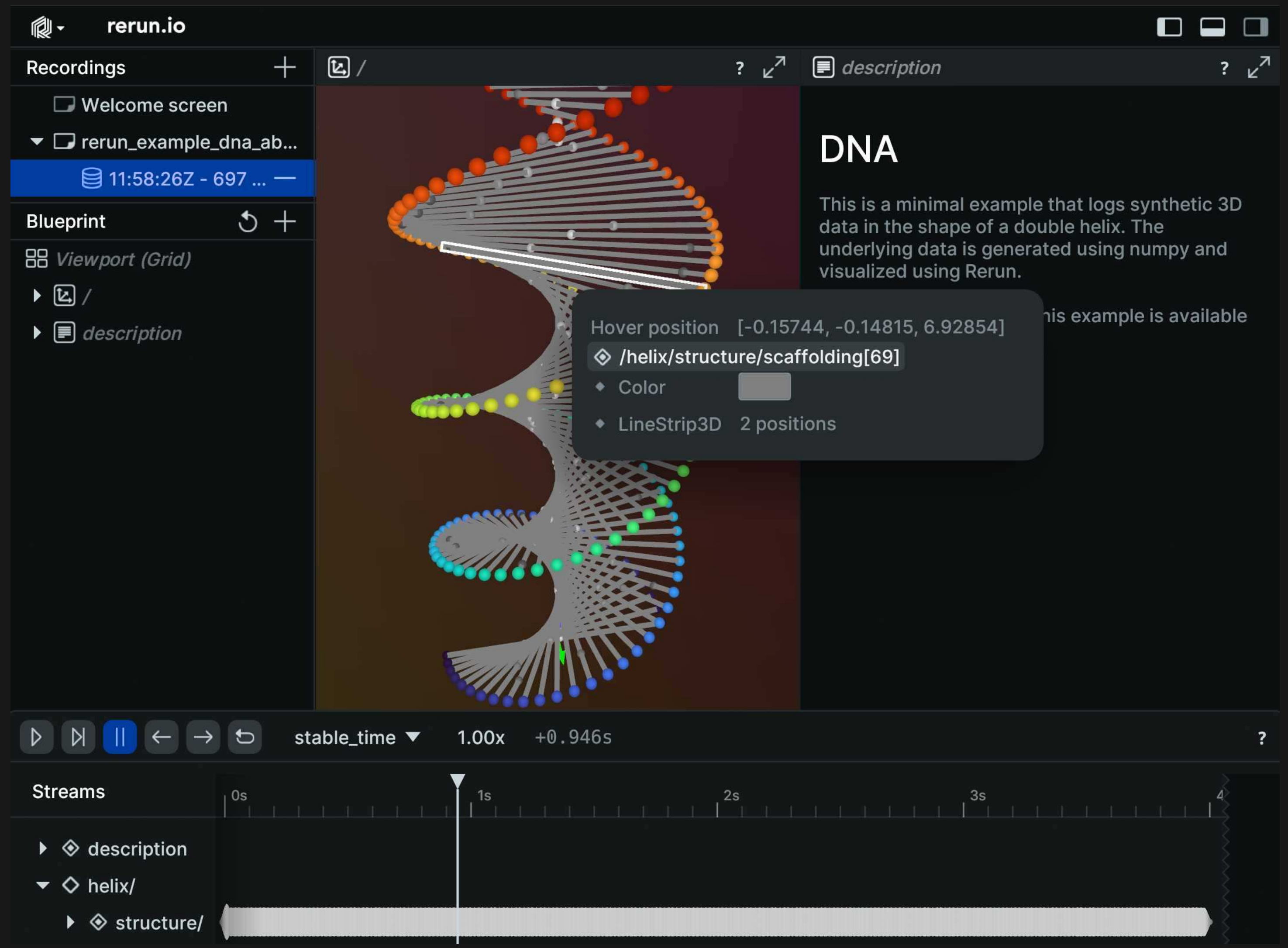
Pick your Deadline

Date: 2022-5-17
Time: 12 : 0 AM

Start!

Give me a ☆ here Source Code

github.com/deadliner-app/Deadliner



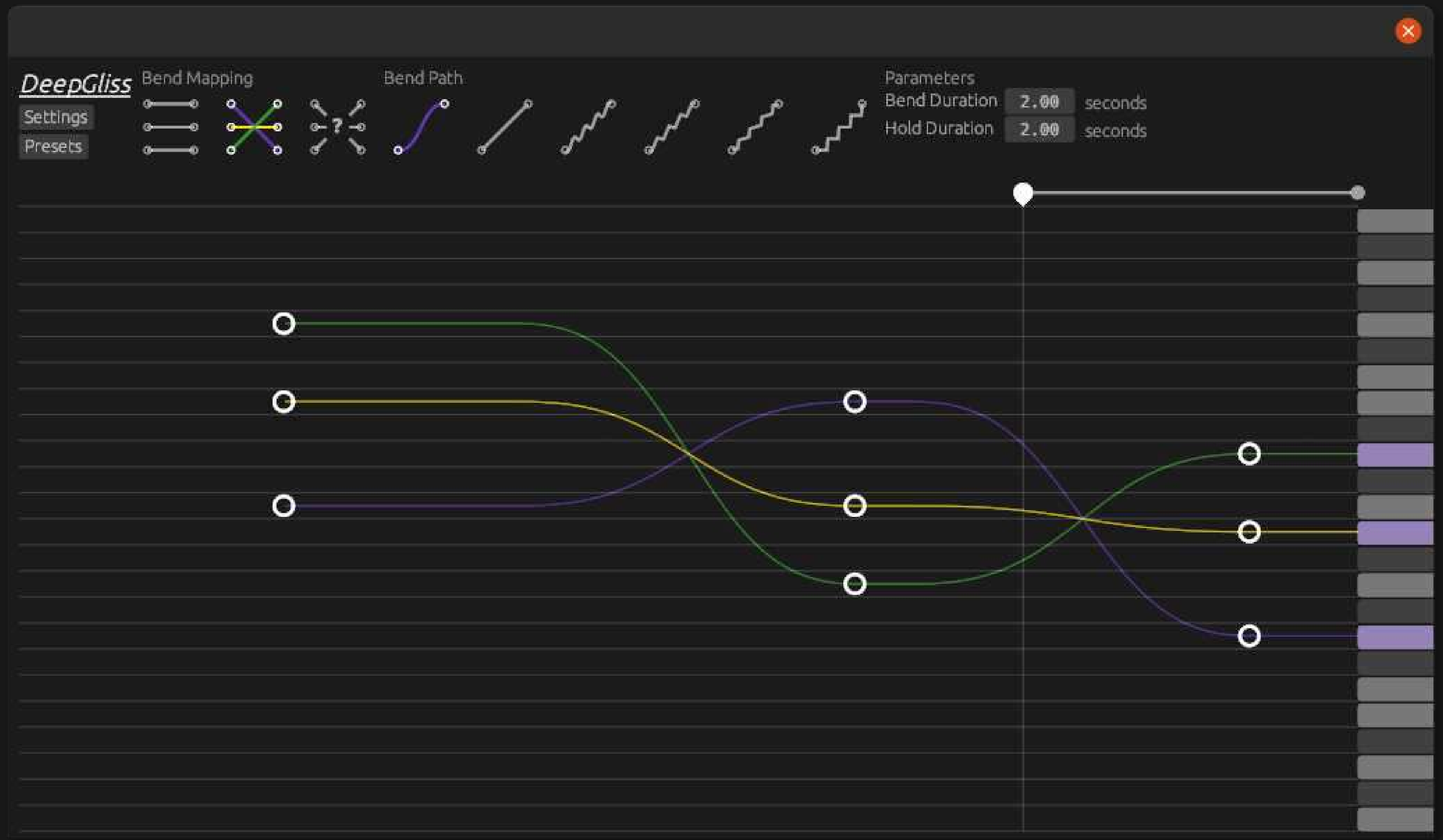
DNA

This is a minimal example that logs synthetic 3D data in the shape of a double helix. The underlying data is generated using numpy and visualized using Rerun.

...is example is available

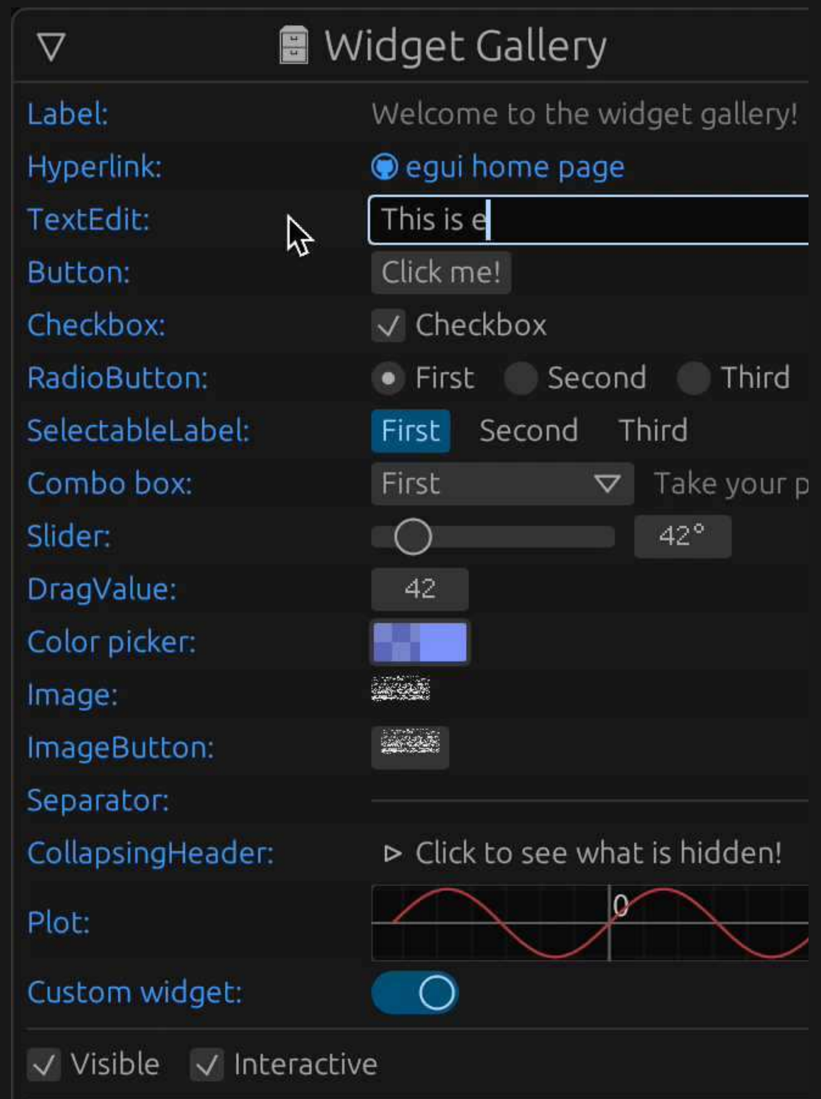
Hover position [-0.15744, -0.14815, 6.92854]
/helix/structure/scaffolding[69]
Color
LineStrip3D 2 positions

rerun.io demo



Parameters
Bend Duration 2.00 seconds
Hold Duration 2.00 seconds

VST2: github.com/JoshuaPostel/DeepGliss

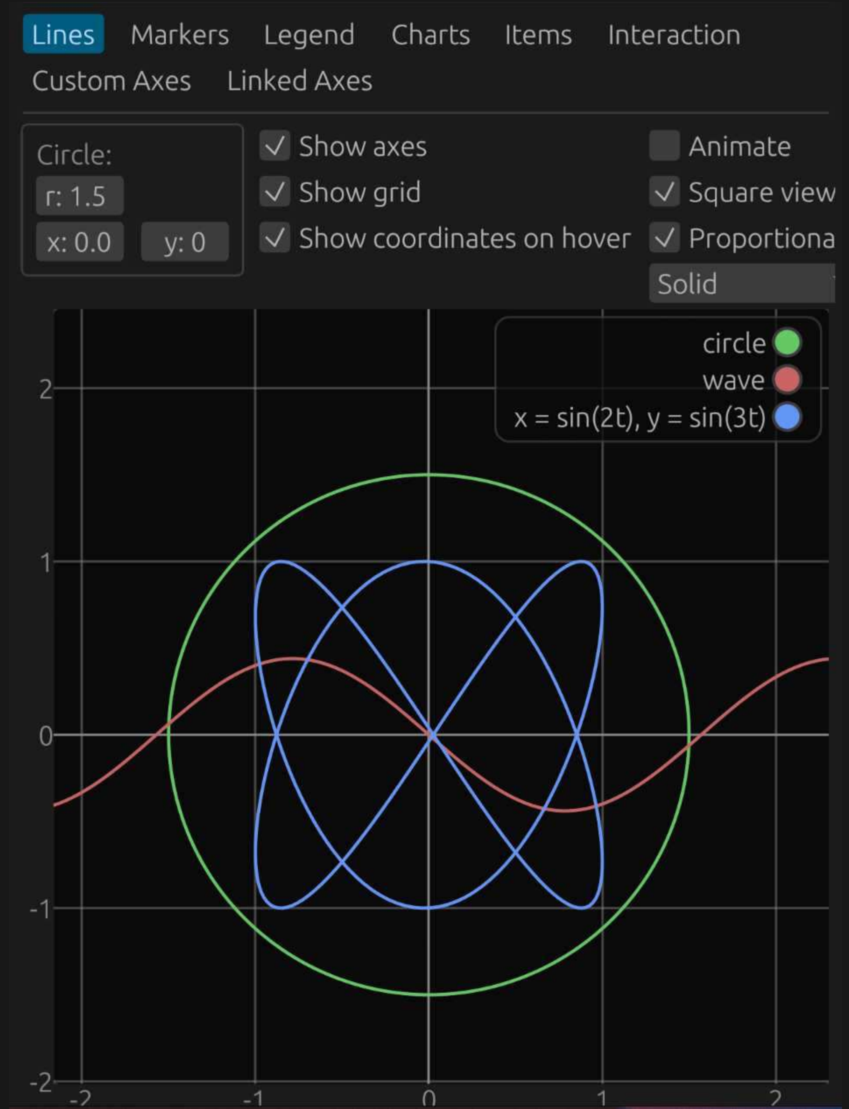


Widget Gallery

Label:	Welcome to the widget gallery!
Hyperlink:	egui home page
TextEdit:	This is e
Button:	Click me!
Checkbox:	✓ Checkbox
RadioButton:	● First ○ Second ○ Third
SelectableLabel:	First Second Third
Combo box:	First Take your p
Slider:	42°
DragValue:	42
Color picker:	
Image:	
ImageButton:	
Separator:	
CollapsingHeader:	▷ Click to see what is hidden!
Plot:	
Custom widget:	

✓ Visible ✓ Interactive

egui.rs



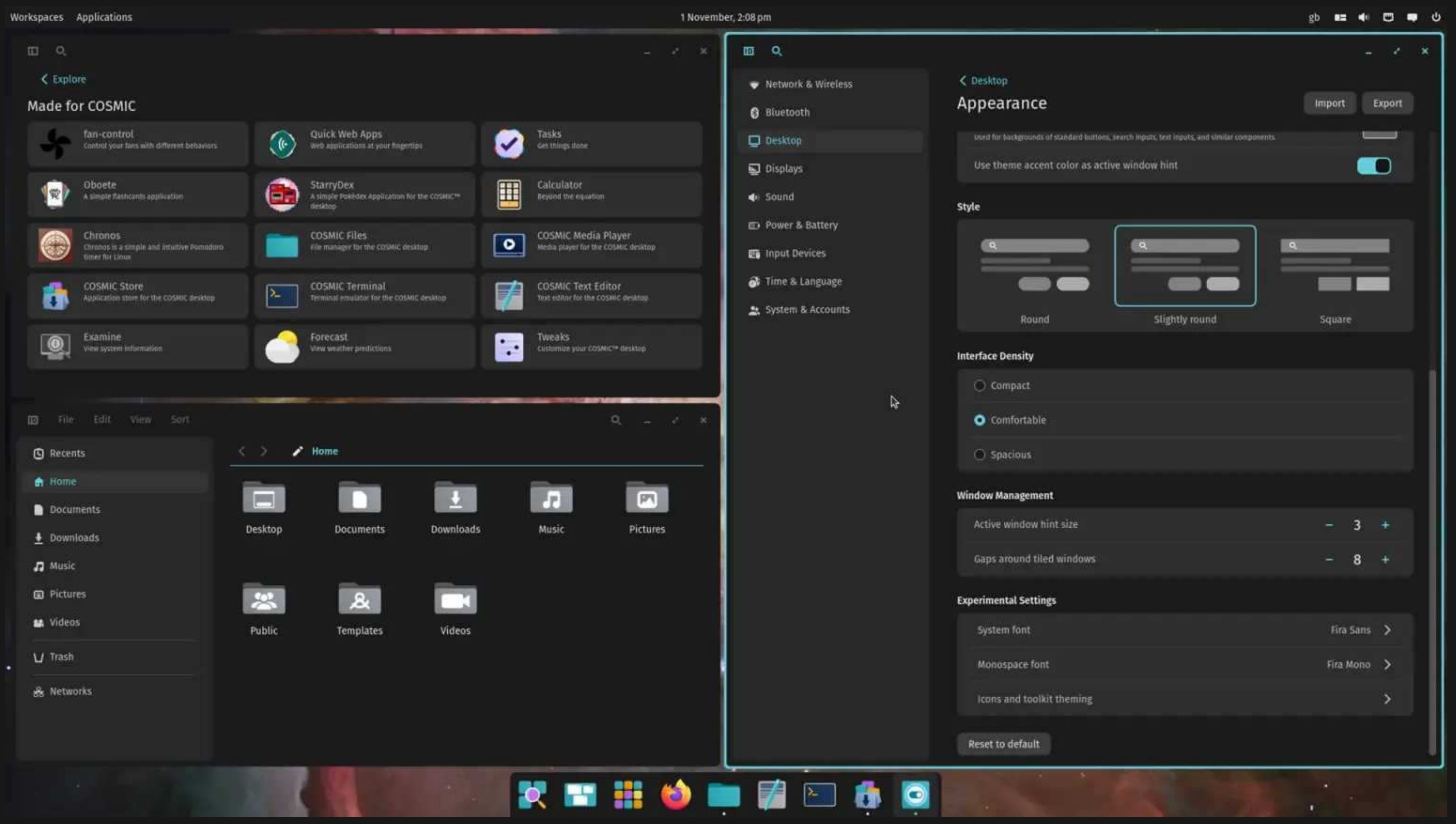
Circle:
r: 1.5
x: 0.0 y: 0
✓ Show axes
✓ Show grid
✓ Show coordinates on hover
☐ Animate
✓ Square view
✓ Proportiona
Solid

emilk.github.io/egui_plot

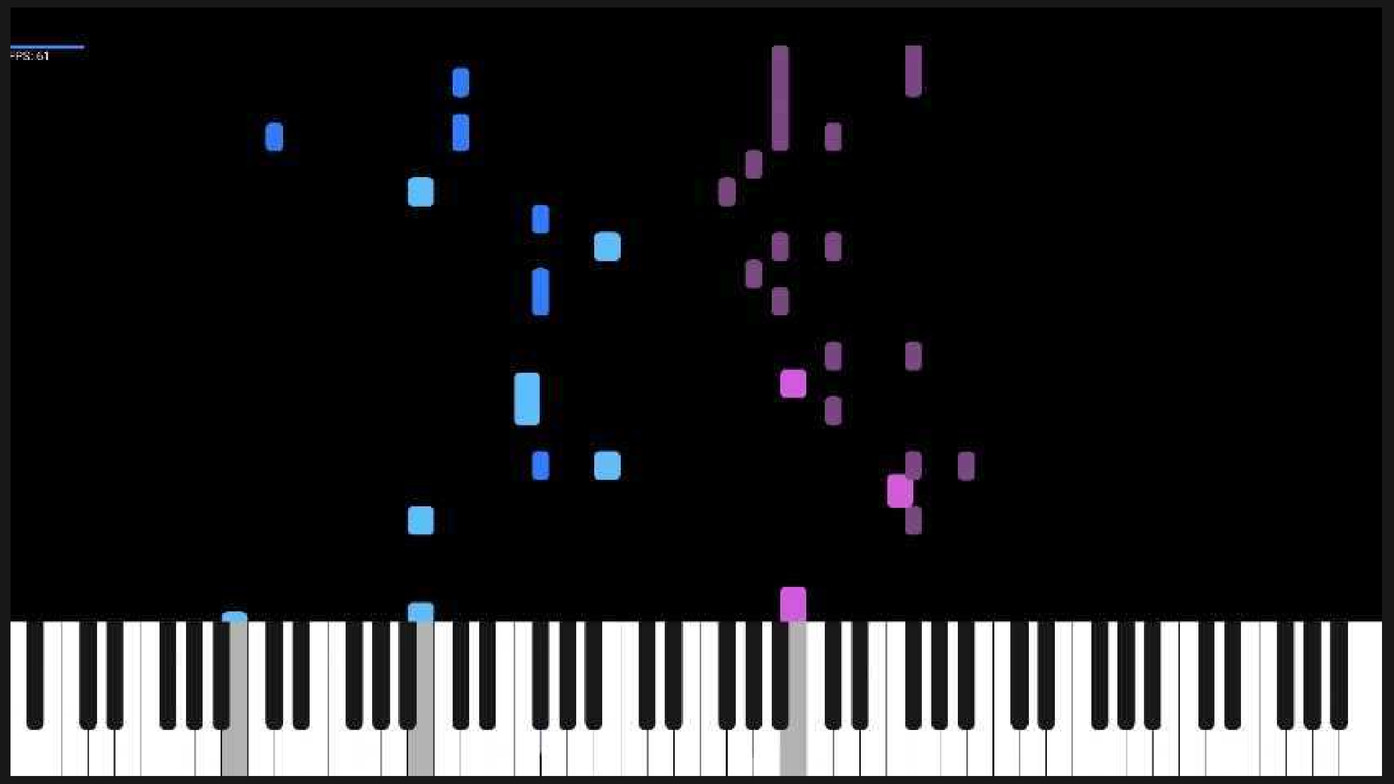
Check the demo!

iced

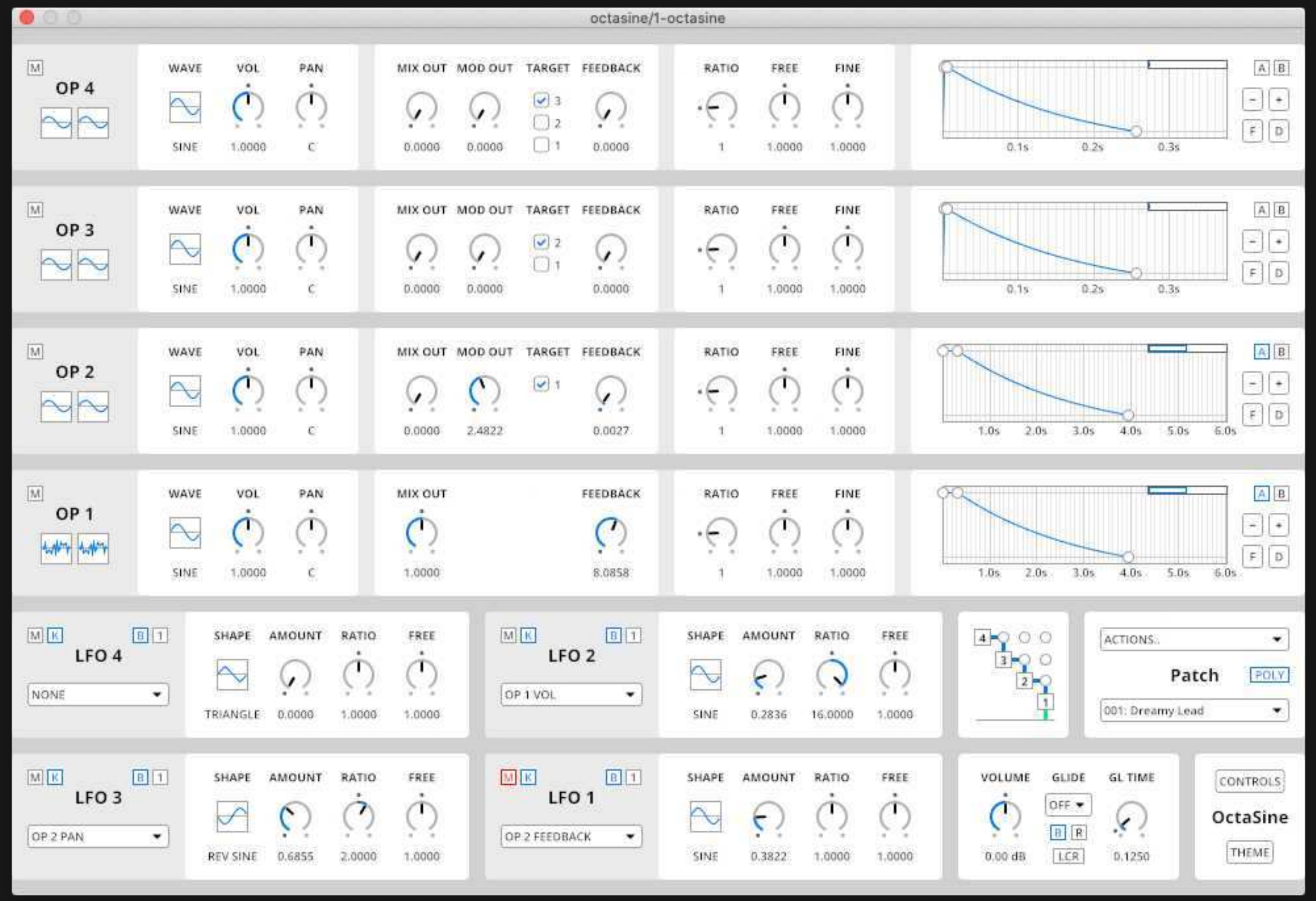
- backed by system76
  → new Pop!OS DE written in iced
- 24k stars, 248 contributors
- MIT licensed



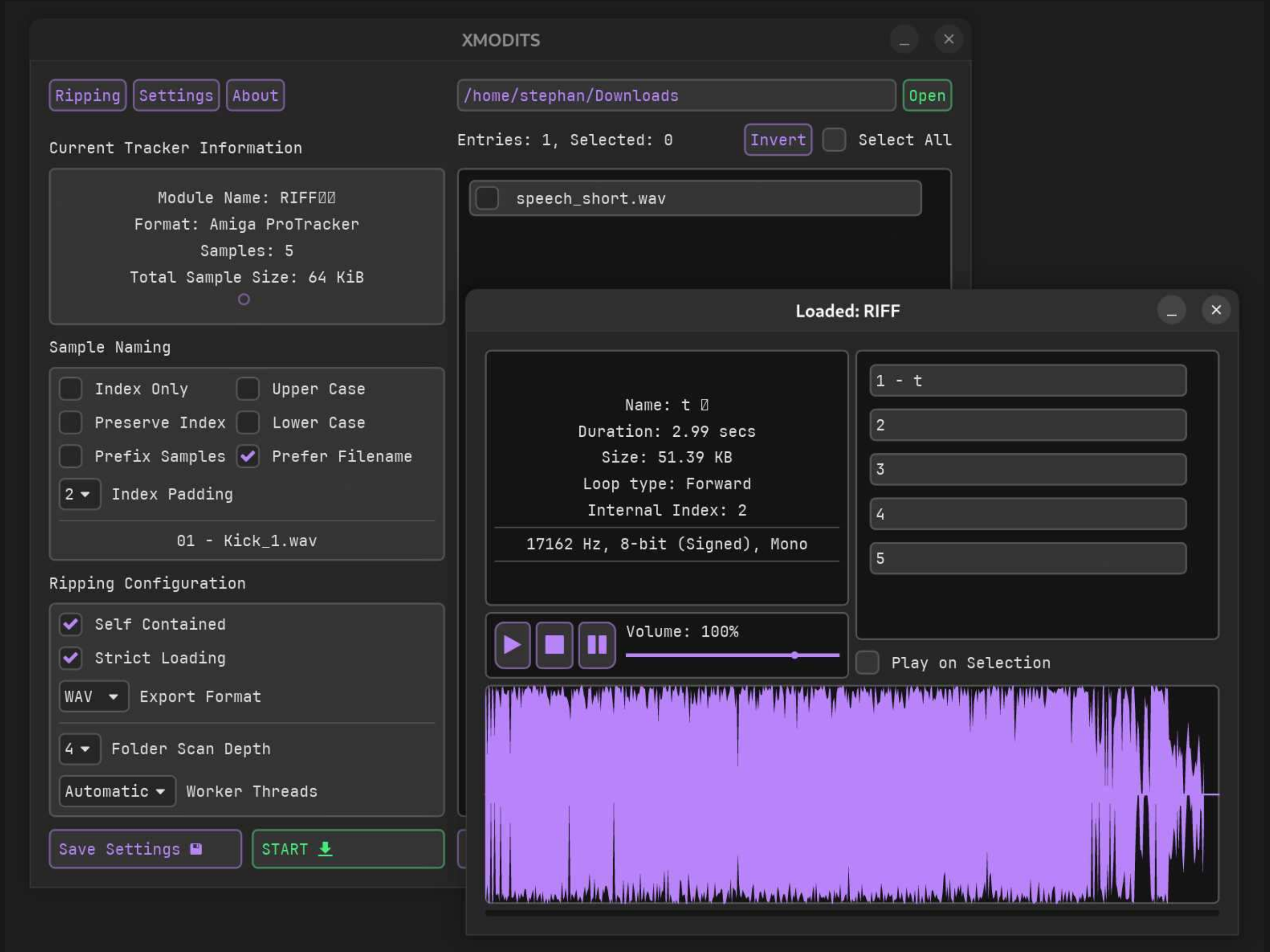
system76.com/cosmic



github.com/PolyMeilex/Neothesia/



github.com/greatest-ape/OctaSine



github.com/B0ney/xmodits

egui

- immediate mode
- inspired by Dear ImGui
- easier to use - less powerful

```rust
fn update(&mut self, ui: &mut Ui) {
    if button("Increment").clicked() {
        self.value += 1;
    }
}
```

iced

- retained mode
- inspired by Elm architecture
- harder to use - more powerful

```rust
fn update(&mut self, message: Message) {
    match message {
        Message::Increment => {
            self.value += 1;
        }
    }
}

fn view(&self) -> Element<Message> {
    button("Increment")
      .on_press(Message::Increment)
      .into()
}
```

Compatibility

both frameworks work for

- Windows
- macOS
- Linux
- Web
- Audio Plug-Ins (with nih-plug)

Resources

iced

- amazing iced tutorial https://github.com/fogarecious/iced_tutorial
- official examples https://github.com/iced-rs/iced/tree/latest/examples
- app showcase on their website https://iced.rs/

egui

- feature demo https://www.egui.rs/
- plot demo https://emilk.github.io/egui_plot/
- official examples https://github.com/emilk/egui/tree/master/examples
- app showcase https://github.com/emilk/egui/issues/996

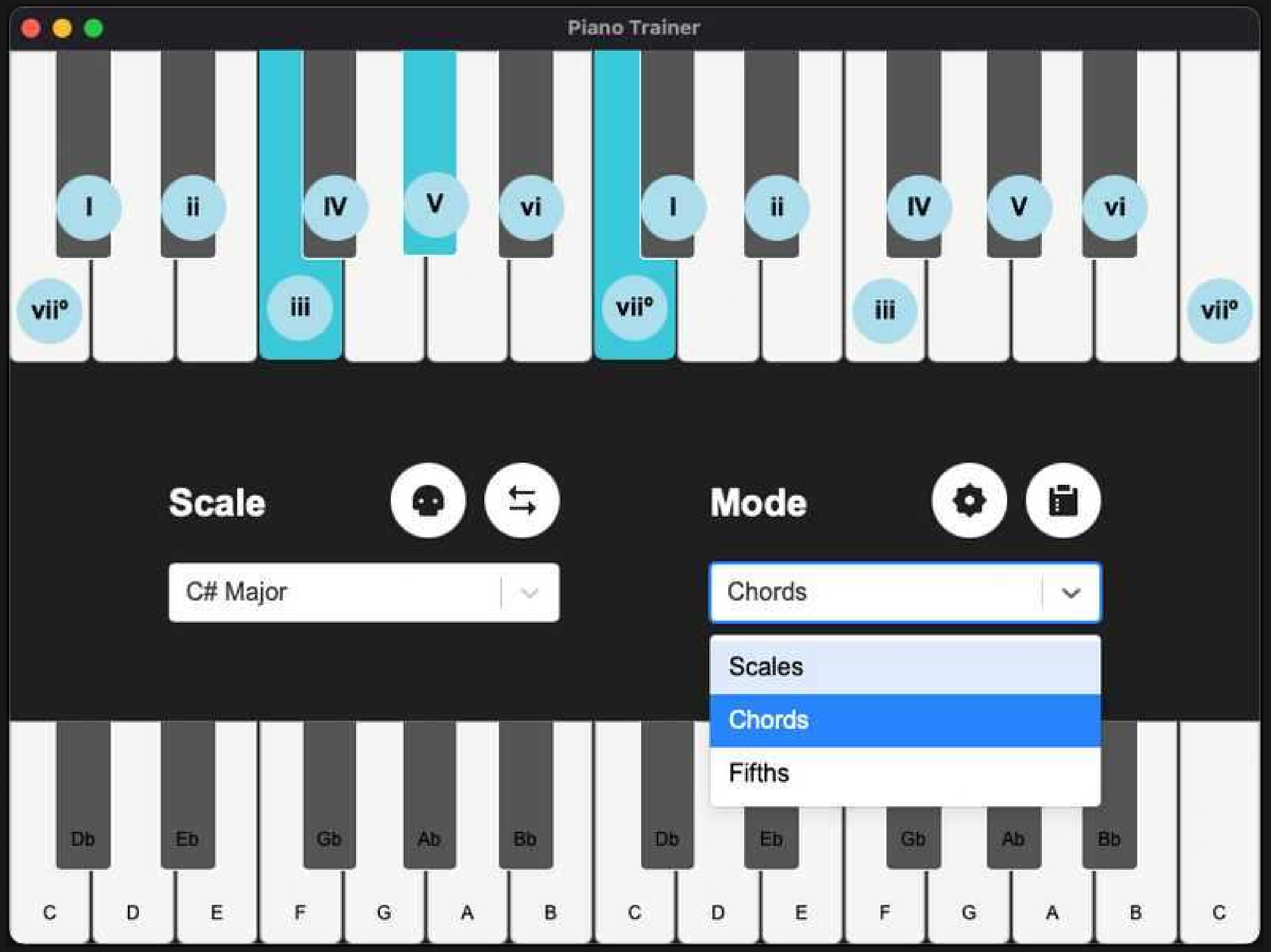
Honorable Mention:
Slint

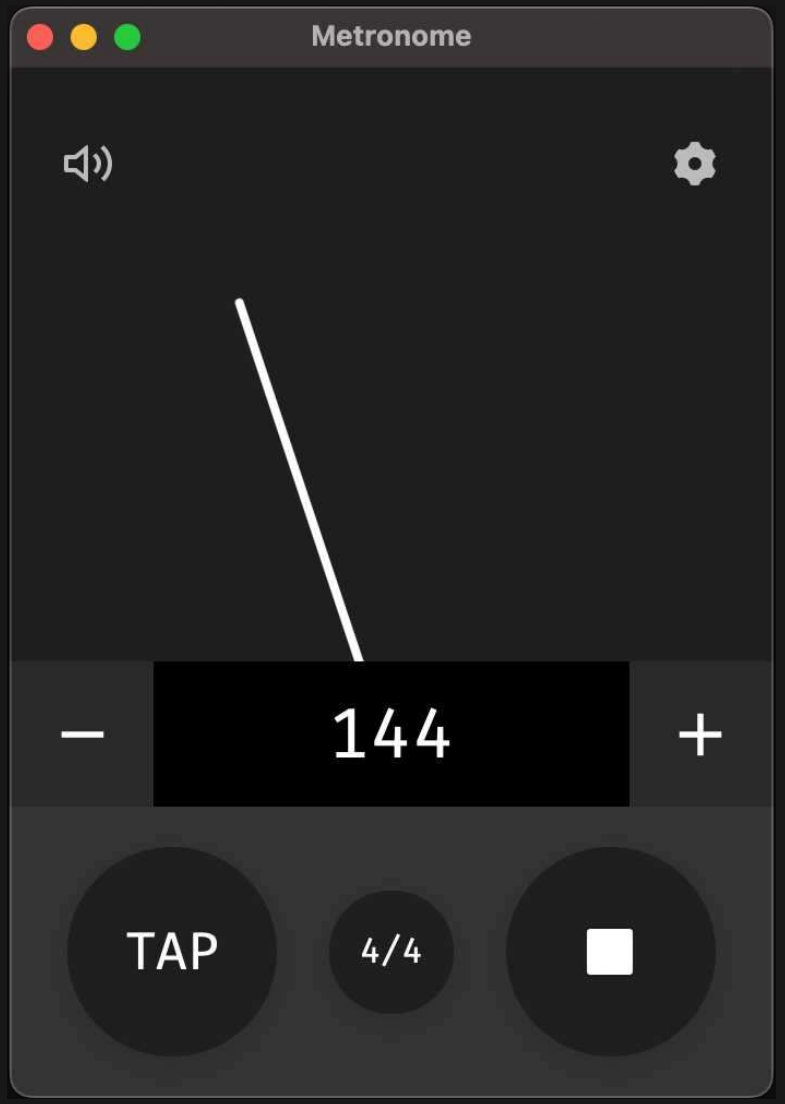
- created by former Qt engineers
- works additionally on Mobile and MCUs
- Qt-style markup language
- more complicated licensing (royalty free option)
- no official or easy way to to plugins with it yet

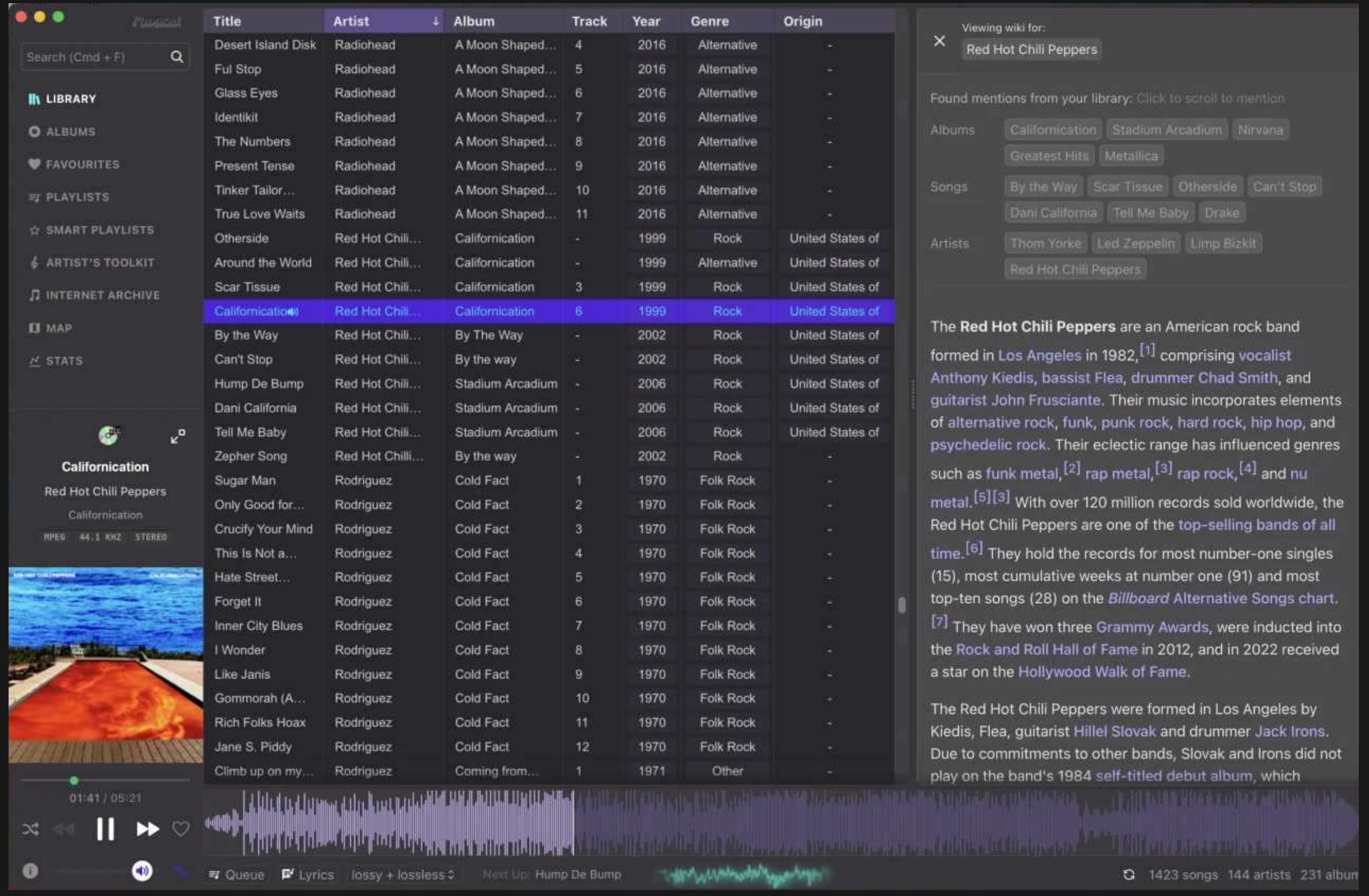
Web User Interface

Tauri

- backed CrabNebula, 1Password and many more
- MIT or Apache-2.0 license
- 84k stars, 401 contributors







Title	Artist	Album	Track	Year	Genre	Origin
Desert Island Disk	Radiohead	A Moon Shaped…	4	2016	Alternative	-
Ful Stop	Radiohead	A Moon Shaped…	5	2016	Alternative	-
Glass Eyes	Radiohead	A Moon Shaped…	6	2016	Alternative	-
Identikit	Radiohead	A Moon Shaped…	7	2016	Alternative	-
The Numbers	Radiohead	A Moon Shaped…	8	2016	Alternative	-
Present Tense	Radiohead	A Moon Shaped…	9	2016	Alternative	-
Tinker Tailor…	Radiohead	A Moon Shaped…	10	2016	Alternative	-
True Love Waits	Radiohead	A Moon Shaped…	11	2016	Alternative	-
Otherside	Red Hot Chili…	Californication	-	1999	Rock	United States of
Around the World	Red Hot Chili…	Californication	-	1999	Alternative	United States of
Scar Tissue	Red Hot Chili…	Californication	3	1999	Rock	United States of
Californication	Red Hot Chili…	Californication	6	1999	Rock	United States of
By the Way	Red Hot Chili…	By The Way	-	2002	Rock	United States of
Can't Stop	Red Hot Chili…	By the way	-	2002	Rock	United States of
Hump De Bump	Red Hot Chili…	Stadium Arcadium	-	2006	Rock	United States of
Dani California	Red Hot Chili…	Stadium Arcadium	-	2006	Rock	United States of
Tell Me Baby	Red Hot Chili…	Stadium Arcadium	-	2006	Rock	United States of
Zepher Song	Red Hot Chilli…	By the way	-	2002	Rock	-
Sugar Man	Rodriguez	Cold Fact	1	1970	Folk Rock	-
Only Good for…	Rodriguez	Cold Fact	2	1970	Folk Rock	-
Crucify Your Mind	Rodriguez	Cold Fact	3	1970	Folk Rock	-
This Is Not a…	Rodriguez	Cold Fact	4	1970	Folk Rock	-
Hate Street…	Rodriguez	Cold Fact	5	1970	Folk Rock	-
Forget It	Rodriguez	Cold Fact	6	1970	Folk Rock	-
Inner City Blues	Rodriguez	Cold Fact	7	1970	Folk Rock	-
I Wonder	Rodriguez	Cold Fact	8	1970	Folk Rock	-
Like Janis	Rodriguez	Cold Fact	9	1970	Folk Rock	-
Gommorah (A…	Rodriguez	Cold Fact	10	1970	Folk Rock	-
Rich Folks Hoax	Rodriguez	Cold Fact	11	1970	Folk Rock	-
Jane S. Piddy	Rodriguez	Cold Fact	12	1970	Folk Rock	-
Climb up on my…	Rodriguez	Coming from…	1	1971	Other	-

Viewing wiki for:
Red Hot Chili Peppers

Found mentions from your library: Click to scroll to mention

Albums	Californication, Stadium Arcadium, Nirvana, Greatest Hits, Metallica
Songs	By the Way, Scar Tissue, Otherside, Can't Stop, Dani California, Tell Me Baby, Drake
Artists	Thom Yorke, Led Zeppelin, Limp Bizkit, Red Hot Chili Peppers

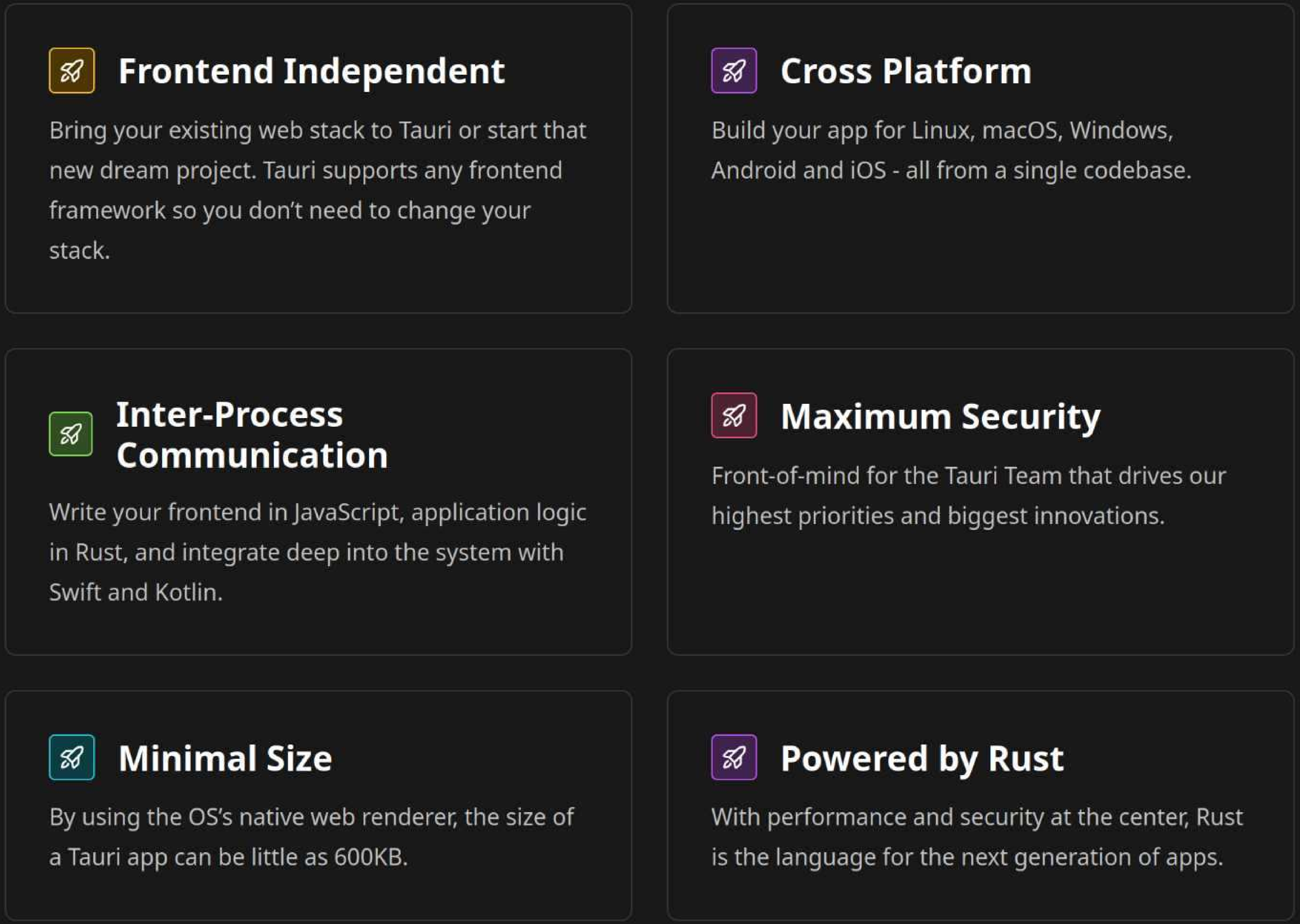
The **Red Hot Chili Peppers** are an American rock band formed in Los Angeles in 1982,[1] comprising vocalist Anthony Kiedis, bassist Flea, drummer Chad Smith, and guitarist John Frusciante. Their music incorporates elements of alternative rock, funk, punk rock, hard rock, hip hop, and psychedelic rock. Their eclectic range has influenced genres such as funk metal,[2] rap metal,[3] rap rock,[4] and nu metal.[5][3] With over 120 million records sold worldwide, the Red Hot Chili Peppers are one of the top-selling bands of all time.[6] They hold the records for most number-one singles (15), most cumulative weeks at number one (91) and most top-ten songs (28) on the *Billboard* Alternative Songs chart.[7] They have won three Grammy Awards, were inducted into the Rock and Roll Hall of Fame in 2012, and in 2022 received a star on the Hollywood Walk of Fame.

The Red Hot Chili Peppers were formed in Los Angeles by Kiedis, Flea, guitarist Hillel Slovak and drummer Jack Irons. Due to commitments to other bands, Slovak and Irons did not play on the band's 1984 self-titled debut album, which

1423 songs 144 artists 231 albums

Tauri Features

- built in app bundlers
  - + github actions
  - + notarization
- built in self-updater
- sidecars
- rich plugin system
  - global shortcuts
  - system tray icons
  - biometrics
  - NFC
  - web sockets
  - many more ...



Frontend Independent

Bring your existing web stack to Tauri or start that new dream project. Tauri supports any frontend framework so you don't need to change your stack.

Cross Platform

Build your app for Linux, macOS, Windows, Android and iOS - all from a single codebase.

Inter-Process Communication

Write your frontend in JavaScript, application logic in Rust, and integrate deep into the system with Swift and Kotlin.

Maximum Security

Front-of-mind for the Tauri Team that drives our highest priorities and biggest innovations.

Minimal Size

By using the OS's native web renderer, the size of a Tauri app can be little as 600KB.

Powered by Rust

With performance and security at the center, Rust is the language for the next generation of apps.

Commands

- only work Frontend → Rust
- can accept arguments and return objects and errors
- can be used for small and large data (with ipc::Response)
- can be *async* or *blocking*
- can access global app state and the app window

Rust

```rust
#[tauri::command]
fn greet(
  name: String,
  state: State<'_, Mutex<AppState>>
) -> Result<String, String> {
    let state = state.lock();
    if state.allowed_names.contains(&name) {
        Ok(format!("Hello {name}!"))
    } else {
        Err(format!("Name {name} is not allowed"))
    }
}
```

JS

```js
invoke("greet", { name: "Stephan" })
  .then((message) => console.log(message))
  .catch((error) => console.error(error));
```

Events

- work bi-directional
- are simpler and not type safe
- only work for small data
  - push notifications
- are always async

Channels

- work bi-directional
- deliver data ordered
- faster and good for streaming data
  - download progress
  - level meters
  - web socket messages

Compatibility

- Windows
- macOS
- Linux (bundlers for Debian, Fedora, Arch, Flathub)
- Android
- iOS

(for audio plugins there is an *experimental* project
nih-plug-webview, using Tauri's webview renderer as editor)

Resources

- Official Tauri guide is extremely well written https://tauri.app/start/
- Awesome Tauri https://github.com/tauri-apps/awesome-tauri

Audio Plugins

clap-sys, vst3-sys

- unsafe bindings of the C-API
- highest level of control
- difficult to write and not “rusty” (raw pointers 😱)

```rust
#[repr(C)]
#[derive(Debug, Copy, Clone)]
pub struct clap_process {
    pub steady_time: i64,
    pub frames_count: u32,
    pub transport: *const clap_event_transport,
    pub audio_inputs: *const clap_audio_buffer,
    pub audio_outputs: *mut clap_audio_buffer,
    pub audio_inputs_count: u32,
    pub audio_outputs_count: u32,
    pub in_events: *const clap_input_events,
    pub out_events: *const clap_output_events,
}
```

Clack

- safe wrapper around clap-sys
- hosts and plugins possible
- still quite low-level

```rust
pub struct GainPluginAudioProcessor {}

impl PluginAudioProcessor for GainPluginAudioProcessor {
    fn activate(
        host: HostAudioProcessorHandle<'a>,
        main_thread: &mut GainPluginMainThread,
        shared: &'a GainPluginShared,
        audio_config: PluginAudioConfiguration
    ) -> Result<Self, PluginError> {
        ...
    }

    fn process(
        &mut self,
        process: Process,
        mut audio: Audio,
        events: Events
    ) -> Result<ProcessStatus, PluginError> {
        ...
    }
}
```

nih-plug

- full plugin framework for VST3 and CLAP
- very simple to use (whole plugin can be one file)
- three UI library choices already “wired up”
- well documented example plugins
- many features
  - thread safe parameters
  - parameter smoothing
  - automatically stores plugin state
  - *optional:* fails to compile if process function allocates memory

nih-plug

```rust
struct MyPlugin {
    params: Arc<MyParams>,
}

#[derive(Params)]
struct MyParams {
    #[id = "gain"]
    pub gain: FloatParam,
}

impl Default for MyParams {
    fn default() -> Self {
        Self {
            gain: FloatParam::new("Gain", util::db_to_gain(0.0), FloatRange::Skewed {
                min: util::db_to_gain(-30.0),
                max: util::db_to_gain(30.0),
                factor: FloatRange::gain_skew_factor(-30.0, 30.0),
            })
                .with_smoother(SmoothingStyle::Logarithmic(50.0))
                .with_unit(" dB")
                .with_value_to_string(formatters::v2s_f32_gain_to_db(2))
                .with_string_to_value(formatters::s2v_f32_gain_to_db()),
        }
    }
}
```

nih-plug

```rust
impl Plugin for MyPlugin {
    const NAME: &'static str = "My Plugin";
    const VENDOR: &'static str = "Me";
    const URL: &'static str = "me.com";
    const EMAIL: &'static str = "info@me.com";
    const VERSION: &'static str = env!("CARGO_PKG_VERSION");

    const AUDIO_IO_LAYOUTS: &'static [AudioIOLayout] = &[
        AudioIOLayout {
            main_input_channels: NonZeroU32::new(2),
            main_output_channels: NonZeroU32::new(2),
            aux_input_ports: &[],
            aux_output_ports: &[],
            names: PortNames::const_default(),
        },
    ];
  ...
```

nih-plug

```
impl Plugin for MyPlugin {
    fn initialize(
        &mut self,
        audio_io_layout: &AudioIOLayout,
        buffer_config: &BufferConfig,
        context: &mut impl InitContext<Self>
    ) -> bool {
        true
    }

    fn reset(&mut self) {}

    fn process(
        &mut self,
        buffer: &mut Buffer,
        aux: &mut AuxiliaryBuffers,
        context: &mut impl ProcessContext<Self>
    ) -> ProcessStatus {
        ProcessStatus::Normal
    }

    fn deactivate(&mut self) {}
}
```

nih-plug

```rust
impl ClapPlugin for MyPlugin {
    const CLAP_ID: &'static str = "com.me.myplugin";
    const CLAP_DESCRIPTION: Option<&'static str> = Some("A plugin");
    const CLAP_MANUAL_URL: Option<&'static str> = Some(Self::URL);
    const CLAP_SUPPORT_URL: Option<&'static str> = None;
    const CLAP_FEATURES: &'static [ClapFeature] = &[
        ClapFeature::AudioEffect,
        ClapFeature::Stereo,
        ClapFeature::Mono,
        ClapFeature::Utility,
    ];
}

impl Vst3Plugin for MyPlugin {
    const VST3_CLASS_ID: [u8; 16] = *b"MyPlugin";
    const VST3_SUBCATEGORIES: &'static [Vst3SubCategory] =
        &[Vst3SubCategory::Fx, Vst3SubCategory::Tools];
}

nih_export_clap!(MyPlugin);
nih_export_vst3!(MyPlugin);
```

Problem: Audio Unit / AAX

- no wrapper for Audio Unit or AAX at the moment

- simplest solution for AU → clap-wrapper
  - official solution by CLAP
  - can create VST3, AUv2 and standalone plugins

    (dynamically loading CLAP plugin, so AU only works if CLAP plugin is in the right path)

Debugging Tools

Built-in Tests

- Rust allows running test in the same file as the function under test
- no complicated setup, file-switching or linking necessary
- ***Cargo.toml*** let's you specify dependencies only used in tests
- People write more tests if it is less effort!!

```rust
pub fn add(left: u64, right: u64) -> u64 {
    left + right
}

#[cfg(test)]
mod tests {
    use super::*;

    #[test]
    fn it_works() {
        let result = add(2, 2);
        assert_eq!(result, 4);
    }
}
```

```
$ cargo test
```

```
running 1 test
test tests::it_works ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

   Doc-tests my_tests

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

assert_no_alloc

- quickly check if your code allocates memory
- only works for Rust code (not if you call into a C++ library)
- can be integrated in tests and keep your process function alloc-free

1. define global allocator

```rust
use assert_no_alloc::*;

#[cfg(debug_assertions)]
#[global_allocator]
static A: AllocDisabler = AllocDisabler;
```

2. test if your code allocates

```rust
#[test]
fn test_no_alloc() {
    assert_no_alloc(|| {
        let sum = (0..1_000_000).map(|v| v as f32).sum::<f32>();
    });
}
```

🤔 does it allocate?

Criterion Benchmarks

- feature rich benchmarking tool with HTML reports and history
- benchmarks in separate files (nightly rust already in the same file)
- can be run with built-in ***cargo bench*** command

```rust
use criterion::{black_box, criterion_group, criterion_main, Criterion};

pub fn my_benchmark(c: &mut Criterion) {
    c.bench_function("my benchmark", |b| {
        b.iter(|| {
            let a = (0..1_000_000).map(|v| v as f32).sum::<f32>();
            black_box(a);
        })
    });
}

criterion_group!(benches, my_benchmark);
criterion_main!(benches);
```

```
$ cargo bench
```

```
my benchmark            time:   [532.67 µs 533.86 µs 535.34 µs]
                        change: [-10.944% -9.9111% -8.7945%] (p = 0.00 < 0.05)
                        Performance has improved.
Found 14 outliers among 100 measurements (14.00%)
  11 (11.00%) high mild
  3 (3.00%) high severe
```

quickcheck

- property testing / fuzz testing
- creates a big amount of random data in your ranges and feeds it into your code
- automatic shrinking (tries to find the simplest input where a failure still happens)

```rust
#[cfg(test)]
mod tests {
    fn reverse<T: Clone>(xs: &[T]) -> Vec<T> {
        let mut rev = vec!();
        for x in xs {
            rev.insert(0, x.clone())
        }
        rev
    }

    #[quickcheck]
    fn double_reversal_is_identity(xs: Vec<isize>) -> bool {
        xs == reverse(&reverse(&xs))
    }
}
```

Sanitizers

- only check **this** run on **this** architecture
- usually you will force excessive use and set the system under load to enforce problems

Guide:

https://github.com/japaric/rust-san

Miri

- detection tool for undefined behavior
- can run binaries or your tests
- good for verifying unsafe code
- can check
  - out-of-bounds memory access
  - use after free
  - data races
  - memory leaks
  - ….

Loom

- checks your assumption for every allowed value that your thread could see

Limits:

- can not check every type of reordering for Ordering::Relaxed
- can have false positives with SeqCst

```rust
use loom::sync::Arc;
use loom::sync::atomic::AtomicUsize;
use loom::sync::atomic::Ordering::{Acquire, Release, Relaxed};
use loom::thread;

#[test]
#[should_panic]
fn buggy_concurrent_inc() {
    loom::model(|| {
        let num = Arc::new(AtomicUsize::new(0));

        let ths: Vec<_> = (0..2)
            .map(|_| {
                let num = num.clone();
                thread::spawn(move || {
                    let curr = num.load(Acquire);
                    num.store(curr + 1, Release);
                })
            })
            .collect();

        for th in ths {
            th.join().unwrap();
        }

        assert_eq!(2, num.load(Relaxed));
    });
}
```

Recommended Audio Crates

rubato real-time resampling

rustfft / ***realfft*** / ***easyfft*** (c2c / r2c / tests)

babycat easiest audio file IO + resampling

hound just wav file IO

symphonia audio file decoding and disk streaming neo neodsp

work in progess:

neolab research and offline audio

github.com/neodsp/neolab

neort real-time audio

github.com/neodsp/neort

Contact Info

Mail: stephan@neodsp.com

LinkedIn: linkedin.com/in/stephan-eckes/

GitHub: github.com/sainteckes

Thank you!