Rust Audio/Music Programming

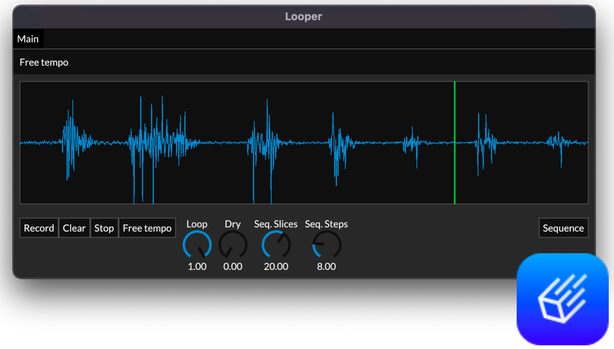
Pedro Tacla Yamada - 05/12/2023

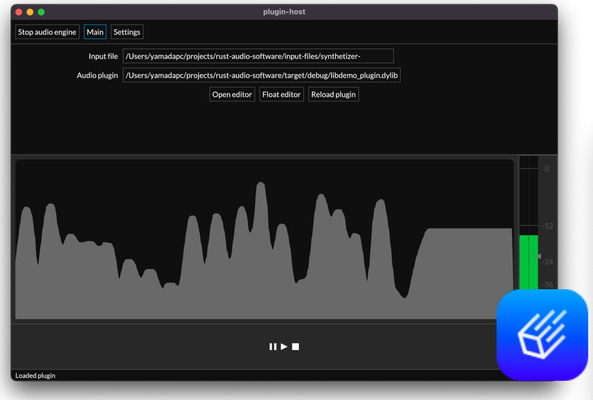
Rust Applications

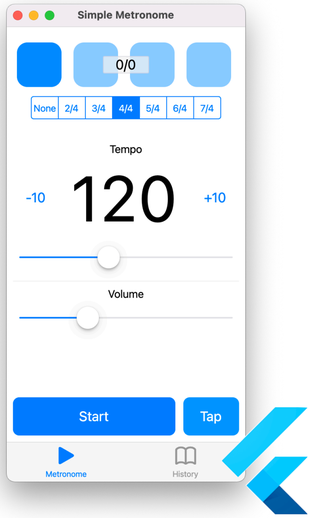
Augmented Audio Libraries

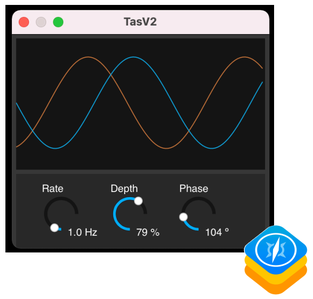
Fully open-source, MIT licensed*, github.com/yamadapc/augmented-audio

*Some full applications are GPL

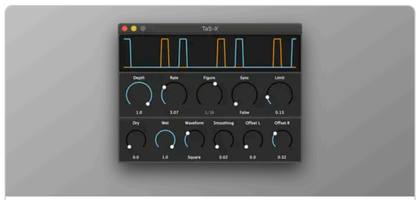








PD1 - Analog Delay
Delay plug-in for weird sounds



TAS-X
Stereo tremolo and slicer

JUCE

Topics

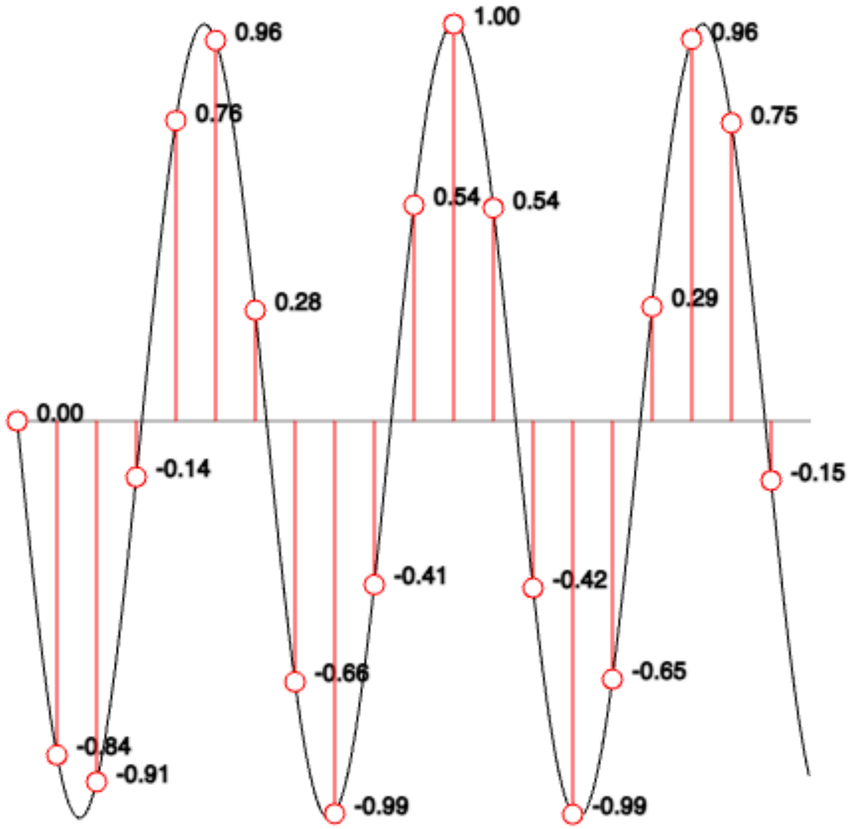
- Digital audio basics
- Real-time constraints and Rust
- Architecture of audio/music software
- UI Development

Topics

- Digital audio basics
- Real-time constraints and Rust
- Architecture of audio/music software
- UI Development

Digital Audio

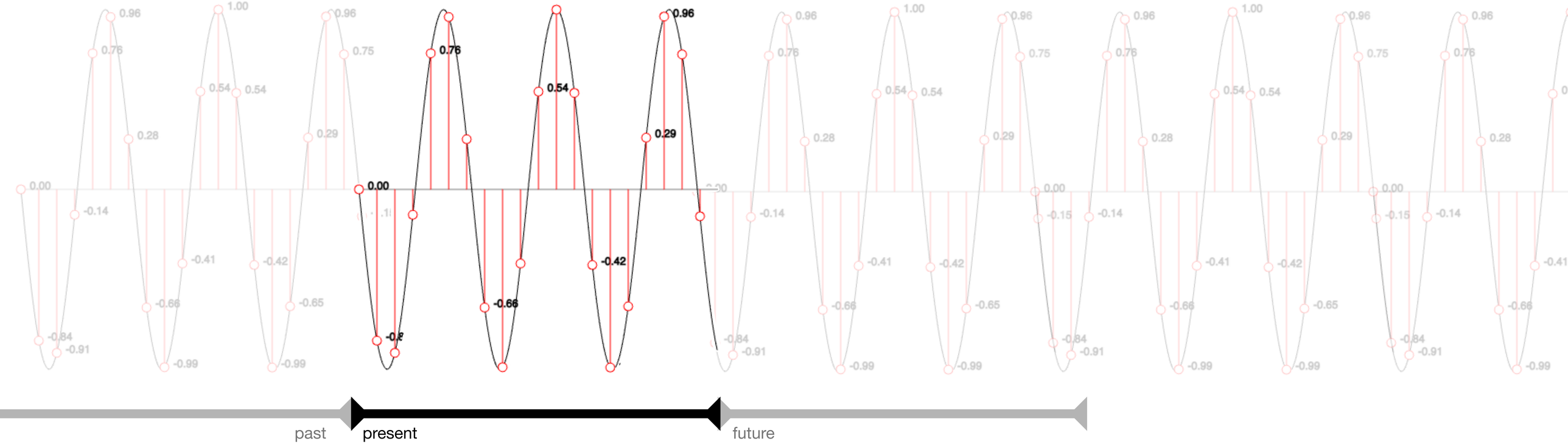
Audio signal representation



- Audio signal will be discretised onto sequence of numbers through sampling
- Samples are taken at a *sample rate*, such as 44.1kHz

Digital Audio

Buffering



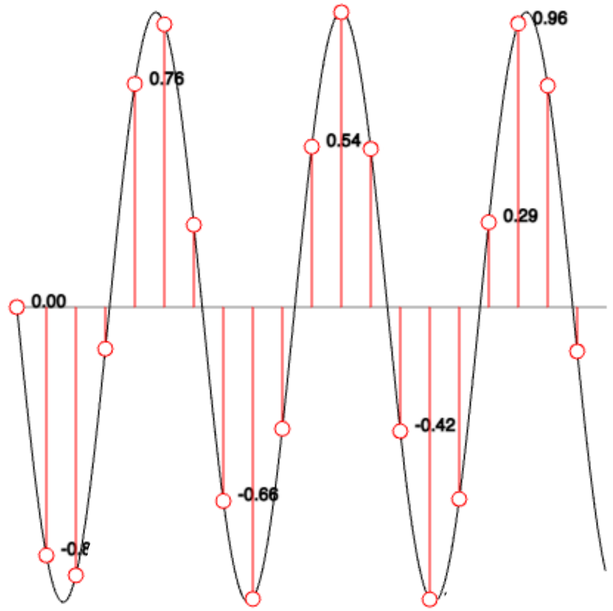
Read into buffer

Process

Output

Digital Audio

Buffering



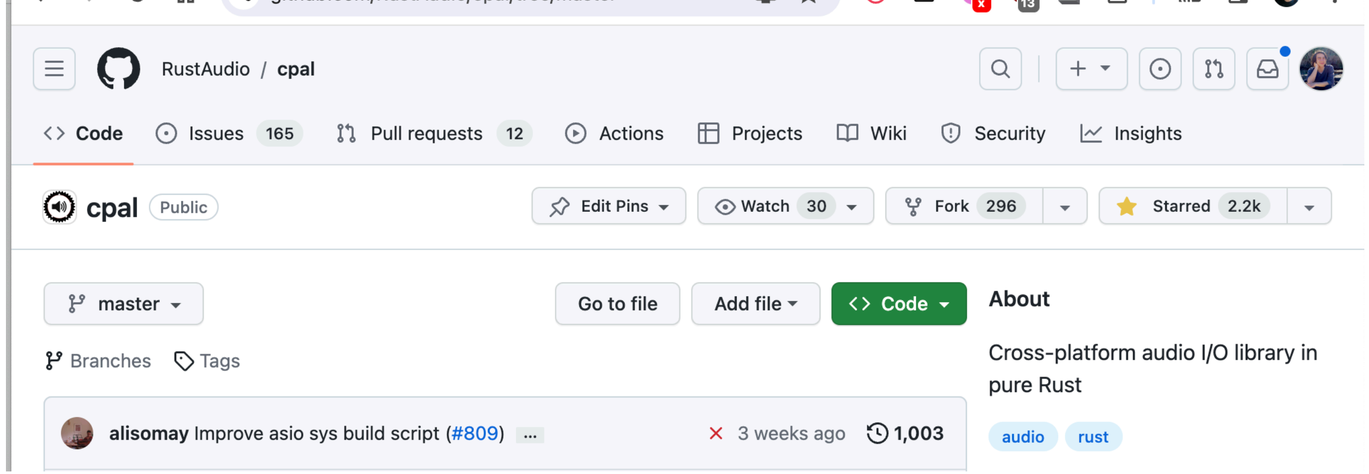
present

Read into buffer → **Process** → Output

- There is a fixed deadline for “process” to run for

CPAL

```rust
let host = cpal::default_host();
let output_device = host
    .default_output_device()
    .expect("No output device");

let stream_config = cpal::StreamConfig {
    channels: 2,
    sample_rate: cpal::SampleRate(44100),
    buffer_size: cpal::BufferSize::Default,
};

let stream = output_device
    .build_output_stream(
        &stream_config,
        process,
        |err| eprintln!("An error occurred on stream: {}", err),
        None,
    )
    .expect("Failed to build output stream");

stream.play().expect("Failed to play stream");
```



CPAL

```rust
let host = cpal::default_host();
let output_device = host
    .default_output_device()
    .expect("No output device");

let stream_config = cpal::StreamConfig {
    channels: 2,
    sample_rate: cpal::SampleRate(44100),
    buffer_size: cpal::BufferSize::Default,
};

let stream = output_device
    .build_output_stream(
        &stream_config,
        process,
        |err| eprintln!("An error occurred on stream: {}", err),
        None,
    )
    .expect("Failed to build output stream");

stream.play().expect("Failed to play stream");
```

CPAL

```
let host = cpal::default_host();
let output_device = host
    .default_output_device()
    .expect("No output device");

let stream_config = cpal::StreamConfig {
    channels: 2,
    sample_rate: cpal::SampleRate(44100),
    buffer_size: cpal::BufferSize::Default,
};

let stream = output_device
    .build_output_stream(
        &stream_config,
        process,
        |err| eprintln!("An error occurred on stream: {}", err),
        None,
    )
    .expect("Failed to build output stream");

stream.play().expect("Failed to play stream");
```

CPAL

```
let host = cpal::default_host();
let output_device = host
    .default_output_device()
    .expect("No output device");

let stream_config = cpal::StreamConfig {
    channels: 2,
    sample_rate: cpal::SampleRate(44100),
    buffer_size: cpal::BufferSize::Default,
};

let stream = output_device
    .build_output_stream(
        &stream_config,
        process,
        |err| eprintln!("An error occurred on stream: {}", err),
        None,
    )
    .expect("Failed to build output stream");

stream.play().expect("Failed to play stream");
```

CPAL

```
let default_host = default_host();
let output_device = default_host
    .default_output_device()
    .expect("No output device");

let stream_config = cpal::StreamConfig {
    channels: 2,
    sample_rate: cpal::SampleRate(44100),
    buffer_size: cpal::BufferSize::Default,
};

let stream = output_device
    .build_output_stream(
        &stream_config,
        process,
        |err| eprintln!("An error occurred on stream: {}", err),
        None,
    )
    .expect("Failed to build output stream");

stream.play().expect("Failed to play stream");
```

CPAL

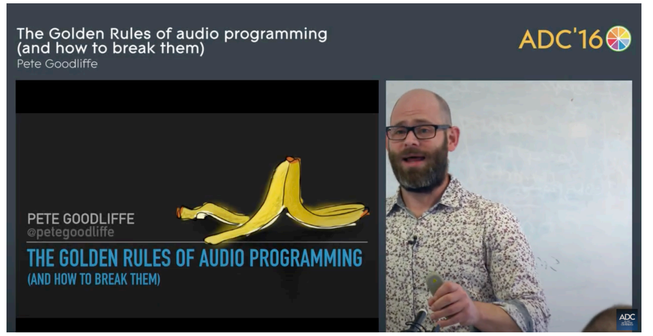
```
let frequency = 440.0;
let mut phase: f32 = 0.0;
let inverse_sample_rate = 1.0 / 44100.0;
let phase_increment = frequency * inverse_sample_rate * 2.0 * std::f32::consts::PI;

let process = move |data: &mut [f32], _: &cpal::OutputCallbackInfo| {
    for chans in data.chunks_mut(2) {
        chans[0] = phase.sin();
        chans[1] = phase.sin();
        phase += phase_increment;
    }
};
```

Topics

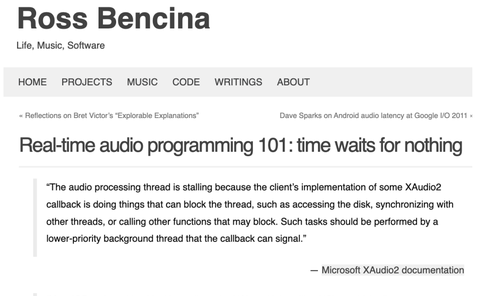
- Digital audio basics
- Real-time constraints and Rust
- Architecture of audio/music software
- UI Development

Realtime constraints and Rust

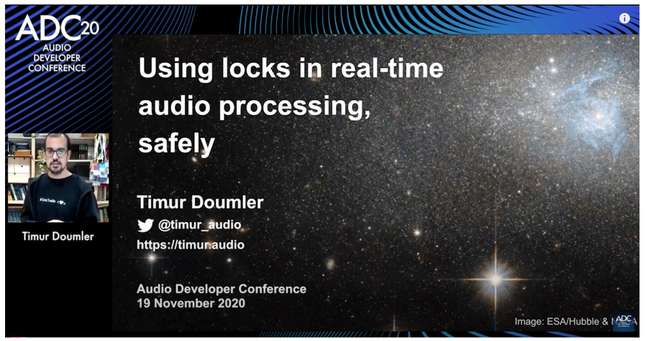
- **There is a fixed hard deadline for “process”**
- This means, on the audio-thread
  - No IO
  - No locks
  - No waiting
  - No memory allocation/de-allocation
  - No unbounded work



https://www.youtube.com/watch?v=SJXGSJ6Zoro



rossbencina.com/code/real-time-audio-programming-101-time-waits-for-nothing



youtube.com/watch?v=zrWYJ6FdOFQ

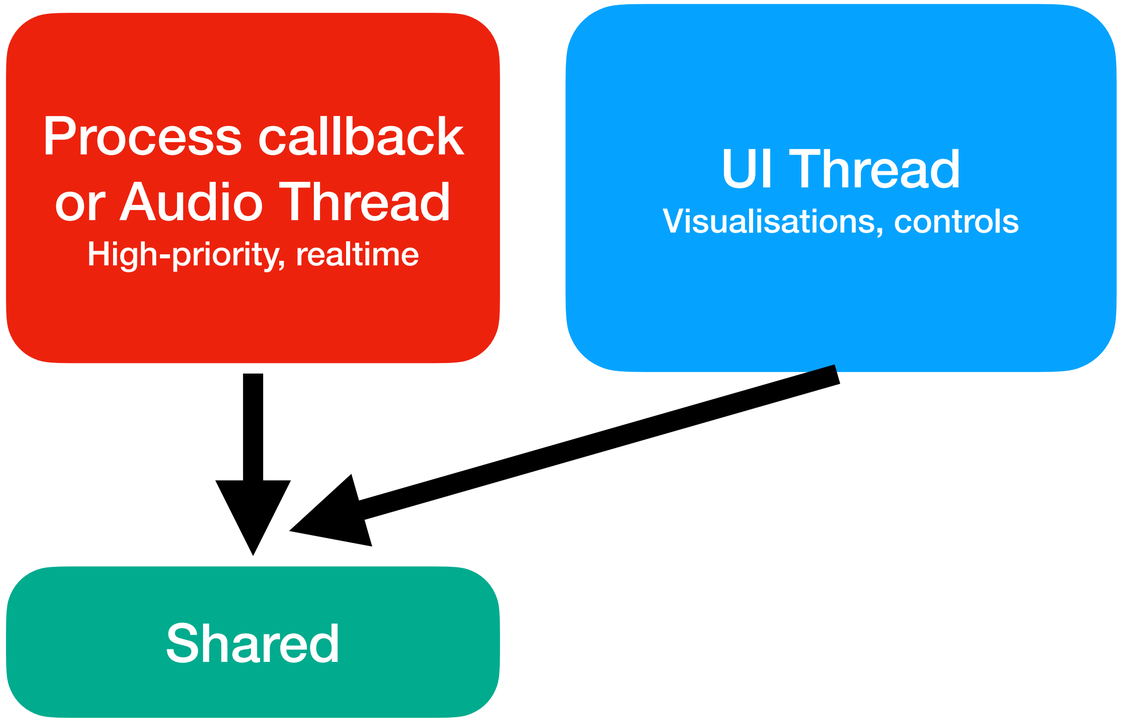
Realtime constraints and Rust

- No IO
- No locks
- No waiting
- No memory allocation/de-allocation
- No unbounded work

- Do IO on separate thread
- Use lock-free data-structures and patterns
  - Use atomics
  - Use message passing via lock-free ring-buffer or queue
  - Use pointer to data, and swap its reference atomically
  - Solve “memory reclamation” problem in a realtime safe manner

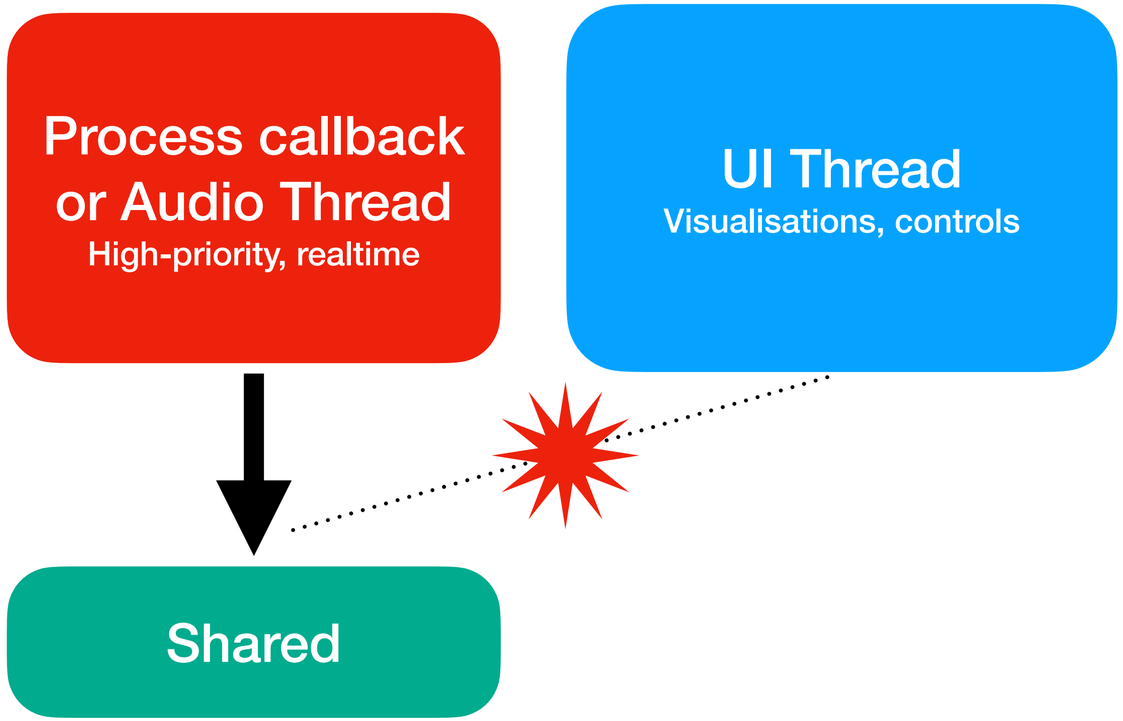
Realtime constraints and Rust

No memory allocation/de-allocation on audio-thread



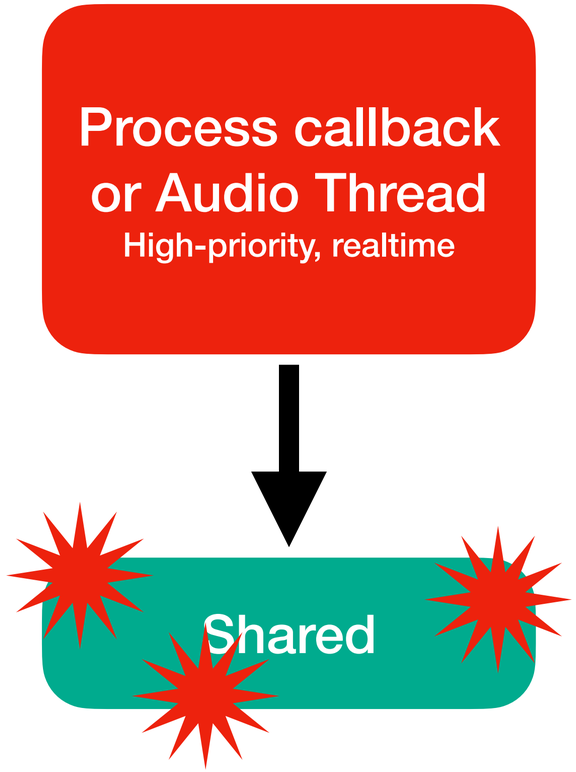
Realtime constraints and Rust

No memory allocation/de-allocation on audio-thread



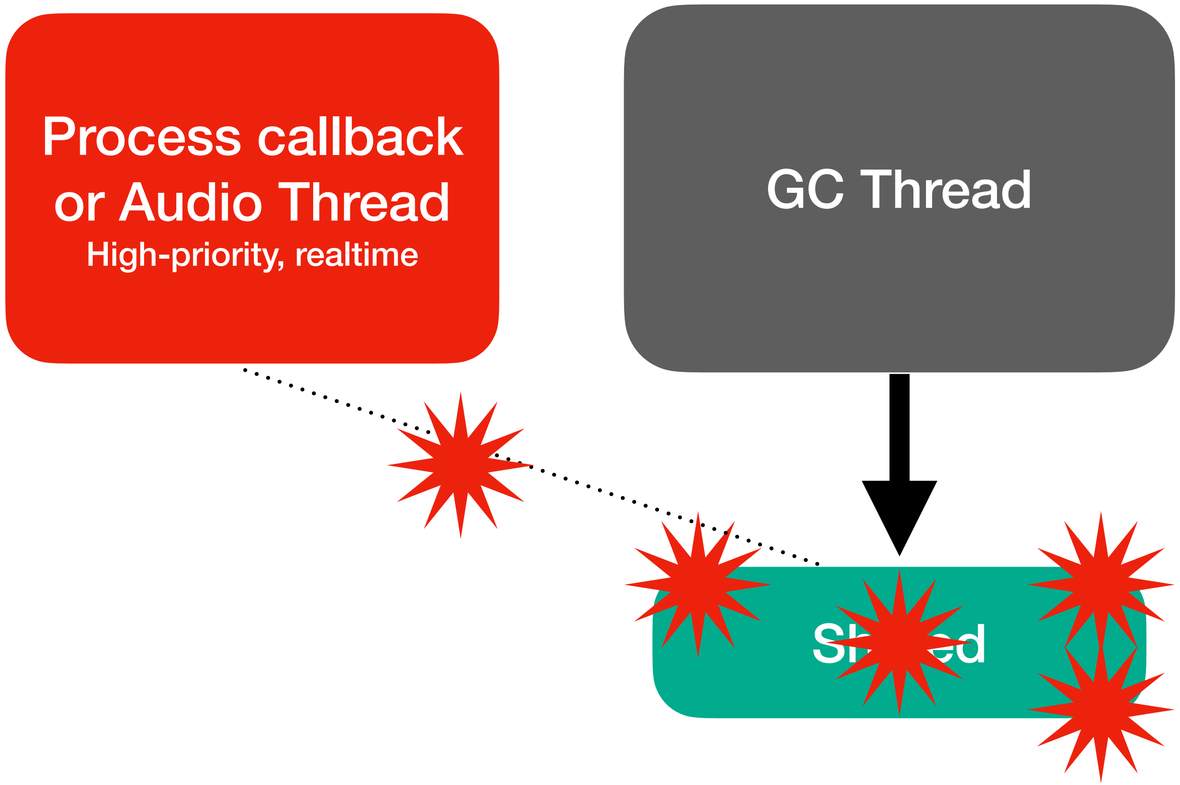
Realtime constraints and Rust

No memory allocation/de-allocation on audio-thread



Realtime constraints and Rust

No memory allocation/de-allocation on audio-thread



Realtime constraints and Rust

No memory allocation/de-allocation on audio-thread

```
let mut collector = Collector::new();
let ref_counted = Shared::new(collector.handle(), _);

// You can control when / where this runs
collector.collect();
```

https://github.com/glowcoil/basedrop

Realtime constraints and Rust

No memory allocation/de-allocation on audio-thread

https://crates.io/crates/audio-garbage-collector

https://github.com/glowcoil/basedrop

```
use audio_garbage_collector::*;

let ref_counted = make_shared(10);

// This crate is wrapping a background thread that does
// poll based clean-up, default returns the global instance
let mut gc = GarbageCollector::default(); // or ::new
```

Realtime constraints and Rust

Use atomics

```
use augmented_atomics::AtomicF32;

let gain_parameter = AtomicF32::new(value);

let gain_plugin_handle = make_shared(gain_parameter);
let gain_plugin_handle_for_ui = gain_plugin_handle.clone();

// send ref. to UI
gain_parameter_handle.gain_parameter.set(0.5);
let gain = gain_parameter_handle.gain_parameter.get();
```

https://crates.io/crates/augmented-atomics

Realtime constraints and Rust

Use message passing

```
let (tx, rx) = ringbuf::RingBuffer::new(10).split();
let app = AppAudioProcessor {
  // ...omitted ...
  messages: rx,
};
```

```
impl AppAudioProcessor {
    fn drain_message_queue(&mut self) {
        // Can limit
        while let Some(message) = self.messages.pop() {
            match message {
                AppMessage::SetSound(sound) => {
                    self.set_metronome_sound(sound);
                }
            }
        }
    }
}
```

```
impl AudioProcessor for AppAudioProcessor {
    // ... ...
    fn process(...) {
        self.drain_message_queue();
        // process buffer
   }
}
```

https://github.com/yamadapc/augmented-audio/blob/master/crates/apps/metronome/src/api/processor.rs#L158-L164

https://github.com/yamadapc/augmented-audio/blob/master/crates/augmented/audio/audio-processor-metronome/src/lib.rs

Realtime constraints and Rust

Use message passing

atomic-queue

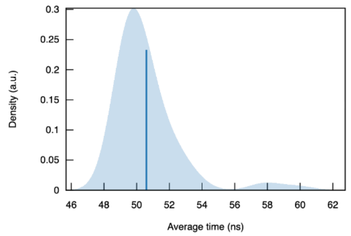
crates.io v1.0.1 | docs passing

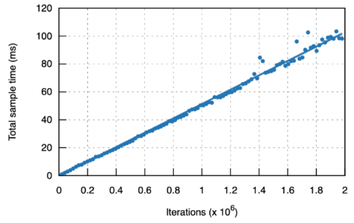
Multi-producer multi-consumer bounded lock-free queue for use in Audio applications, ported from https://github.com/max0x7ba/atomic_queue.

Quite a bit slower than `ringbuf` (~2x) on i7.

I'd think this is fine since this queue supporting multiple consumers and multiple producers while `ringbuf` is single producer single consumer.

30% faster on a M1 Pro Macbook.





Additional Statistics:

	Lower bound	Estimate	Upper bound
Slope	50.788 ns	51.370 ns	52.019 ns
R²	0.7416741	0.7530013	0.7389942
Mean	50.221 ns	50.594 ns	51.014 ns
Std. Dev.	1.3689 ns	2.0459 ns	2.6265 ns
Median	49.762 ns	50.067 ns	50.410 ns
MAD	819.26 ps	1.1105 ns	1.4416 ns

Additional Plots:

- Typical
- Mean
- Std. Dev.
- Median
- MAD
- Slope

License

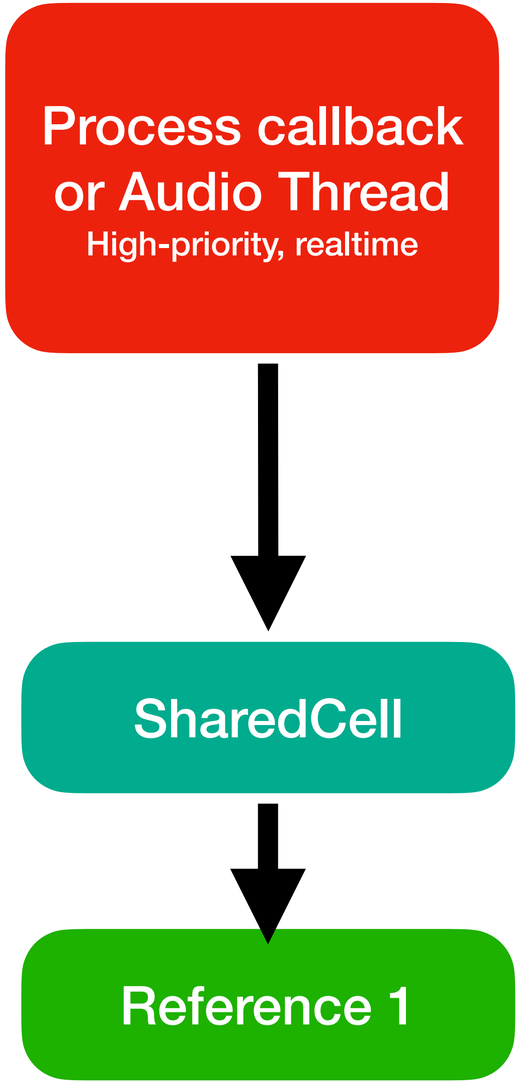
MIT

https://crates.io/crates/atomic-queue

Realtime constraints and Rust

pointer swap

```
let c = make_shared_cell(filter_coeffs);
c.set(make_shared(filter_coeffs2));
```

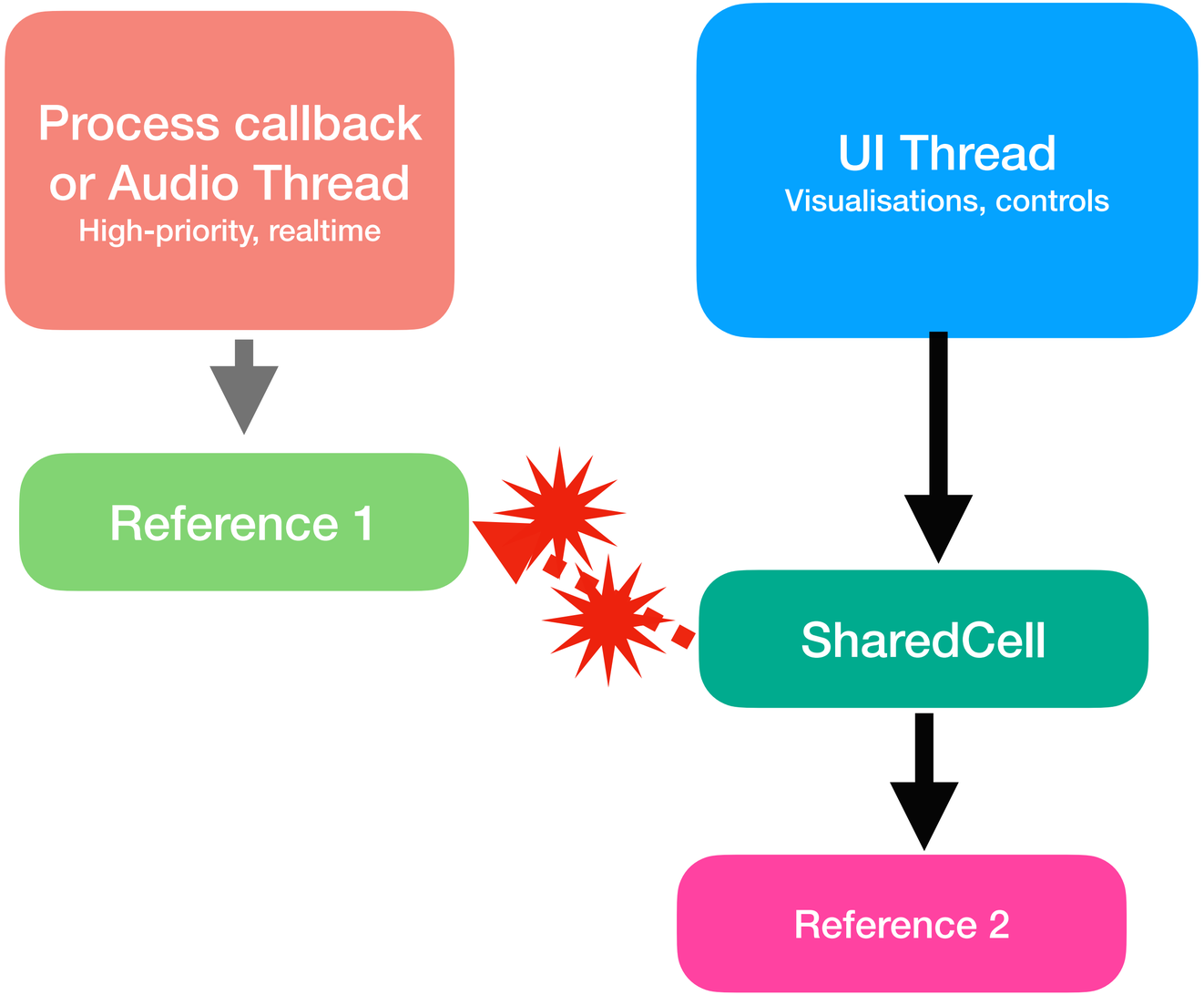


/glowcoil/basedrop

Realtime constraints and Rust

pointer swap

```
let c = make_shared_cell(filter_coeffs);
c.set(make_shared(filter_coeffs2));
```

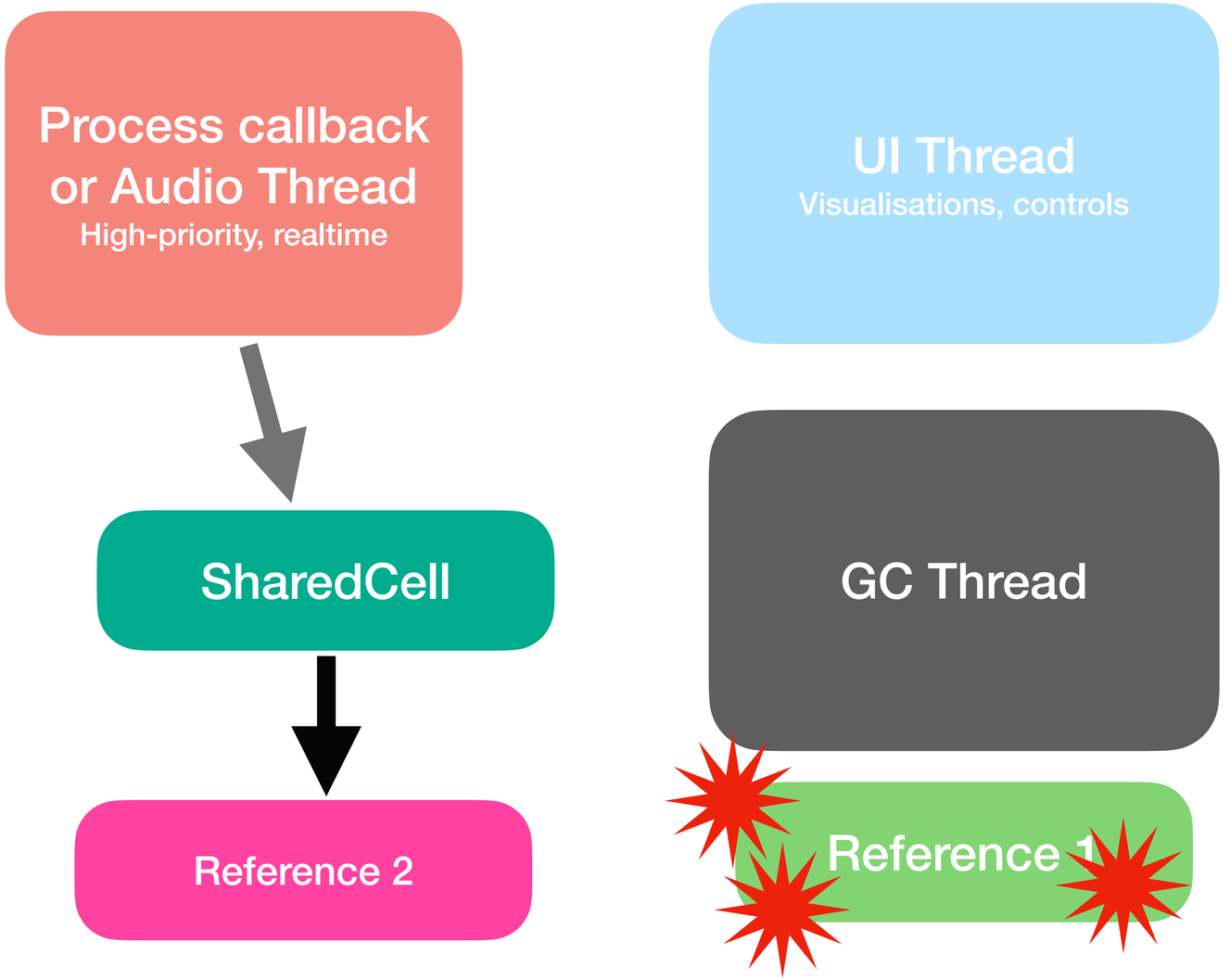


/glowcoil/basedrop

Realtime constraints and Rust

pointer swap

```
let c = make_shared_cell(filter_coeffs);
c.set(make_shared(filter_coeffs2));
```

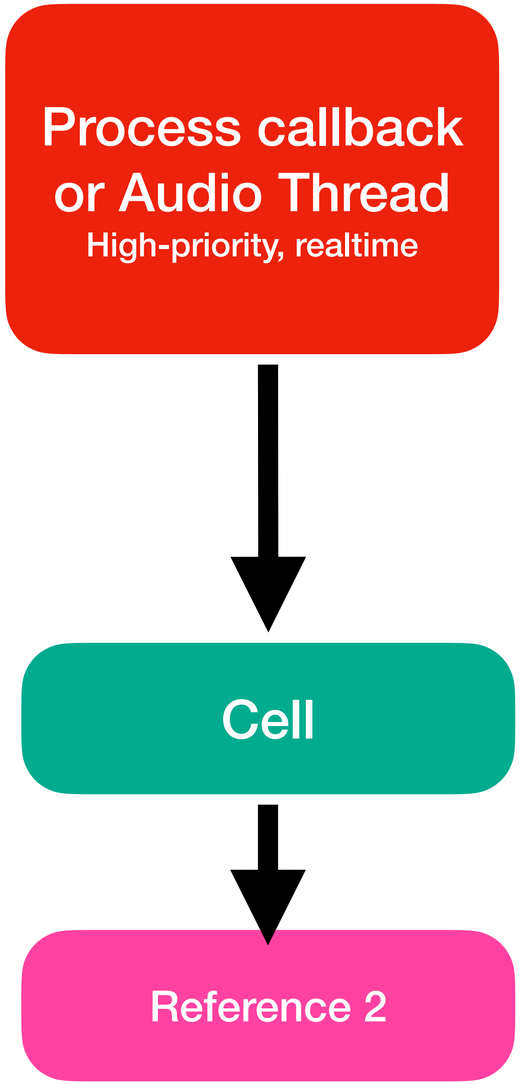


/glowcoil/basedrop

Realtime constraints and Rust

pointer swap

```
let c = make_shared_cell(filter_coeffs);
c.set(make_shared(filter_coeffs2));
```



/glowcoil/basedrop

Realtime constraints and Rust

No memory allocation

```
use assert_no_alloc::assert_no_alloc;

assert_no_alloc(|| {
    // This code will panic/log if it allocates
});
```

https://crates.io/crates/assert_no_alloc

Realtime constraints and Rust

No memory allocation

RealtimeSanitizer (a.k.a. RADSan) is a real time safety testing tool for C and C++ projects.

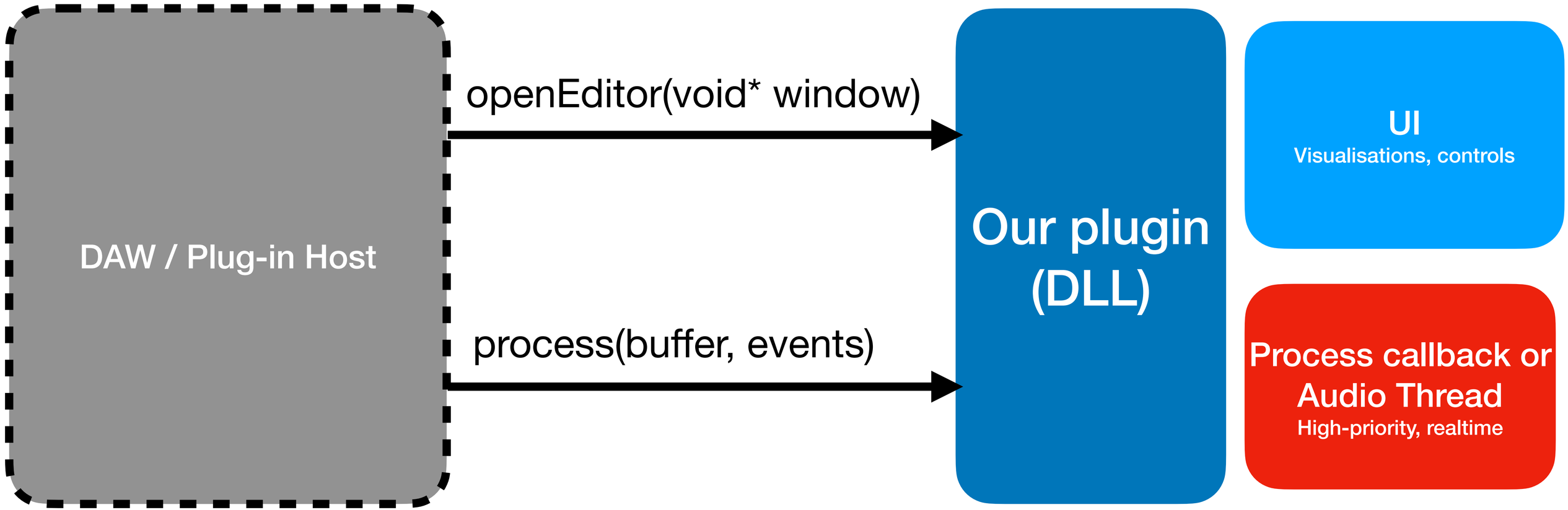
https://github.com/realtime-sanitizer/radsan

Topics

- Digital audio basics
- Real-time constraints and Rust
- Architecture of audio/music software
- UI Development

Architecture of audio/music software

Audio, UI and MIDI



Only wanted to implement “process"

```
let default_host = default_host();
let output_device = default_host
    .default_output_device()
    .expect("No output device");

let stream_config = cpal::StreamConfig {
    channels: 2,
    sample_rate: cpal::SampleRate(44100),
    buffer_size: cpal::BufferSize::Default,
};

let stream = output_device
    .build_output_stream(
        &stream_config,
        process,
        |err| eprintln!("An error occurred on stream: {}", err),
        None,
    )
    .expect("Failed to build output stream");

stream.play().expect("Failed to play stream");
```

audio-processor-traits

```rust
pub trait AudioProcessor {
    type SampleType: Sized;

    /// Prepare for playback based on current audio settings
    fn prepare(&mut self, _context: &mut AudioContext) {}

    /// Process a block of samples by mutating the input `AudioBuffer`
    fn process(&mut self, _context: &mut AudioContext, data: &mut AudioBuffer<Self::SampleType>);
}
```

https://beijaflor.io/blog/02-2022/rust-audio-experiments-4/

audio-processor-traits

```rust
fn main() {
    let processor = SimpleDelayProcessor::new();
    audio_processor_standalone::audio_processor_main(processor);
}
```

```
audio-processor-standalone

USAGE:
my-crate [OPTIONS]

FLAGS:
-h, --help       Prints help information
-V, --version    Prints version information

OPTIONS:
-i, --input-file <INPUT_PATH>              An input audio file to process
--midi-input-file <MIDI_INPUT_FILE>    If specified, this MIDI file will be passed through the processor
-o, --output-file <OUTPUT_PATH>            If specified, will render offline into this file (WAV)
```

https://beijaflor.io/blog/02-2022/rust-audio-experiments-4/

audio-processor-traits

```
fn main() {
    let processor = SimpleDelayProcessor::new();
    let handles = audio_processor_standalone::audio_processor_run(…);
    // …
}
```

https://beijaflor.io/blog/02-2022/rust-audio-experiments-4/

audio-processor-traits

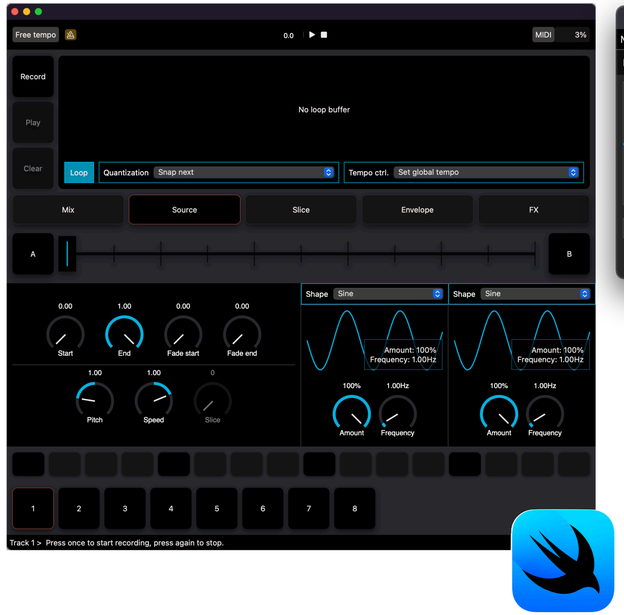
```
#[derive(Default, audio_processor_traits_derive::AudioProcessorHandle)]
struct GainProcessorHandle {
    #[parameter(name = "Value 1")]
    value1: AtomicF32,
    #[parameter(name = "Value 2", min = 30.0, max = 60.0)]
    value2: AtomicF32,
}

let handle = make_shared(GainProcessorHandle {});

pub trait AudioProcessorHandleProvider { }
pub trait AudioProcessorHandle: Send + Sync { }
```

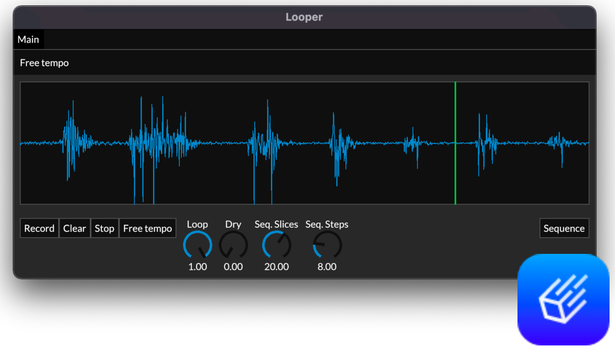
https://beijaflor.io/blog/02-2022/rust-audio-experiments-4/

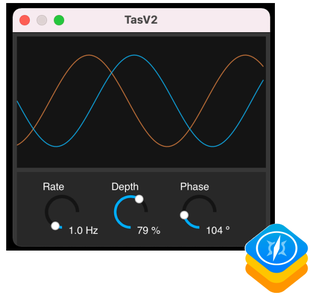
Topics

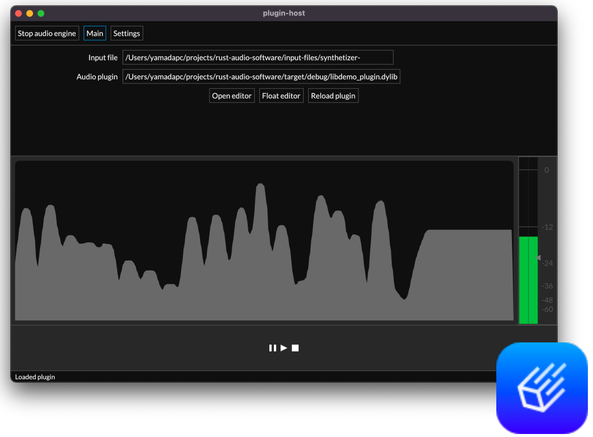
- Digital audio basics
- Real-time constraints and Rust
- Architecture of audio/music software
- UI Development

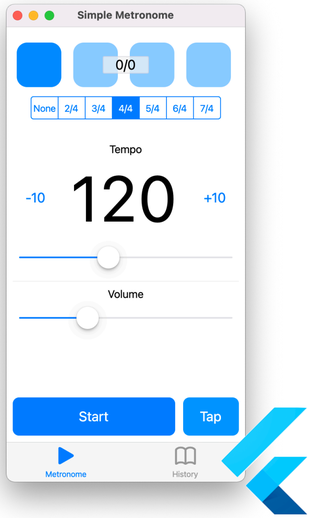
UI Development





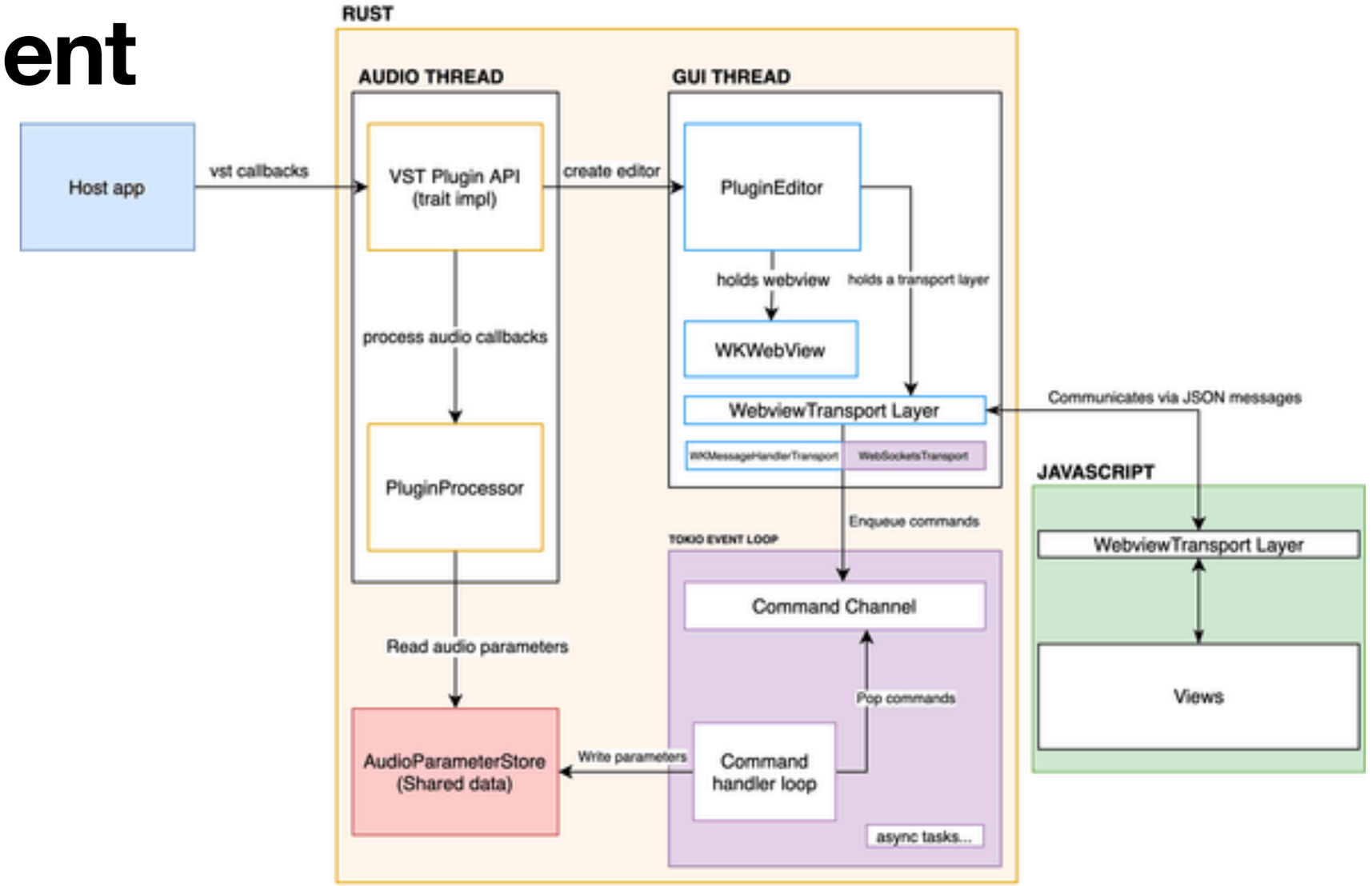






UI Development

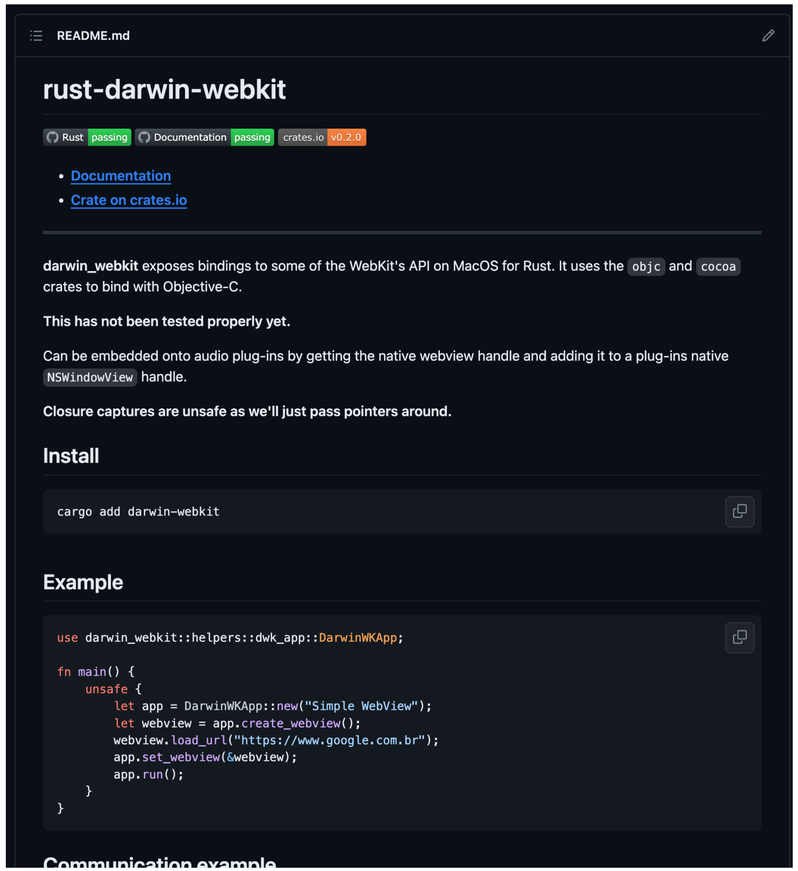
Web UI



https://beijaflor.io/blog/07-2021/rust-audio-experiments-2/

UI Development

Web UI



README.md

rust-darwin-webkit

Rust passing Documentation passing crates.io v0.2.0

- Documentation
- Crate on crates.io

darwin_webkit exposes bindings to some of the WebKit's API on MacOS for Rust. It uses the `objc` and `cocoa` crates to bind with Objective-C.

This has not been tested properly yet.

Can be embedded onto audio plug-ins by getting the native webview handle and adding it to a plug-ins native `NSWindowView` handle.

Closure captures are unsafe as we'll just pass pointers around.

Install

```
cargo add darwin-webkit
```

Example

```
use darwin_webkit::helpers::dwk_app::DarwinWKApp;

fn main() {
    unsafe {
        let app = DarwinWKApp::new("Simple WebView");
        let webview = app.create_webview();
        webview.load_url("https://www.google.com.br");
        app.set_webview(&webview);
        app.run();
    }
}
```

Communication example

- Wrapper for Webkit on macOS/iOS only, works in a hosted environment
- Can’t use wry because it owns the event-loop
- The "cacao" has very usable wrappers, but also comes with its own abstractions for objective-c interop.

https://github.com/yamadapc/rust-darwin-webkit

UI Development

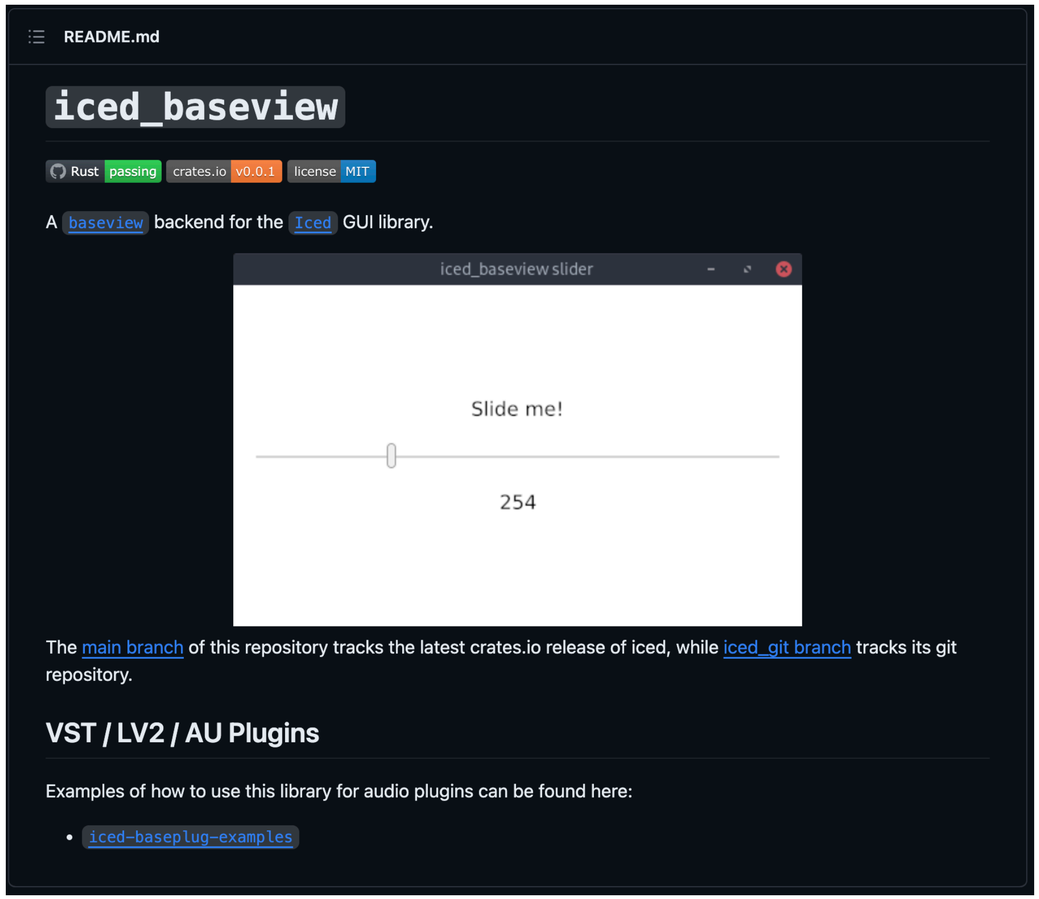
Web UI

- Output presented how they are doing this with JUCE & C++ this year*
- There have been other similar presentations previously on ADC

https://output.com/products/arcade

UI Development

Iced

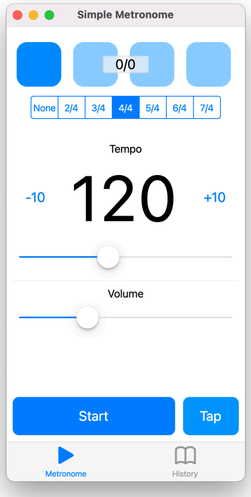


- Unfortunately plug-in UI libraries need to duplicate a lot of general crate code, because rust windowing crates assume ownership/ticking of the event-loop and thus can’t be used in a DAW host context

https://github.com/BillyDM/iced_baseview

UI Development

Flutter / SwiftUI / Skia



- swiftUI and flutter_rust_bridge apps that can be used as examples
- SwiftUI front-end uses Skia for progressive waveform drawing on a background thread

https://beijaflor.io/blog/04-2022/rust-audio-experiments-5/

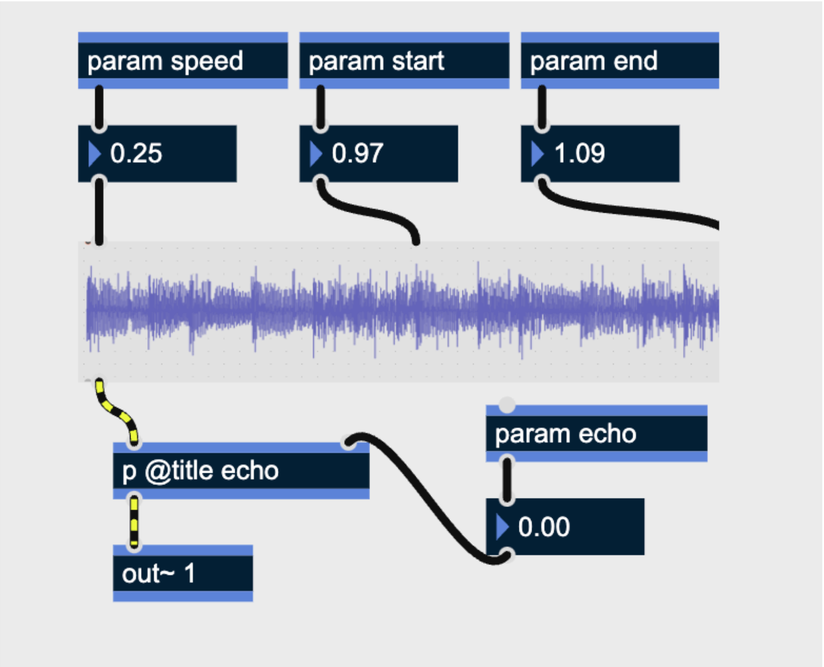
https://beijaflor.io/blog/01-2022/rust-audio-experiments-3/

Wrap-up

- Realtime constraints - a lot of extra work, can simplify code significantly with message passing (ringbuf) & reference counting primitives in basedrop/ audio_garbage_collector
- cpal & midir for standalone
- vst & other crates for plug-ins
- To have generic processors, need abstractions;
  - there is another project called 'nih_plug' which looks great; there was also a talk on ADC-2023 about DSP combinators in Rust
  - augmented-audio is there, but it is very much a study project in a mono-repo

Wrap-up

- Consider using C++ and JUCE at least for the wrapper code as it solves lots of packaging issues that I haven’t discussed in a production ready manner
- For the DSP code itself, there are great DSLs/environments that make things easier
  - Max MSP + RNBO
  - CMajor
  - Faust



Thanks!

beijaflor.io

github.com/yamadapc/augmented-audio

Rust Audio/Music Programming

Pedro Tacla Yamada - 05/12/2023