

Behind CyberEther:

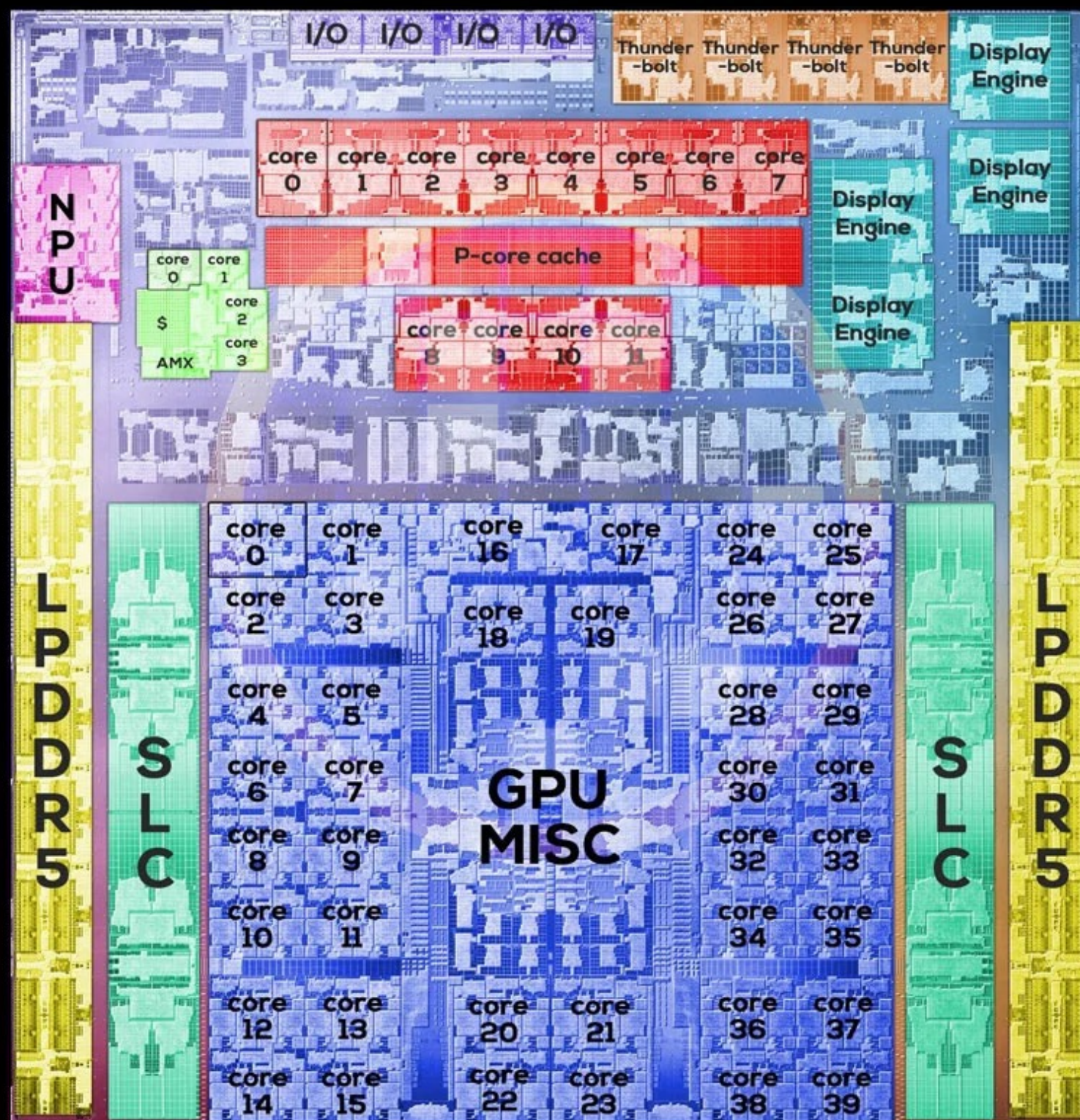
Bridge to Portable GPU Accelerated Interface

Luigi Cruz ([@luigifcruz](https://twitter.com/luigifcruz))

GNU Radio Conference 2024 - Knoxville, TN

CyberEther

Multi-platform GPU-accelerated interface for compute-intensive pipelines.



Apple M3 Max

- Modern computers are heterogeneous.
- Only a fraction of the silicon is dedicated to the CPU.
- Modern software has to take advantage of the entire silicon.

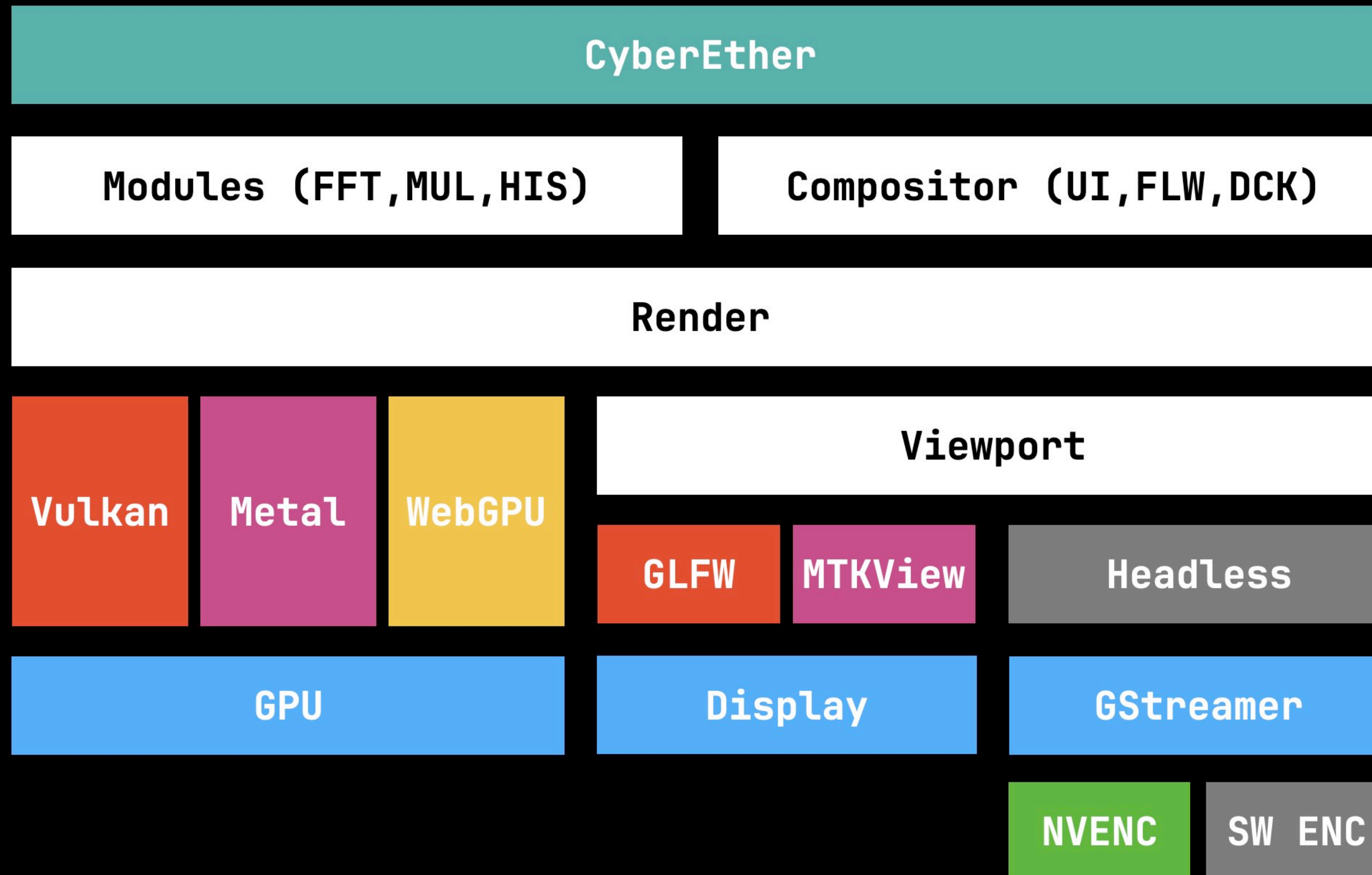
CyberEther

Multi-platform GPU-accelerated interface for compute-intensive pipelines.

- 🗑 Custom Render for graphics built with **Vulkan**, **Metal**, and **WebGPU**.
- 🐙 Compute scheduler supporting **Vulkan**, **Metal**, and **WebGPU**.
- 🧊 Native browser support with **WebAssembly** and **WebGPU**.
- 🤖 Low-latency headless remote interface.
- 🌟 **Superluminal** high-level real-time visualization library for Python and C++.
- 👁 MIT Licensed

CyberEther

Custom Graphical Render



- Abstracts away state-of-the-art low-level graphical frameworks such as **Vulkan**, **Metal**, and **WebGPU**.
- Abstracts away OS viewport creation GLFW (Linux, Windows, macOS), MTKView (iOS), and Headless.
- Supports advanced features:
 - Font rendering with SDF.
 - Instanced rendering.
 - Multi-draw Indirect.
- Freedom from proprietary or stuck-in-time rendering interfaces.

CyberEther

Low-Level Graphical API Abstraction

Jetstream Render

Vulkan

- Linux
- Windows
- Android

Metal

- macOS
- iOS, iPadOS, tvOS

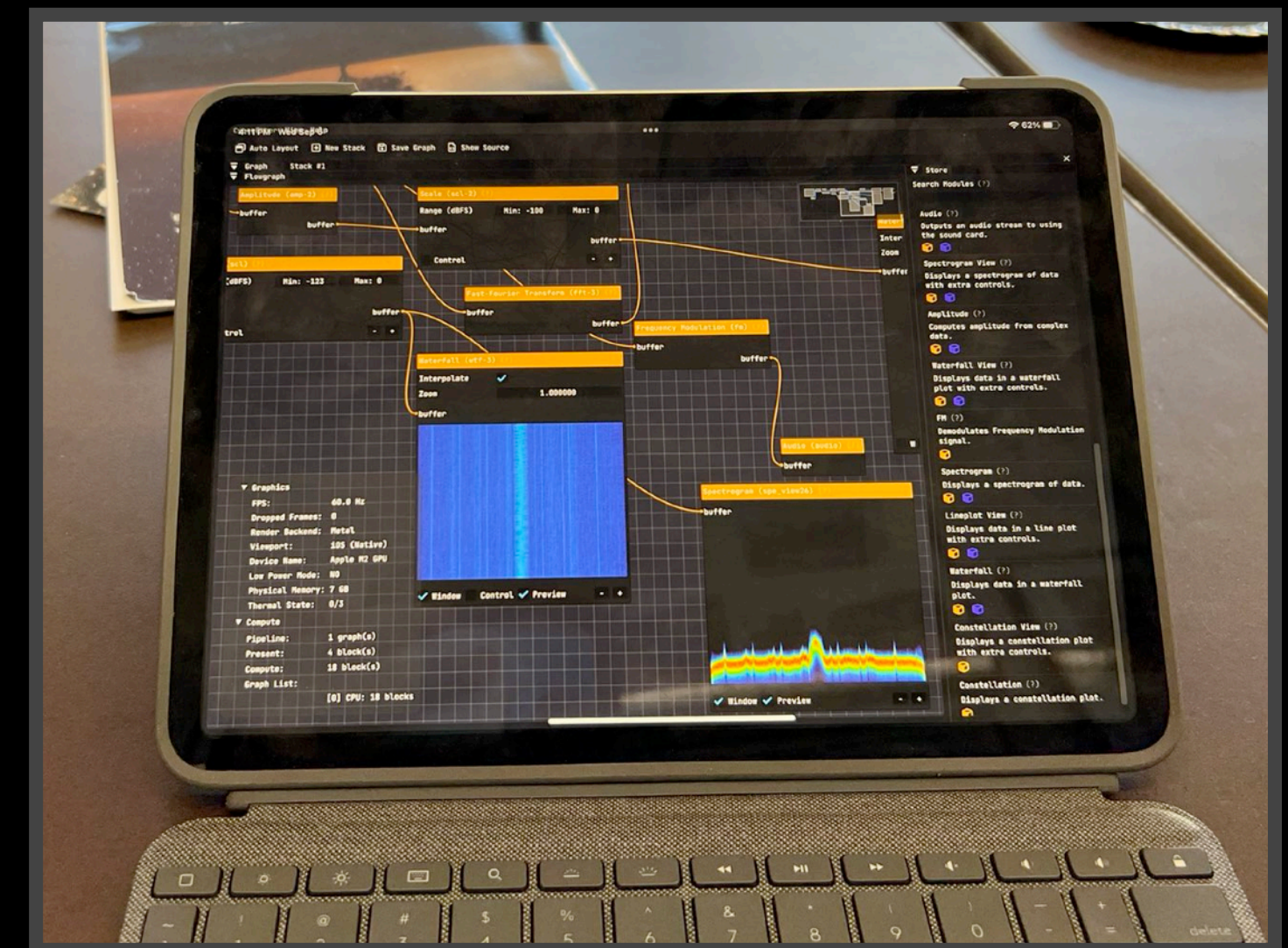
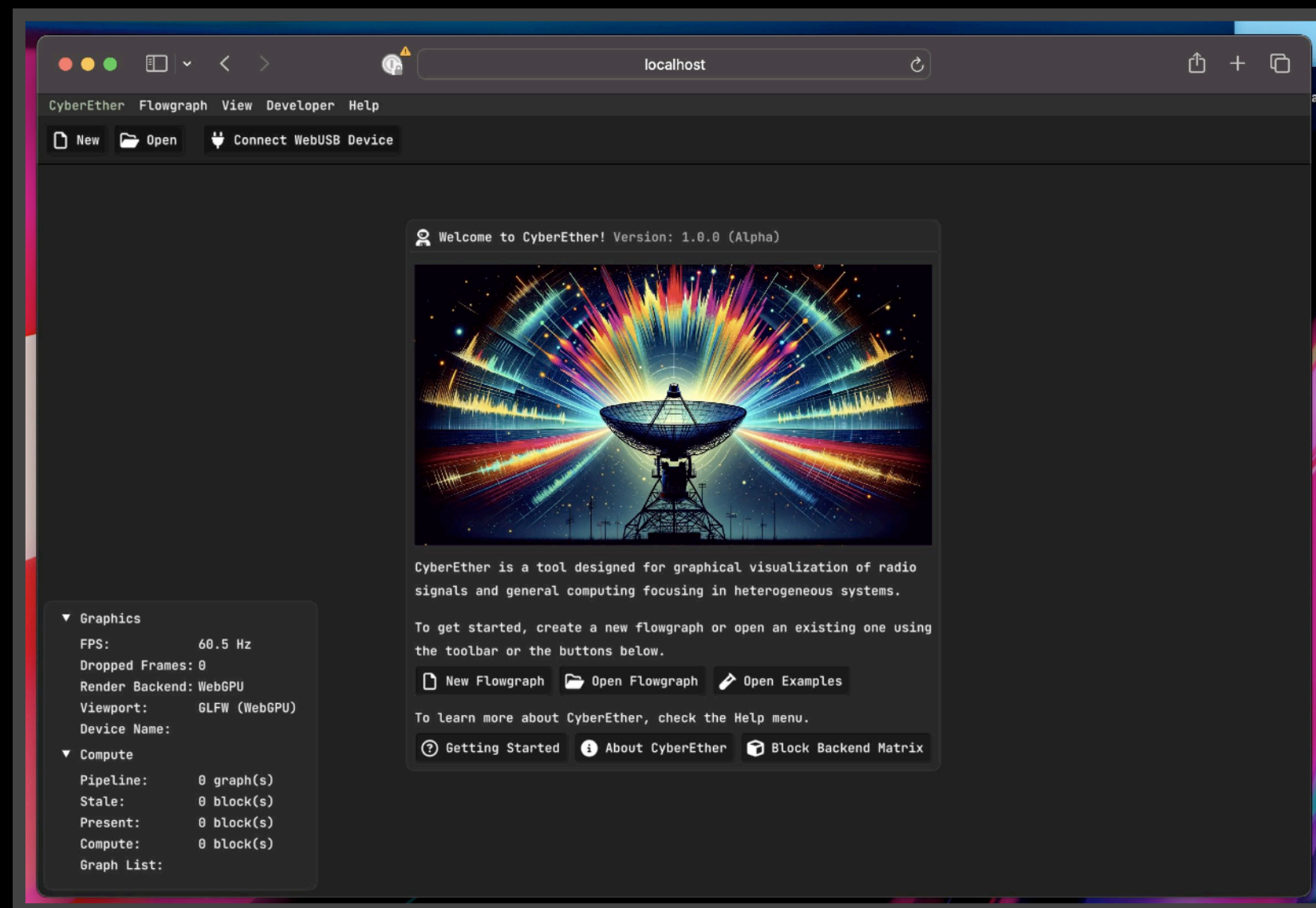
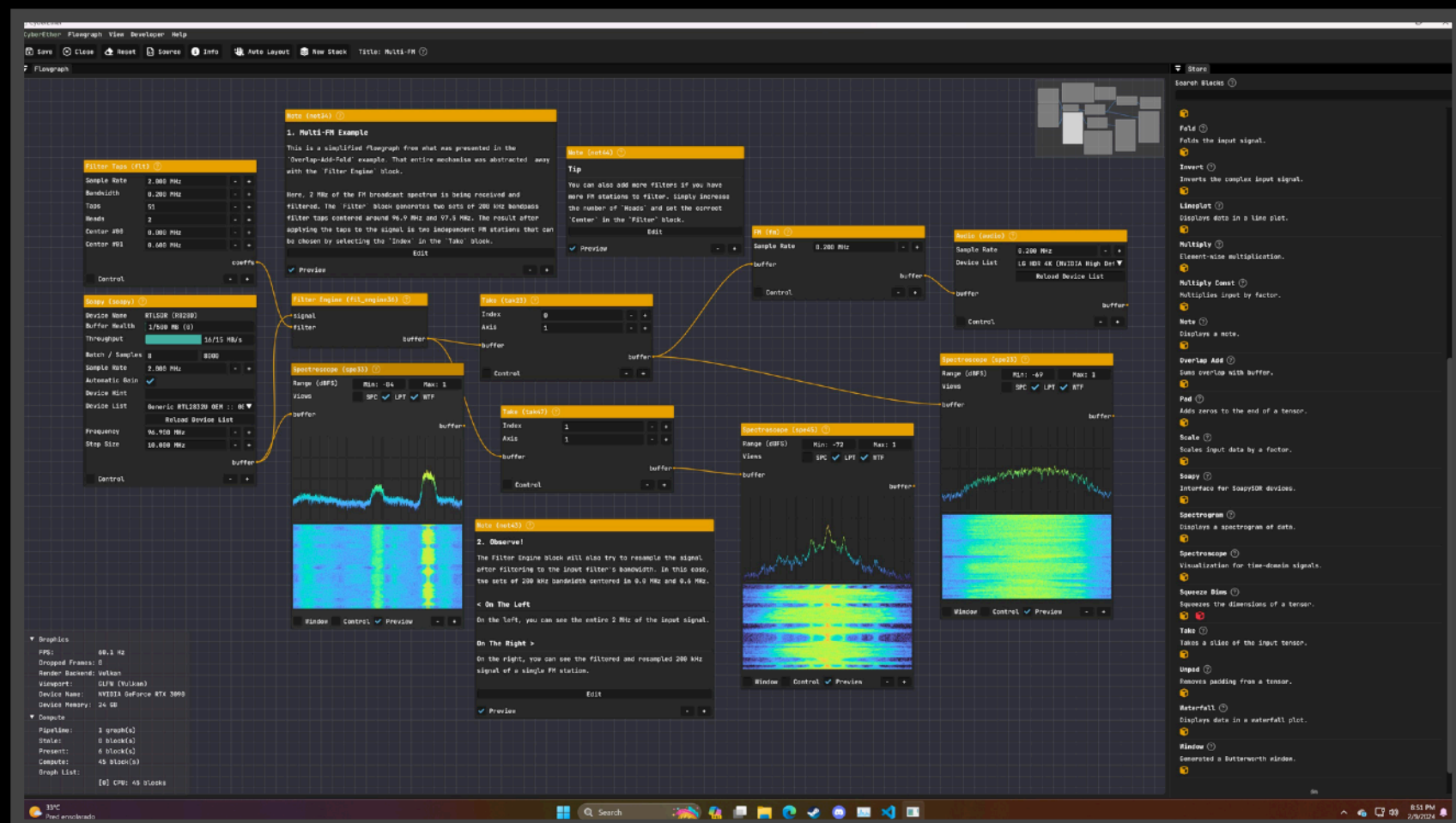
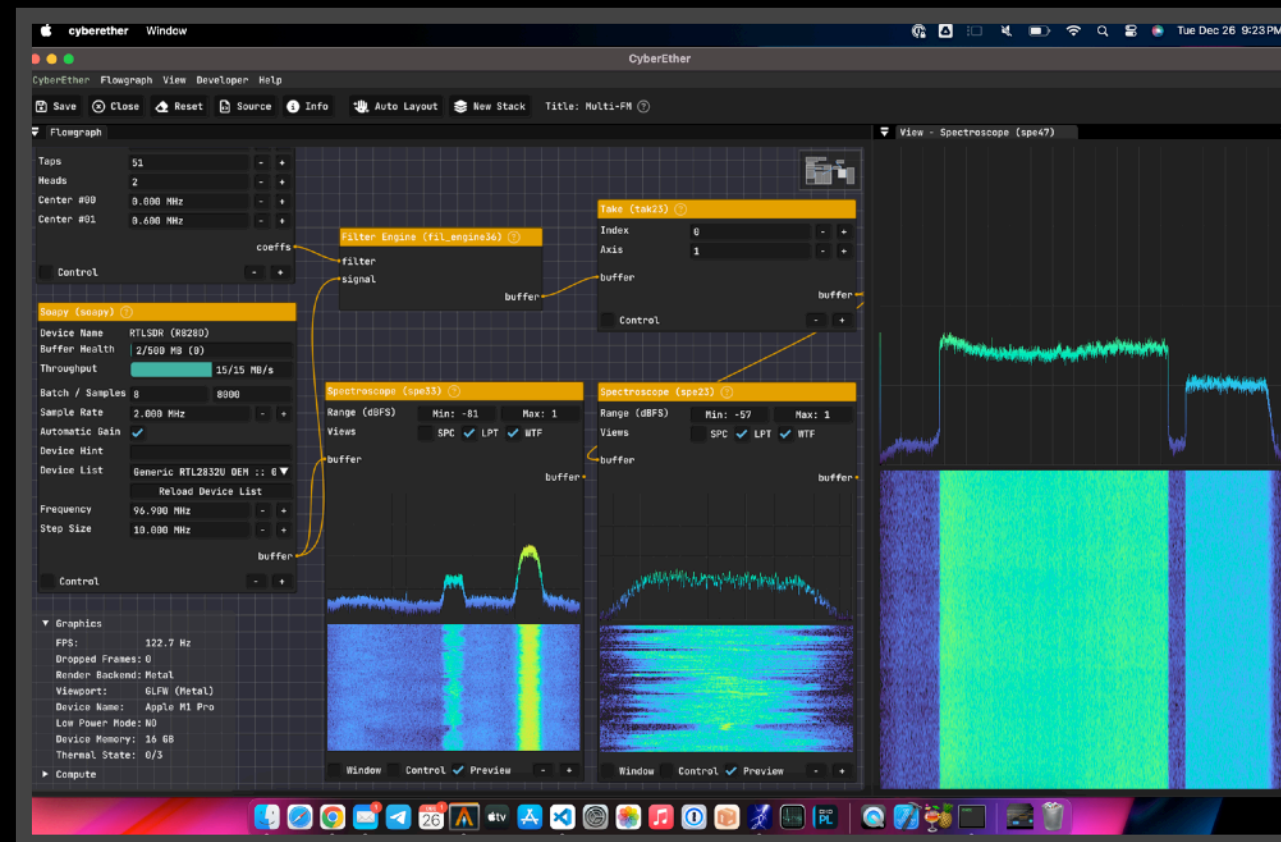
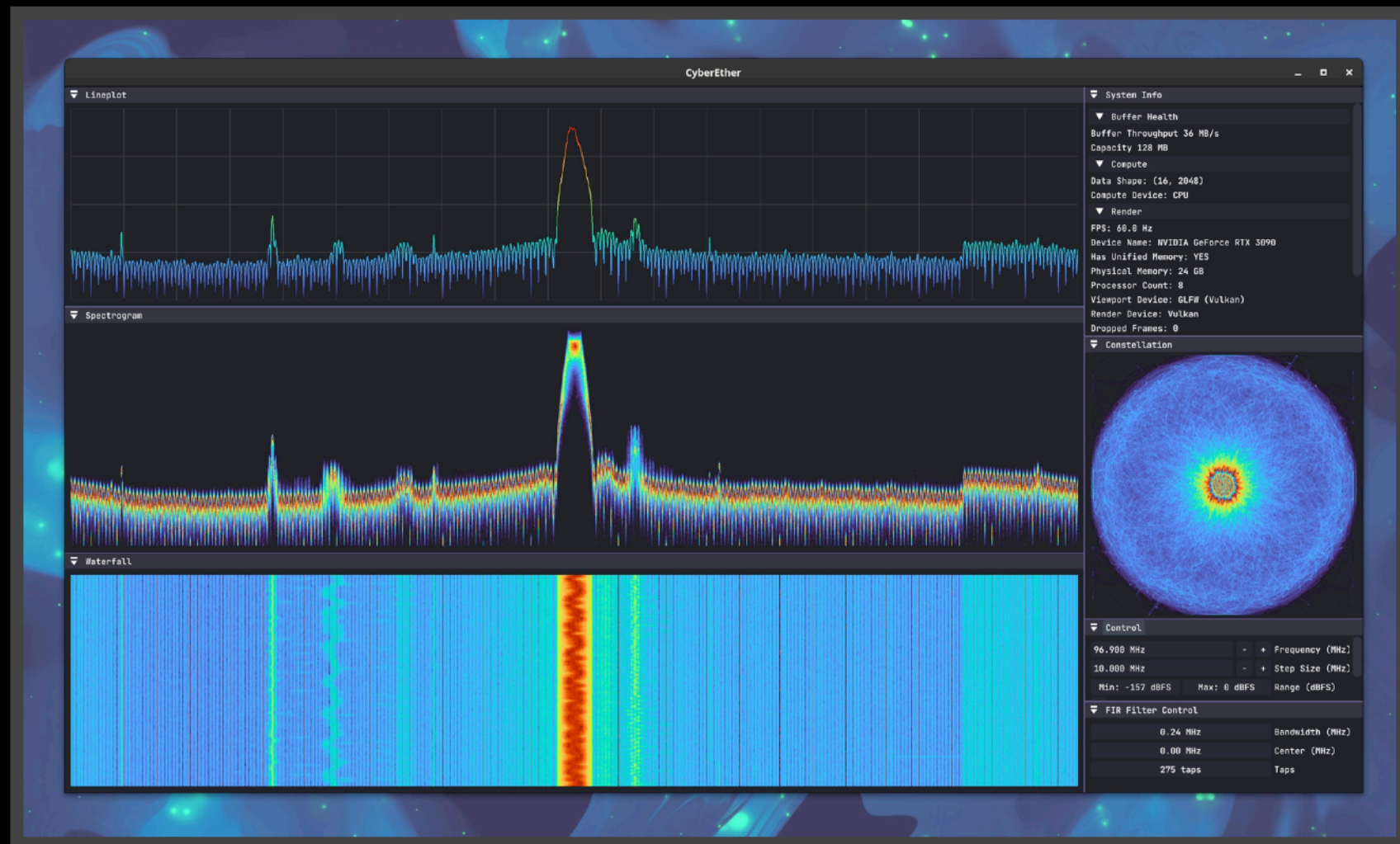
WebGPU

- Chrome
- Safari Technology Preview
- Firefox Nightly

CyberEther

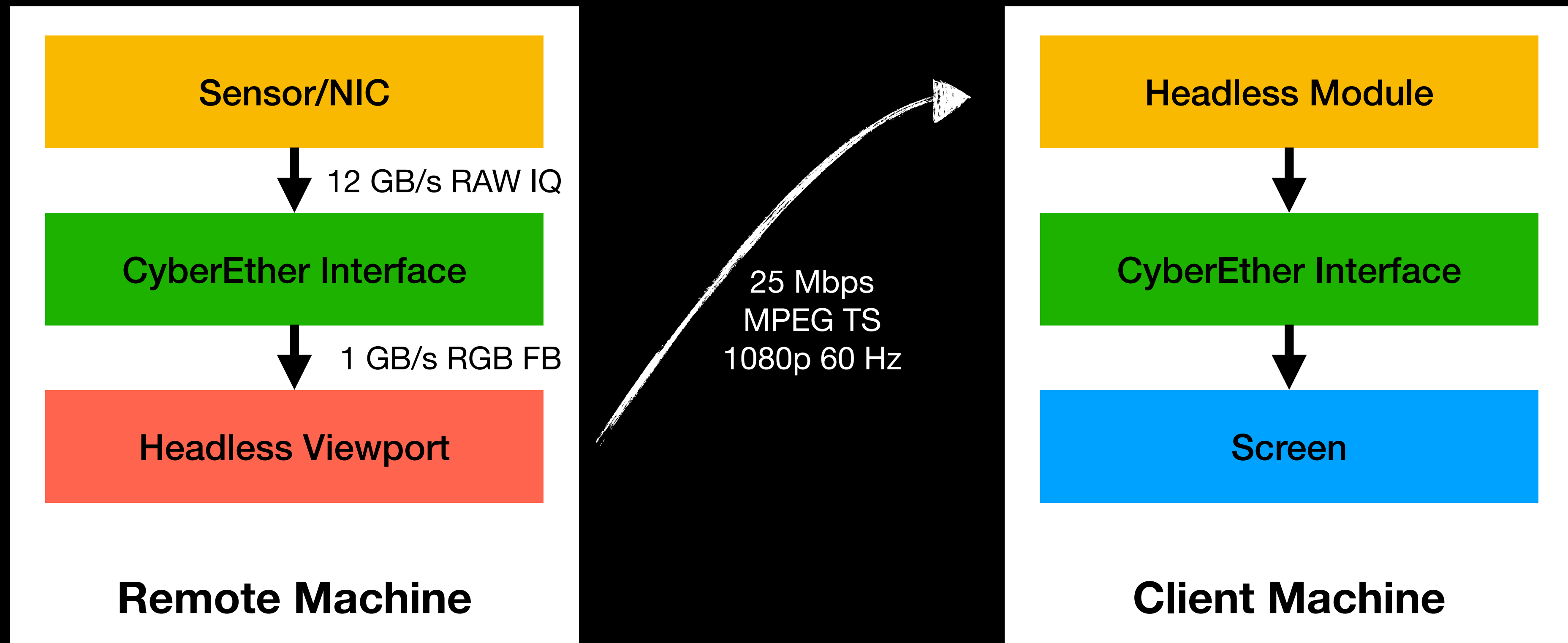
It Runs Everywhere TM

- ✓ Runs on a **Linux** machine.
- ✓ Runs on a **Windows** machine.
- ✓ Runs on a **Mac** machine.
- ✓ Runs on **Chrome** or **Safari**.
- ✓ Runs on an **iPhone**.
- ✓ Runs on an **iPad**.
- ✓ Runs on a **Meta Quest 3?**



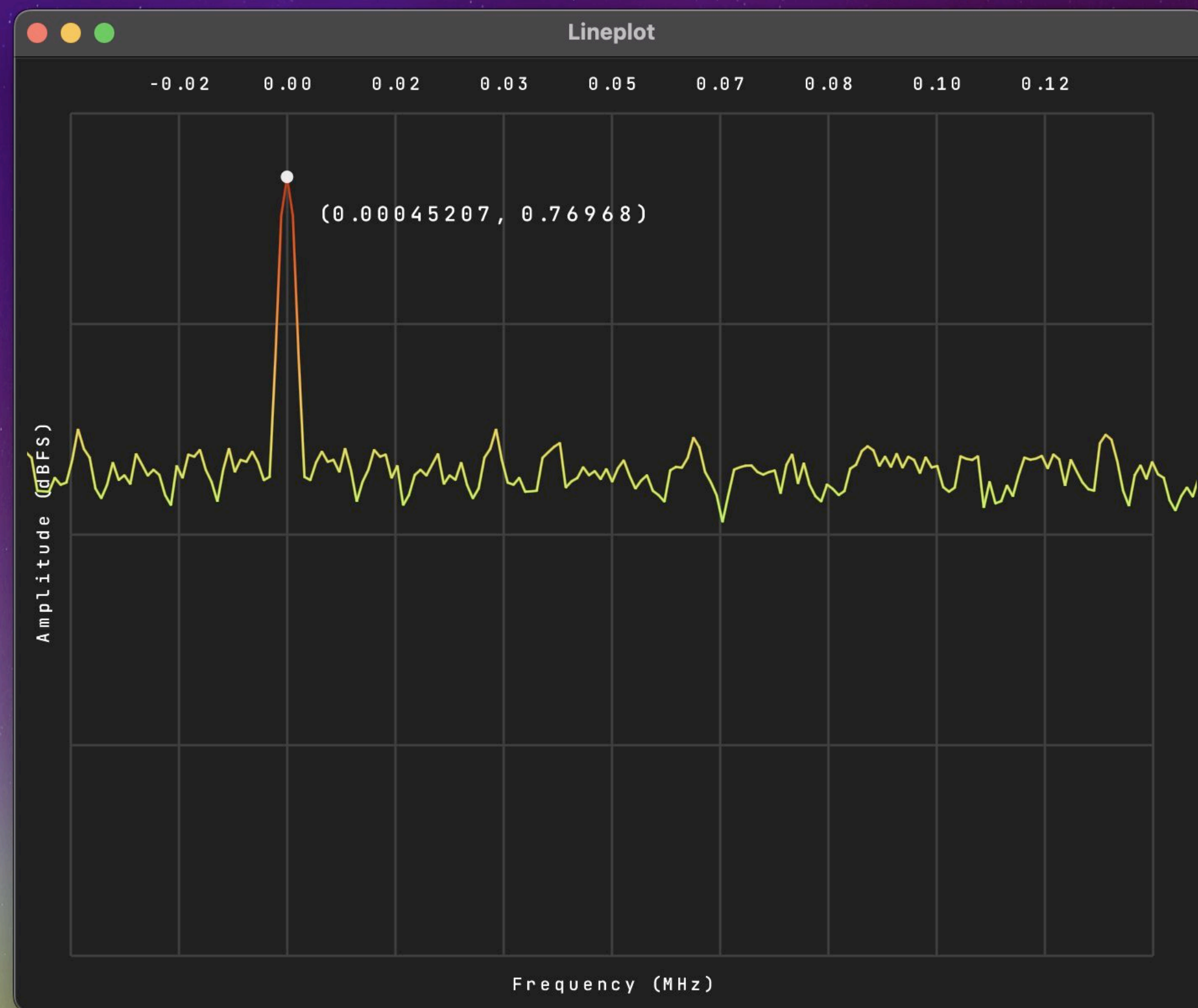
CyberEther

Low-Latency Headless Remote Interface



CyberEther

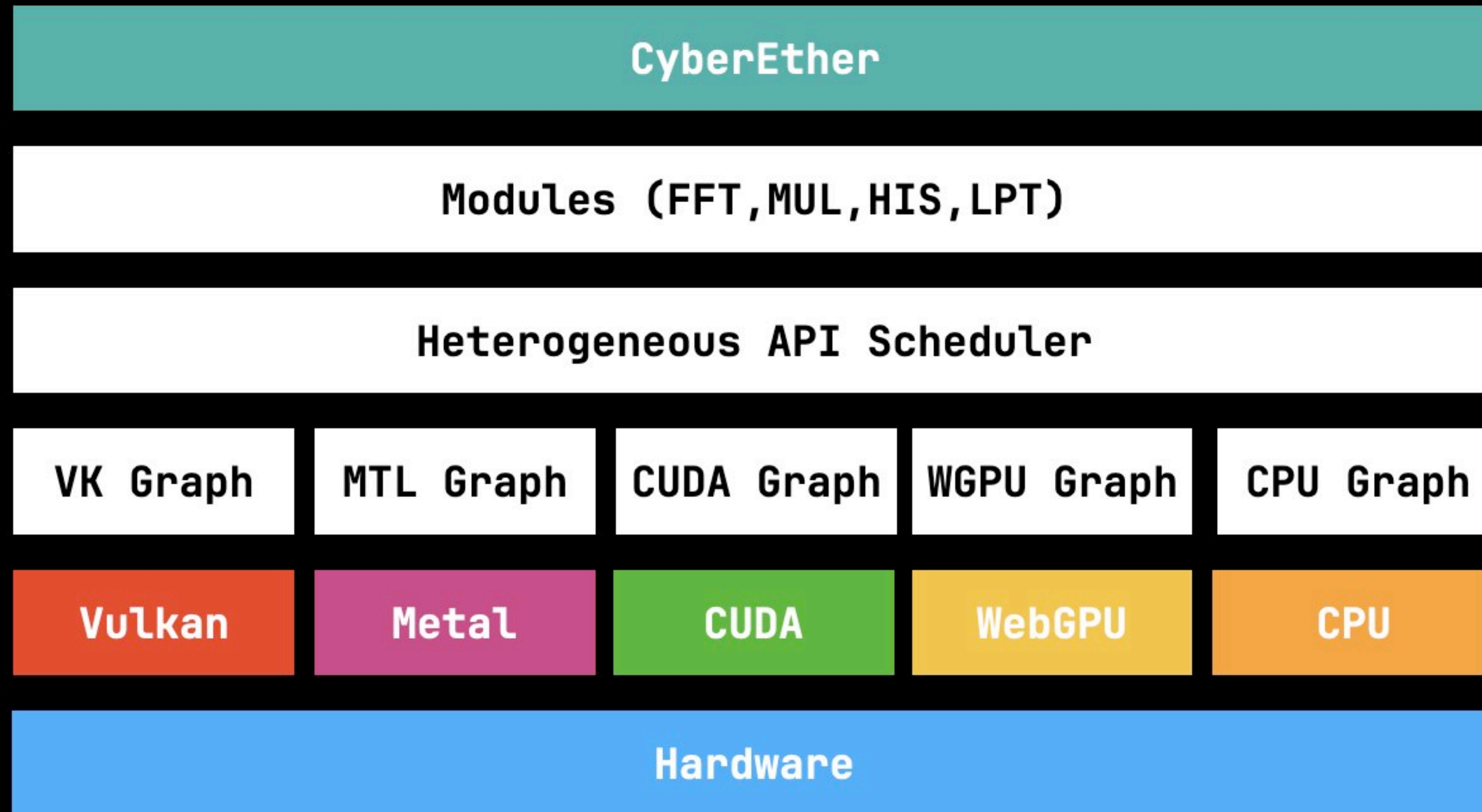
Advanced Render Features



- Thick Lines offloaded to compute shader.
- Font and Lines with Signed Distance Field.
 - Great anti-aliasing.
- Instancing support.
 - Multiple versions of the same object.
- Multi-draw Indirect support.
 - Multiple objects sharing a single buffer.
- Single GLSL shader for all backends.
 - Translated using SPIR-V as intermediate.

CyberEther

Heterogeneous Compute Scheduler



- No vendor lock-in.
- Support to all major compute APIs.
- Default asynchronous instantiation of compute kernels.
- Synchronization between APIs transparently handled by the Scheduler.
- Transparent memory sharing between API and devices.

CyberEther

Seamless Memory Sharing

```
Tensor<Device::CUDA, F32> storage({8192});

REQUIRE(storage.device() == Device::CUDA);
REQUIRE(storage.root_device() == Device::CUDA);
REQUIRE(storage.references() == 1);

Tensor<Device::Vulkan, F32> cloned_storage(storage);

REQUIRE(cloned_storage.device() == Device::Vulkan);
REQUIRE(cloned_storage.root_device() == Device::CUDA);
REQUIRE(cloned_storage.references() == 2);

// OR THE OTHER WAY AROUND

Tensor<Device::Vulkan, F32> storage({8192});
Tensor<Device::CUDA, F32> cloned_storage(storage);

// Jetstream implements CUDA low-level
// Virtual Memory Management (VMM) behind the scenes.
// No user interaction with the API.
```

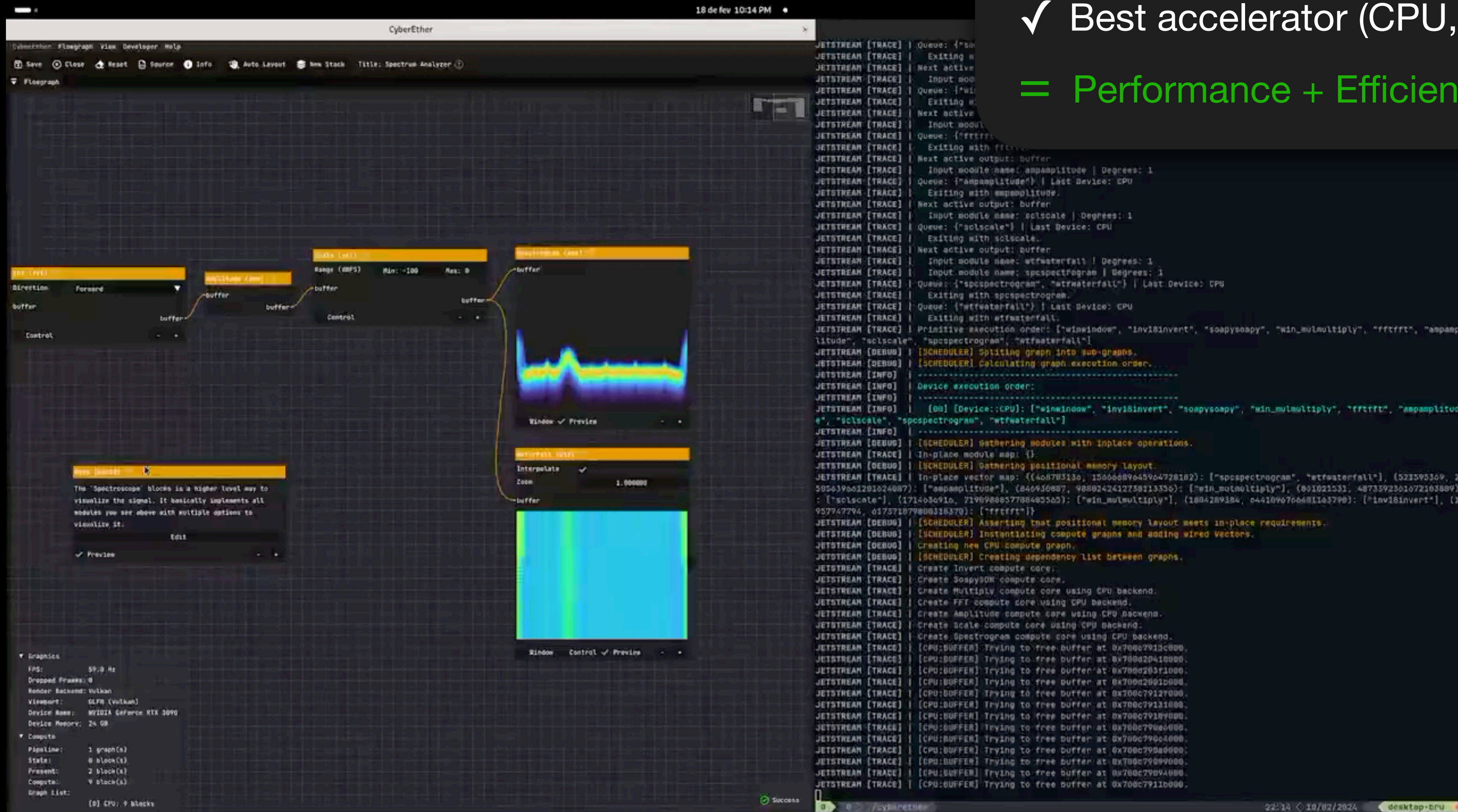
- Zero-copy.
- No wasted time copying data.
- Wraps low-level interoperability API.
- Same memory. Multiple API.
- Possible paths:
 - Vulkan ↔ CPU
 - Vulkan ↔ CUDA
 - CUDA ↔ CPU

CyberEther

Heterogeneous Flowgraph

- ✓ Automatic memory sharing between APIs.
- ✓ Automatic synchronization between APIs.
- ✓ Best accelerator (CPU, GPU, etc) per task.

= Performance + Efficiency



DEMOS

Welcome to CyberEther! Version: 1.0.0



CyberEther is a tool designed for graphical visualization of radio signals and general computing focusing in heterogeneous systems.

To get started, create a new flowgraph or open an existing one using the toolbar or the buttons below.

New Flowgraph Open Flowgraph Open Examples

To learn more about CyberEther, check the Help menu.

Getting Started About CyberEther Block Backend Matrix

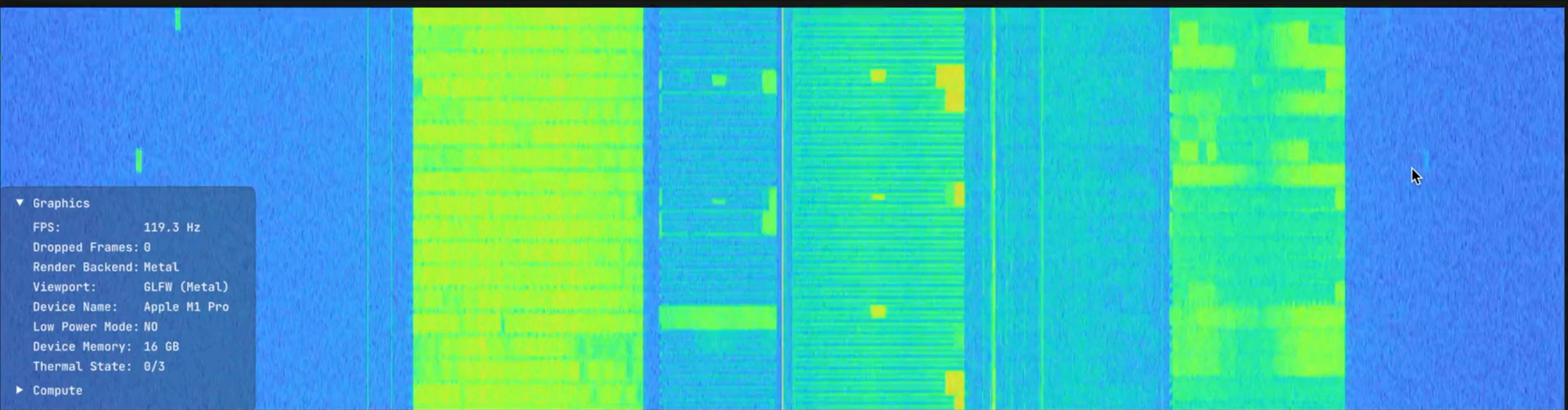
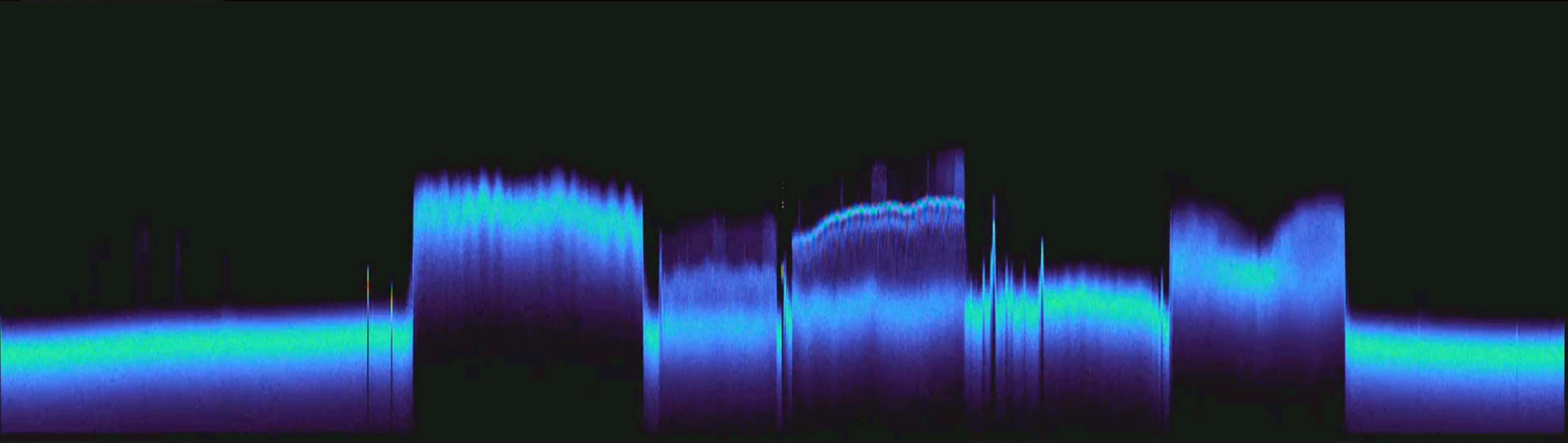
▼ Graphics

FPS: 120.4 Hz
Dropped Frames: 0
Render Backend: Metal
Viewport: GLFW (Metal)
Device Name: Apple M1 Pro
Low Power Mode: NO
Device Memory: 16 GB
Thermal State: 0/3

▼ Compute

Pipeline: 0 graph(s)
Stale: 0 block(s)
Present: 0 block(s)
Compute: 0 block(s)
Graph List:

✓ Success



▼ Graphics

FPS: 119.3 Hz

Dropped Frames: 0

Render Backend: Metal

Viewport: GLFW (Metal)

Device Name: Apple M1 Pro

Low Power Mode: NO

Device Memory: 16 GB

Thermal State: 0/3

► Compute

Welcome to CyberEther! Version: 1.0.0



CyberEther is a tool designed for graphical visualization of radio signals and general computing focusing in heterogeneous systems.

To get started, create a new flowgraph or open an existing one using the toolbar or the buttons below.

New Flowgraph Open Flowgraph Open Examples

To learn more about CyberEther, check the Help menu.

Getting Started About CyberEther Block Backend Matrix

▼ Graphics

FPS: 119.0 Hz
Dropped Frames: 0
Render Backend: Metal
Viewport: GLFW (Metal)
Device Name: Apple M1 Pro
Low Power Mode: NO
Device Memory: 16 GB
Thermal State: 0/3

▼ Compute

Pipeline: 0 graph(s)
Stale: 0 block(s)
Present: 0 block(s)
Compute: 0 block(s)
Graph List:

✓ Success

CyberPunk2077
Bauru
23°
Mostly Cloudy
H:34° L:20°
2 AM 3 AM 4 AM 5 AM 5:32 AM 6 AM
23° 22° 21° 21° 20° 20°
Thu 18° 36°
Fri 22° 37°
Sat 22° 32°
Sun 20° 29°
Mon 20° 28°
WEDNESDAY 27
No events today
TOMORROW
VLA DSP 3-4PM
HCRO Meeting 8:00-9:30PM
Maps Camera App Store Settings
YouTube Freeform Drive Notion X Telegram
Safari Music Calendar Notes Mail Photos App Store X



REDACTED

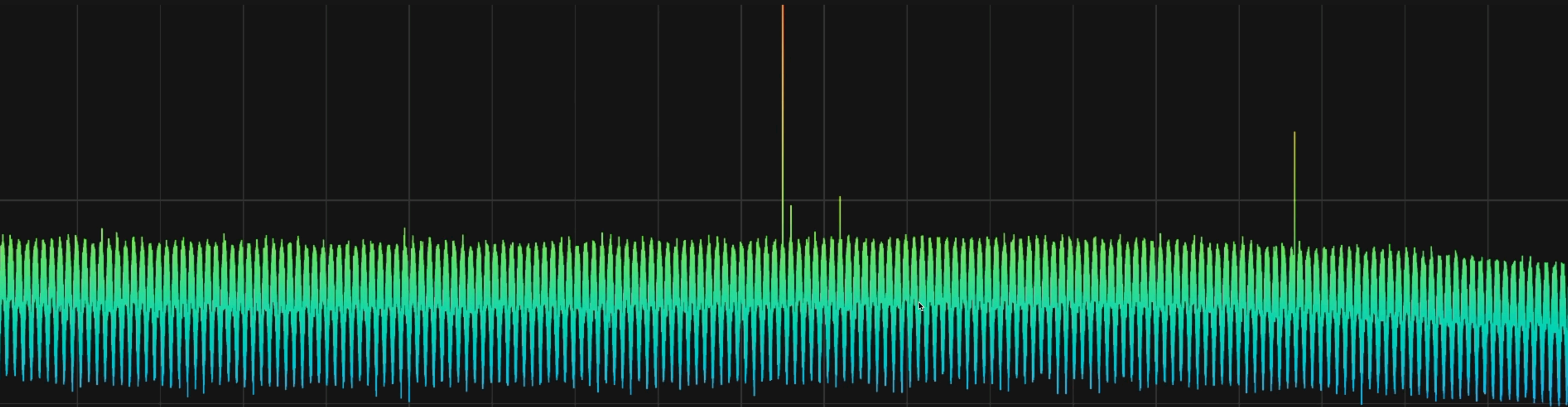
Remote Visualization

Allen Telescope Array Holoscan Pipeline



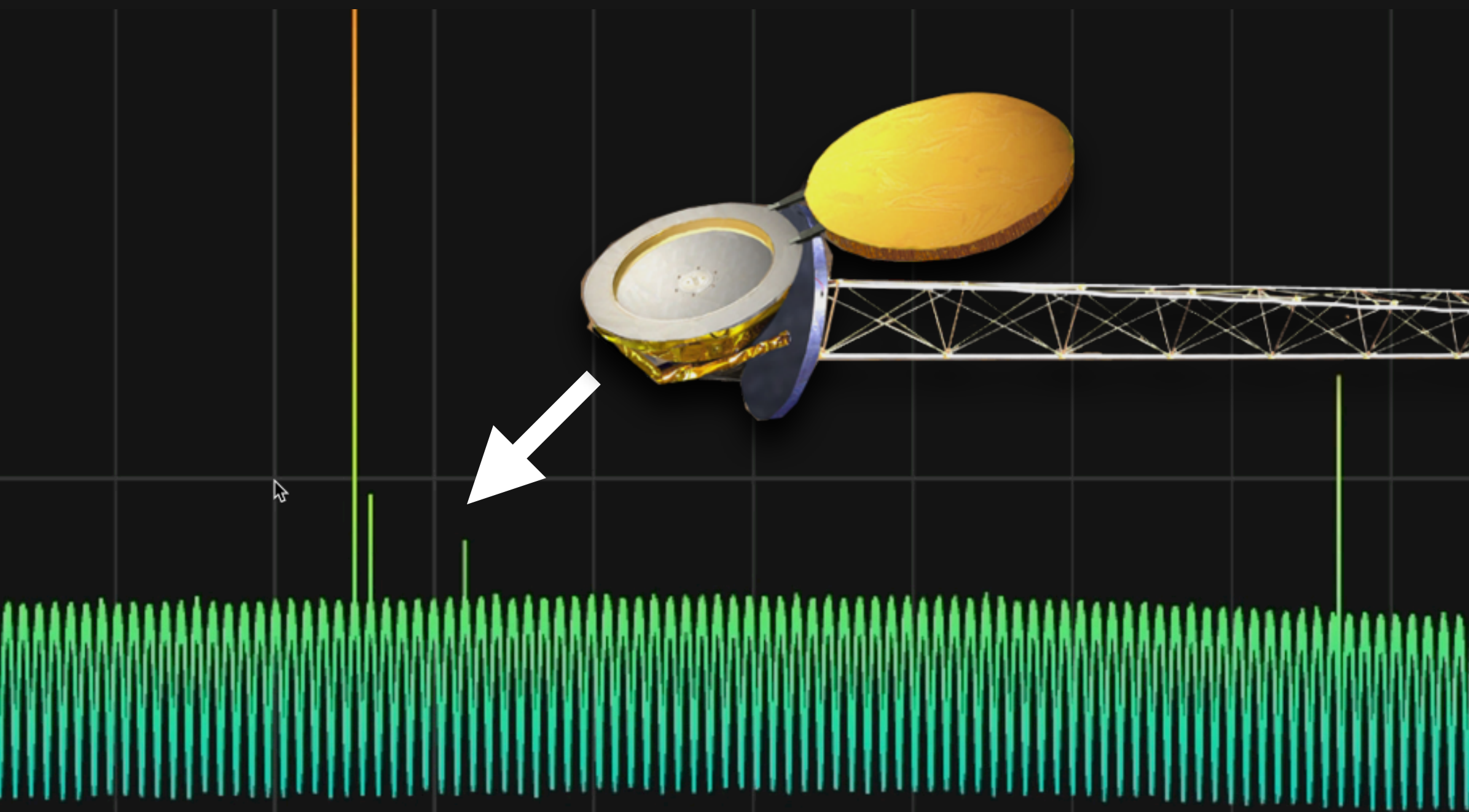
Deep Space Demonstration

Allen Telescope Array Holoscan Pipeline



Deep Space Demonstration

Allen Telescope Array Holoscan Pipeline

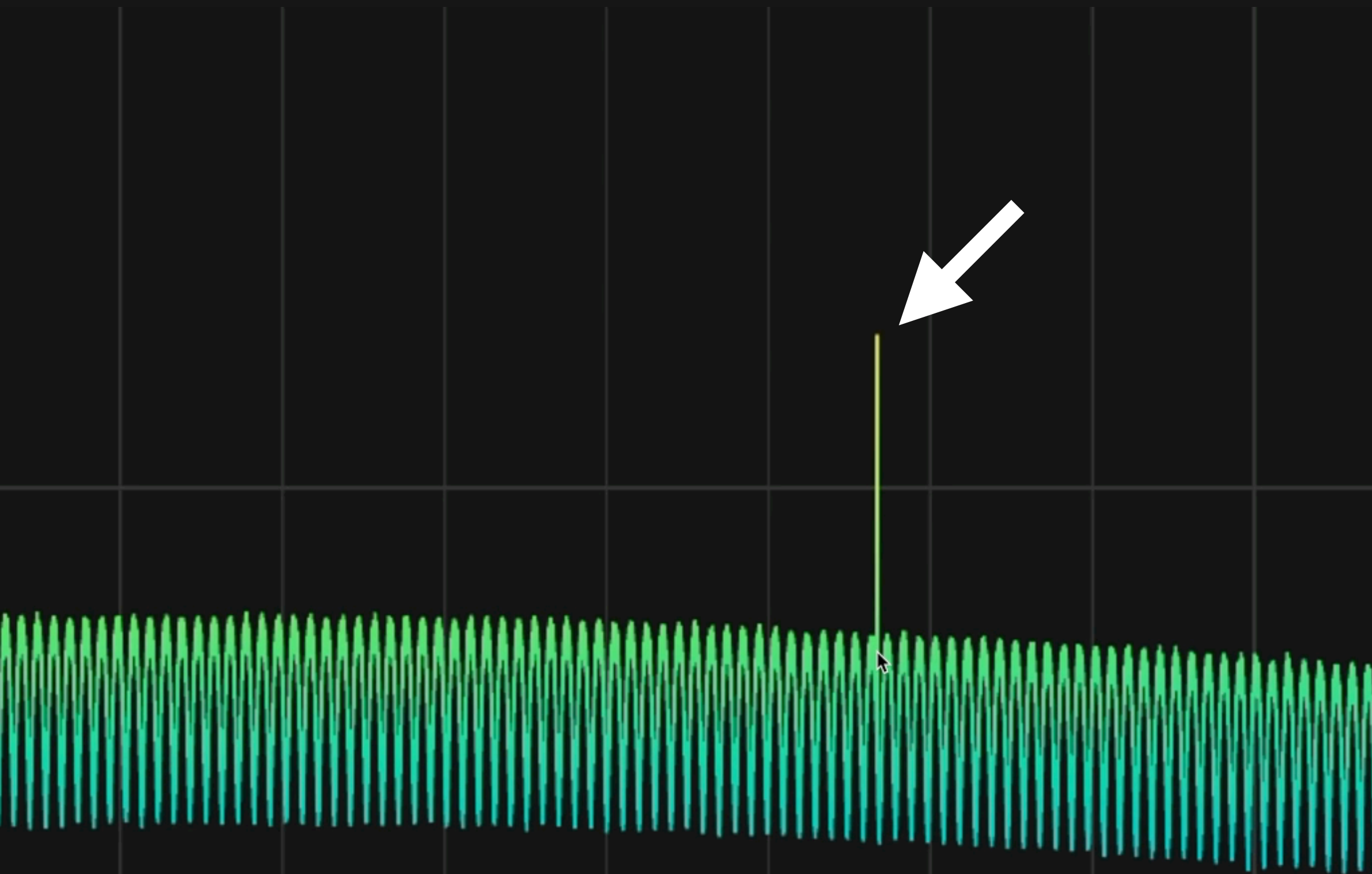


~8406 MHz

Mars Odyssey

Deep Space Demonstration

Allen Telescope Array Holoscan Pipeline



~8430 MHz

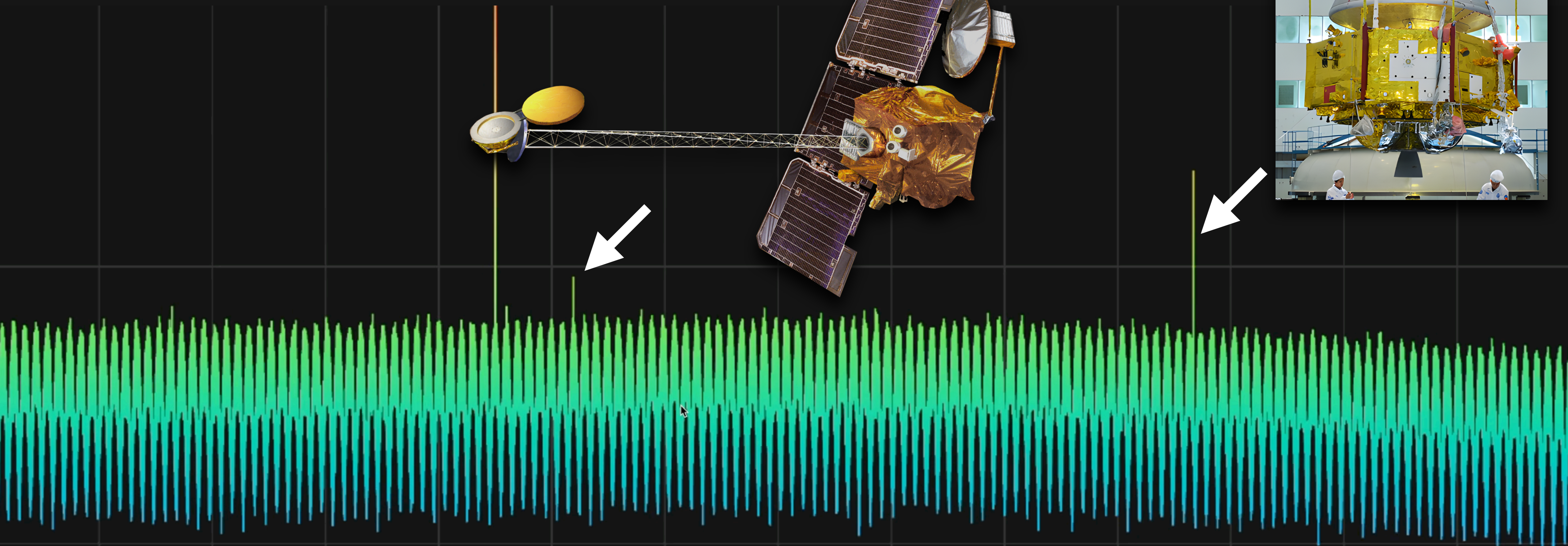
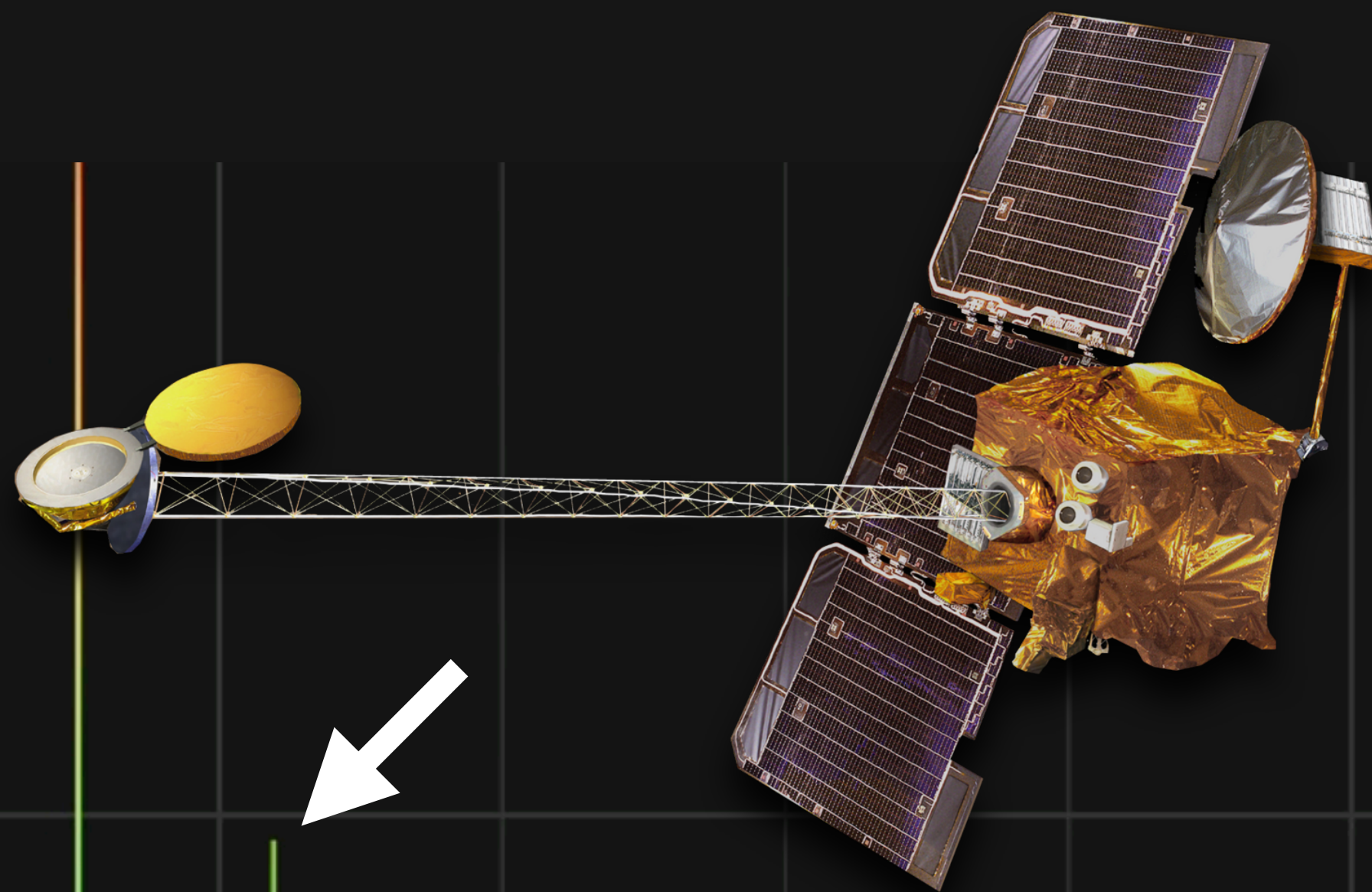


Tianwen

天问一号

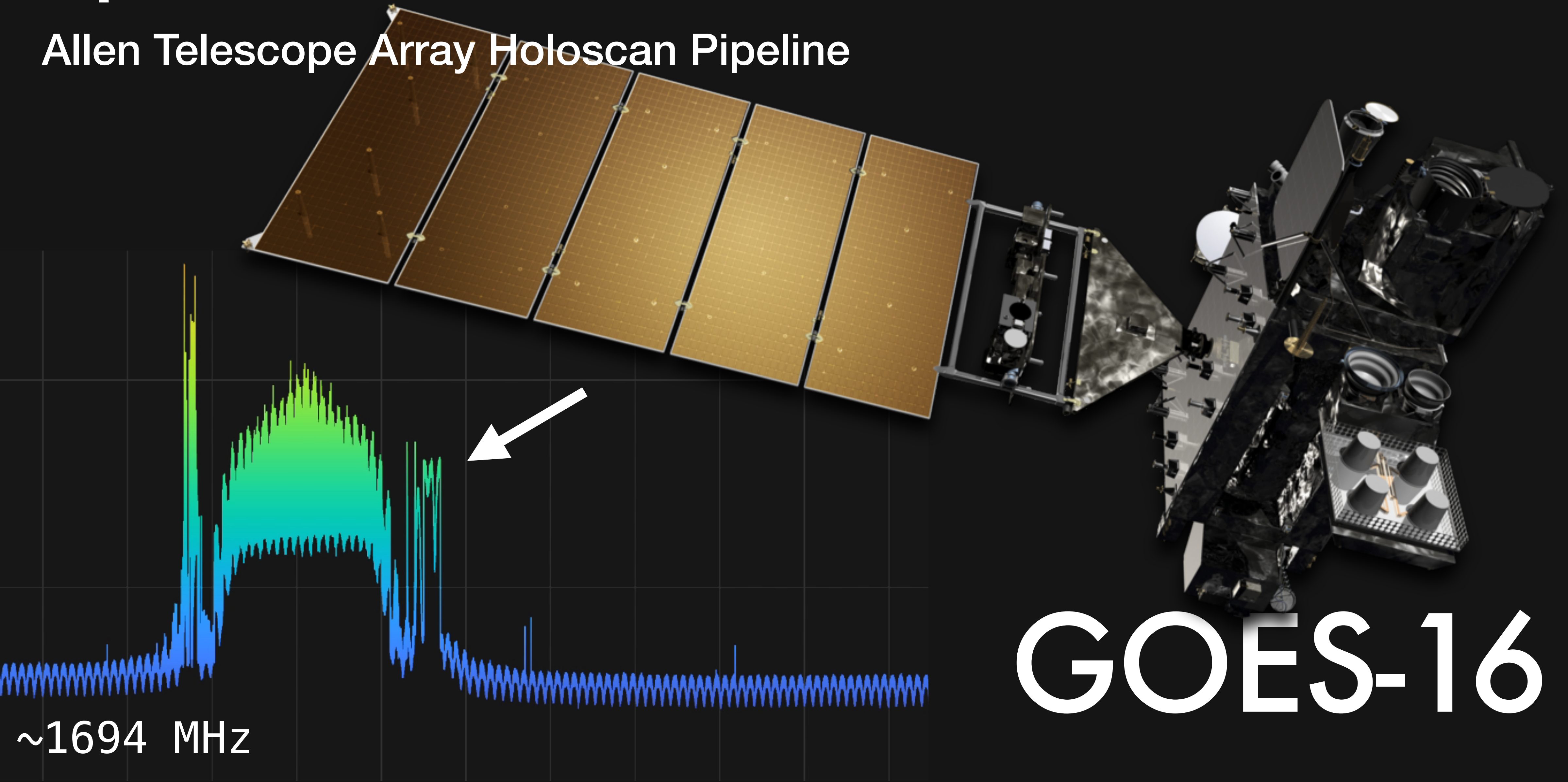
Deep Space Demonstration

Allen Telescope Array Holoscan Pipeline



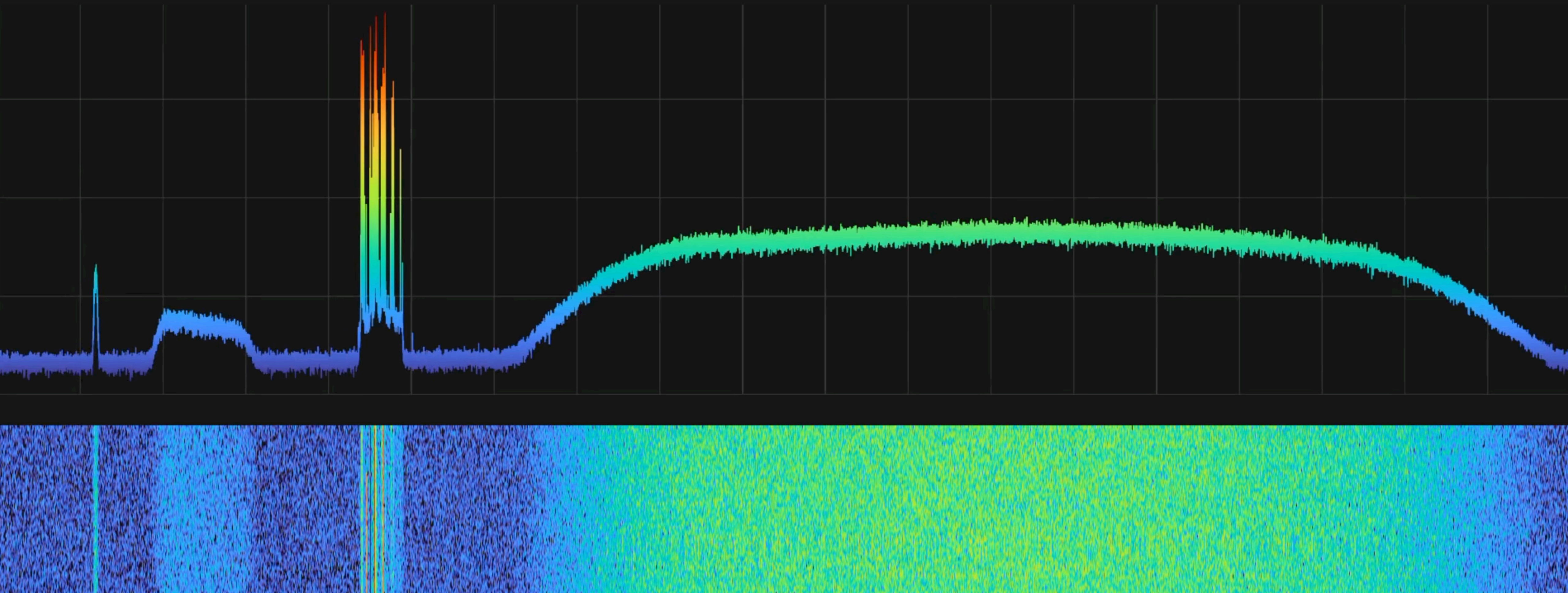
Space Demonstration

Allen Telescope Array Holoscan Pipeline



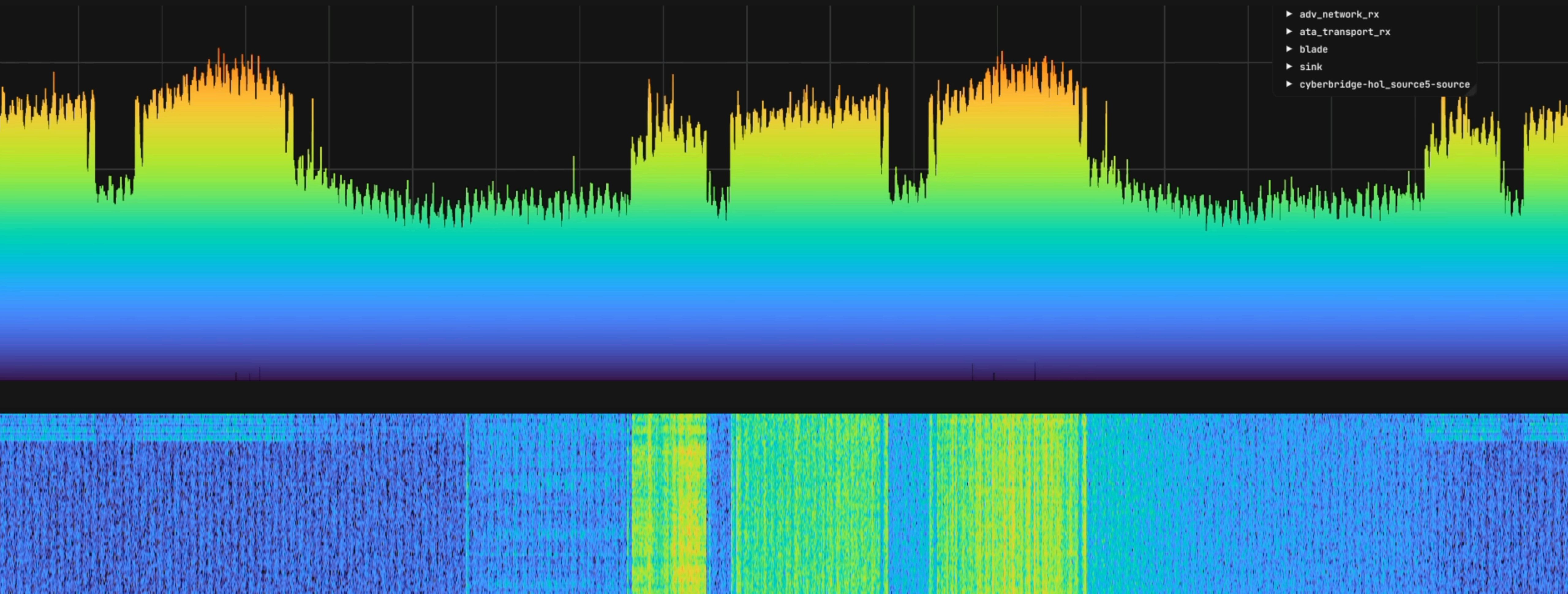
Space Demonstration

Allen Telescope Array Holoscan Pipeline



Space Demonstration

Allen Telescope Array Holoscan Pipeline



SUPERLUMINAL

Superluminal

High-level real-time visualization library for Python and C++.

- Simple high-level interface.
- Same high-performance visualization.
- Similar syntax for **C++** and **Python**.
- Implement in your project in minutes.

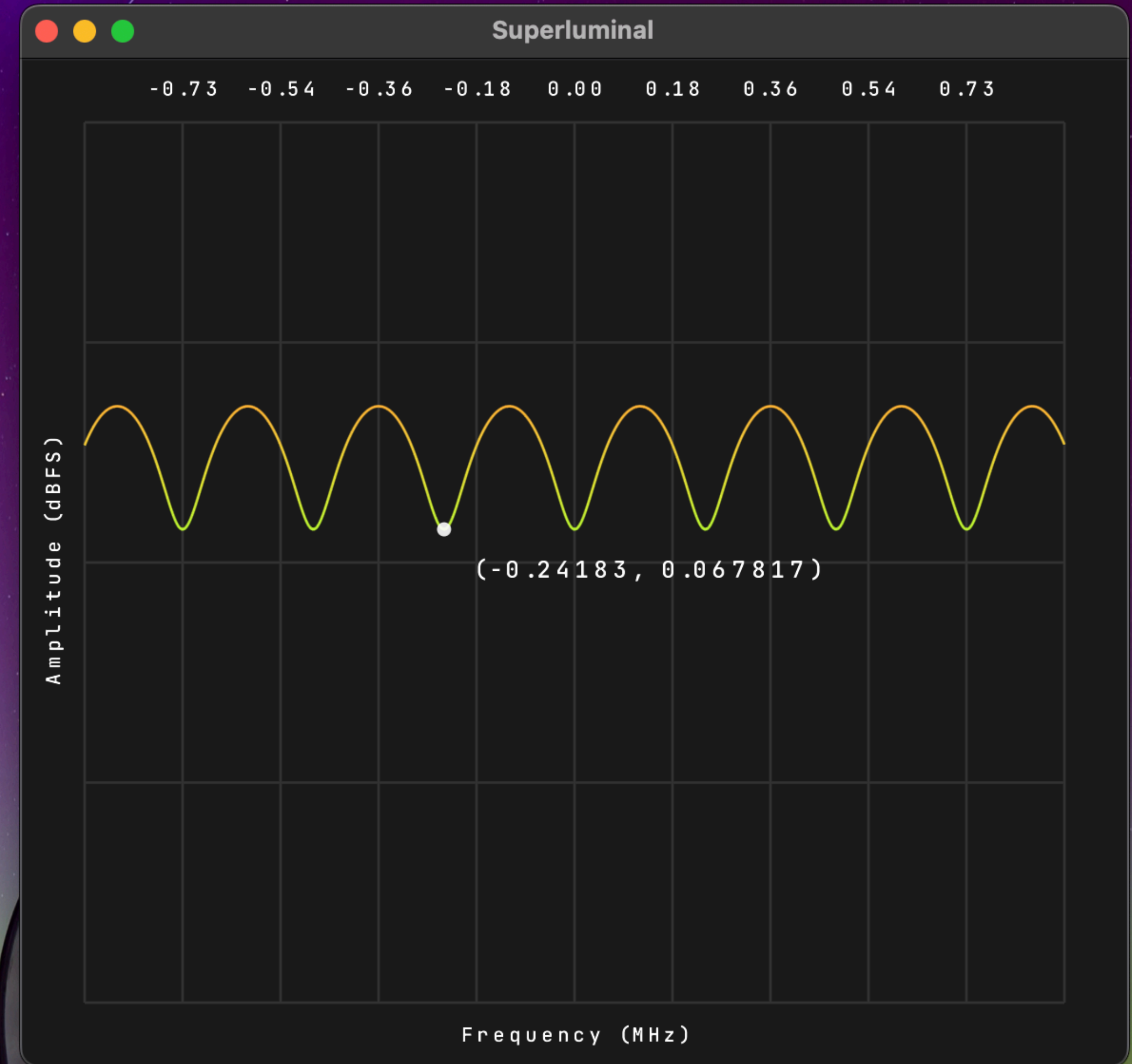
Superluminal

Static visualization

```
import numpy as np
import superluminal as lm

data = np.sin(np.linspace(0, 2 * np.pi, 8192))

lm.plot(data, lm.line, label="Sine")
lm.show()
```



Superluminal

Realtime visualization

```
import time
import numpy as np
import superluminal as lm

data = np.random.rand(1, 8192).astype(np.complex64)

lm.plot(data, lm.line, label="Sine")

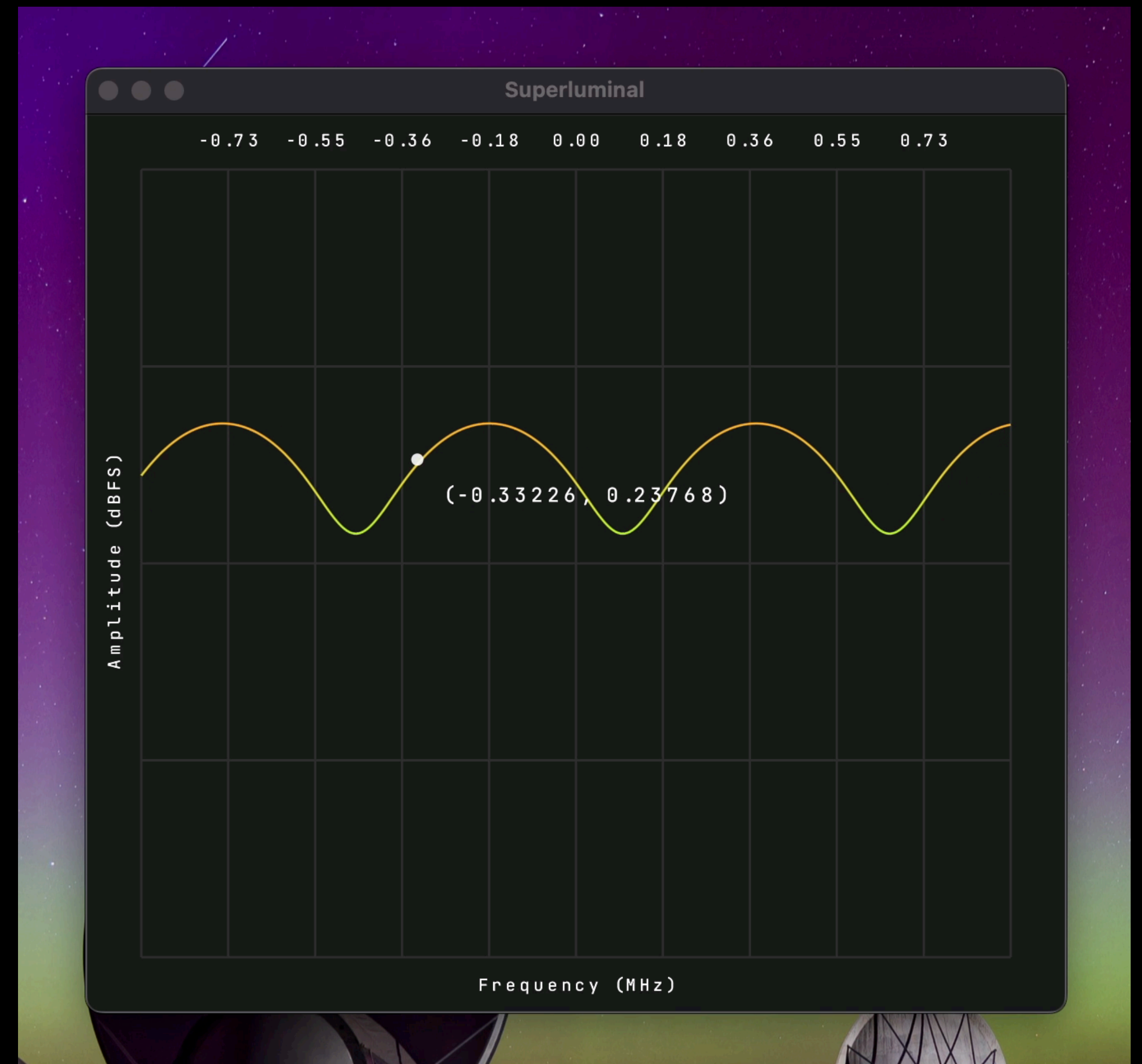
def callback():
    phase = 0

    while lm.running():
        # Generate sine wave
        for i in range(data.shape[1]):
            data[0, i] = np.sin(i * 0.01 + phase)
            phase += 0.1

        # Update the plot.
        lm.update()

        # Sleep for 10 ms.
        time.sleep(0.01)

lm.realtime(callback)
```



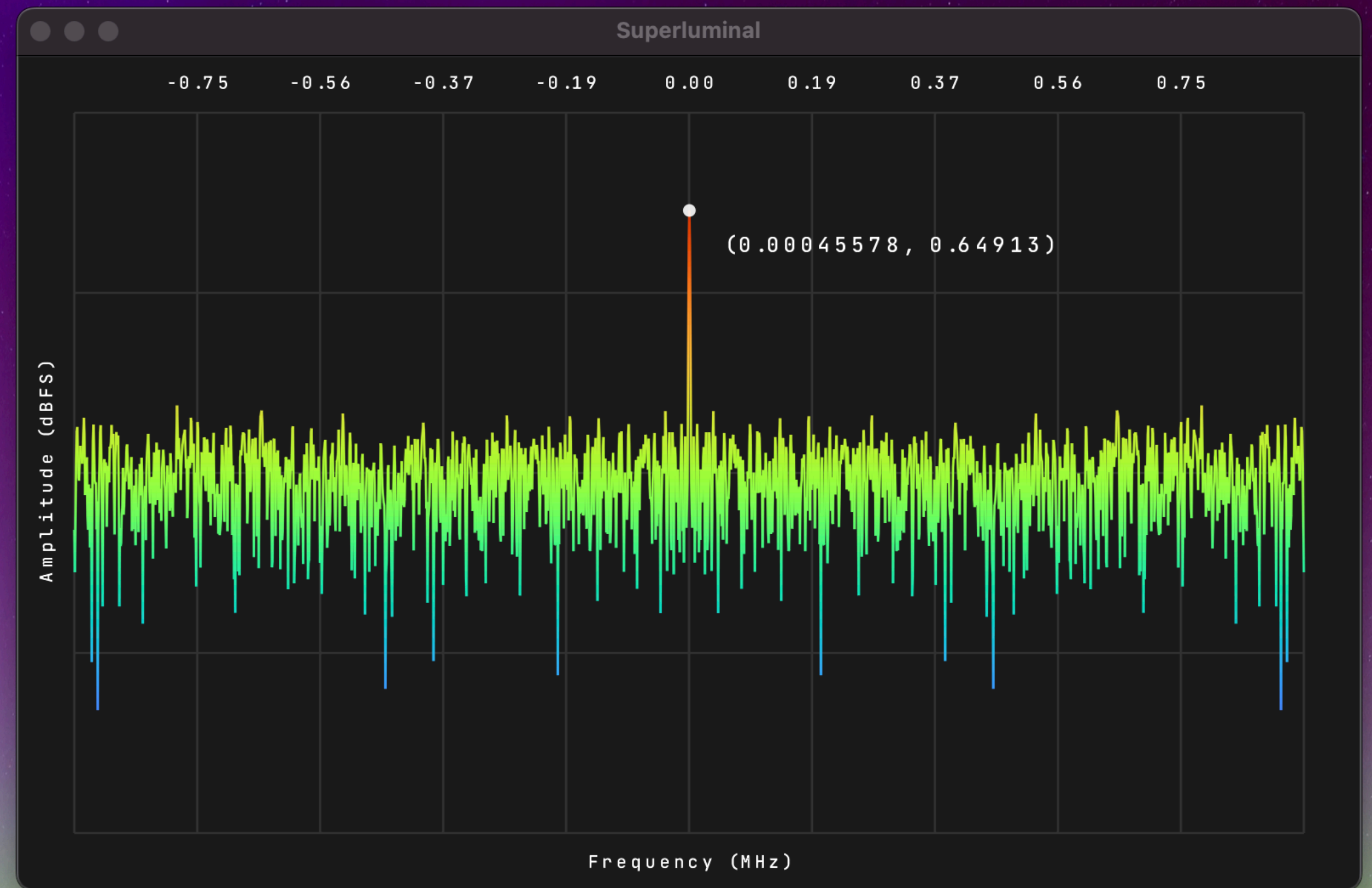
Superluminal

Time & Frequency Domain

```
import numpy as np
import superluminal as lm
```

```
data = np.random.rand(1, 8192).astype(np.complex64)
```

```
lm.plot(data, lm.line, label="Random", domain=(lm.time, lm.frequency))
lm.show()
```



Superluminal

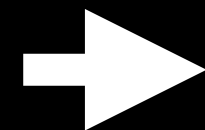
Syntax between languages

```
import numpy as np
import superluminal as lm

data = np.random.rand(1, 8192).astype(np.complex64)

lm.plot(data, lm.line, label="Random")
lm.show()
```

Python



```
#include <jetstream/superluminal.hh>

int main() {
    auto data = Tensor<Device::CPU, CF32>({1, 8192});

    for (U64 j = 0; j < data.shape(1); j++) {
        data[{0, j}] = std::rand() % 3;
    }

    Superluminal::Plot("Random", {{1}}, {
        .buffer = data,
        .type = Superluminal::Type::Line,
    });

    Superluminal::Show();

    return 0;
}
```

C++

Superluminal

Goals for Alpha 6

- Add more visualizations (Waterfall, Constellation, Spectrogram, etc).
- Make it installable with `pip install superluminal`.
- Add support for more data types.
- Add support for CUDA buffers.
- Add more examples.

CyberEther & Superluminal

MIT Licensed



Future Work

Call To Action

- I'm a single person working on this on my free time.
- Turns out it's very hard to do all of this.
- Community support is appreciated.

Help Wanted

- Stabilize Browser support.
- Create demodulation demos.
- Add third-party SDR support.
- Write more tests.
- Expand documentation.
- Stabilize Remote (transport and protocol).



Thanks for listening!

<https://github.com/luigifcruz/CyberEther>

Questions?

Contact me!

<https://luigi.ltd/contact/>

