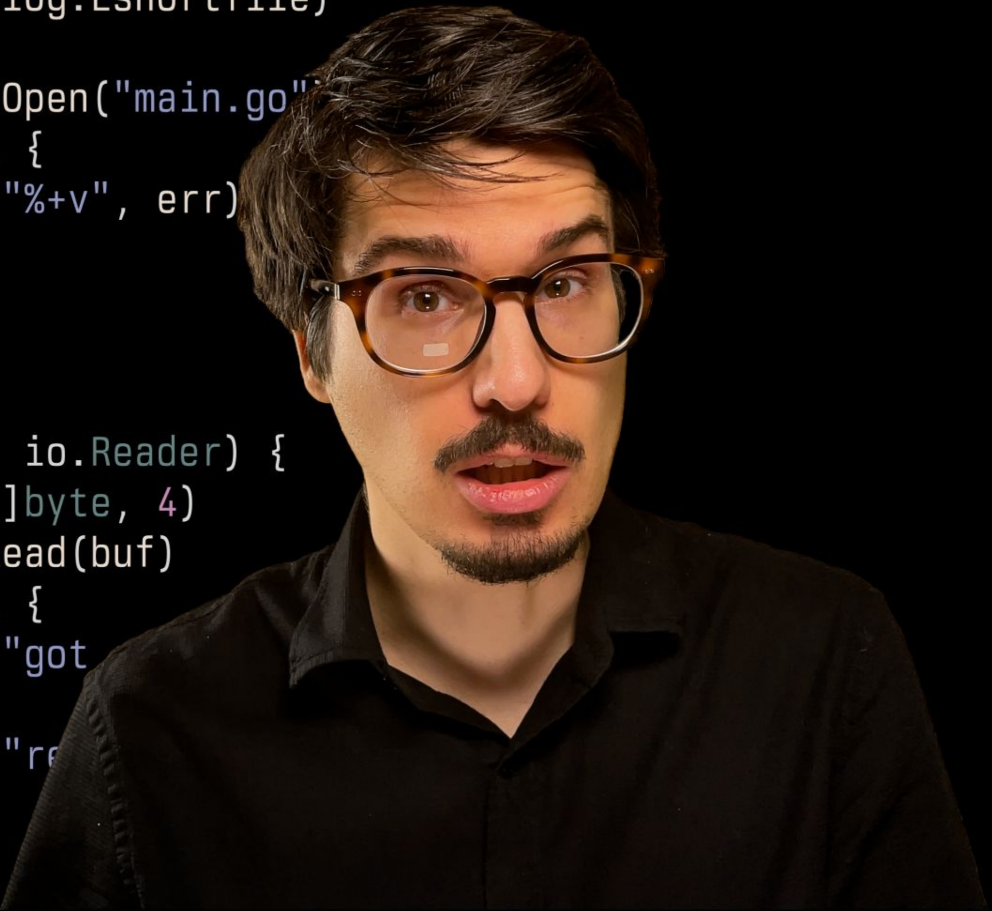
How far can you take OpenFX?

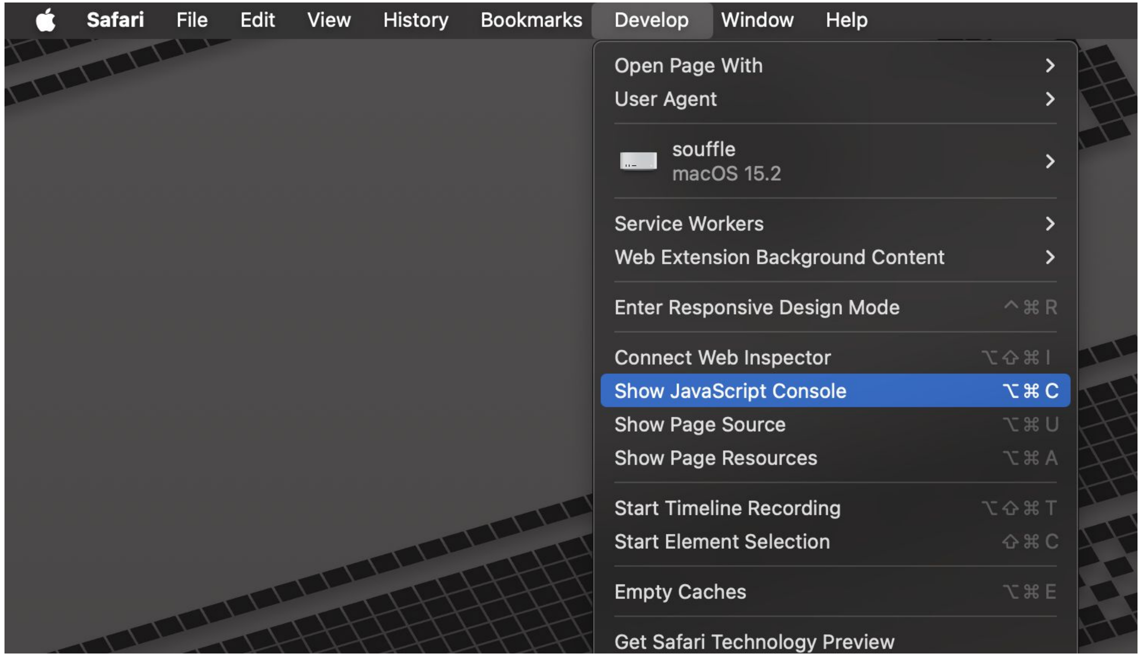
Can you embed a full motion design suite in it?

Previously on SDR

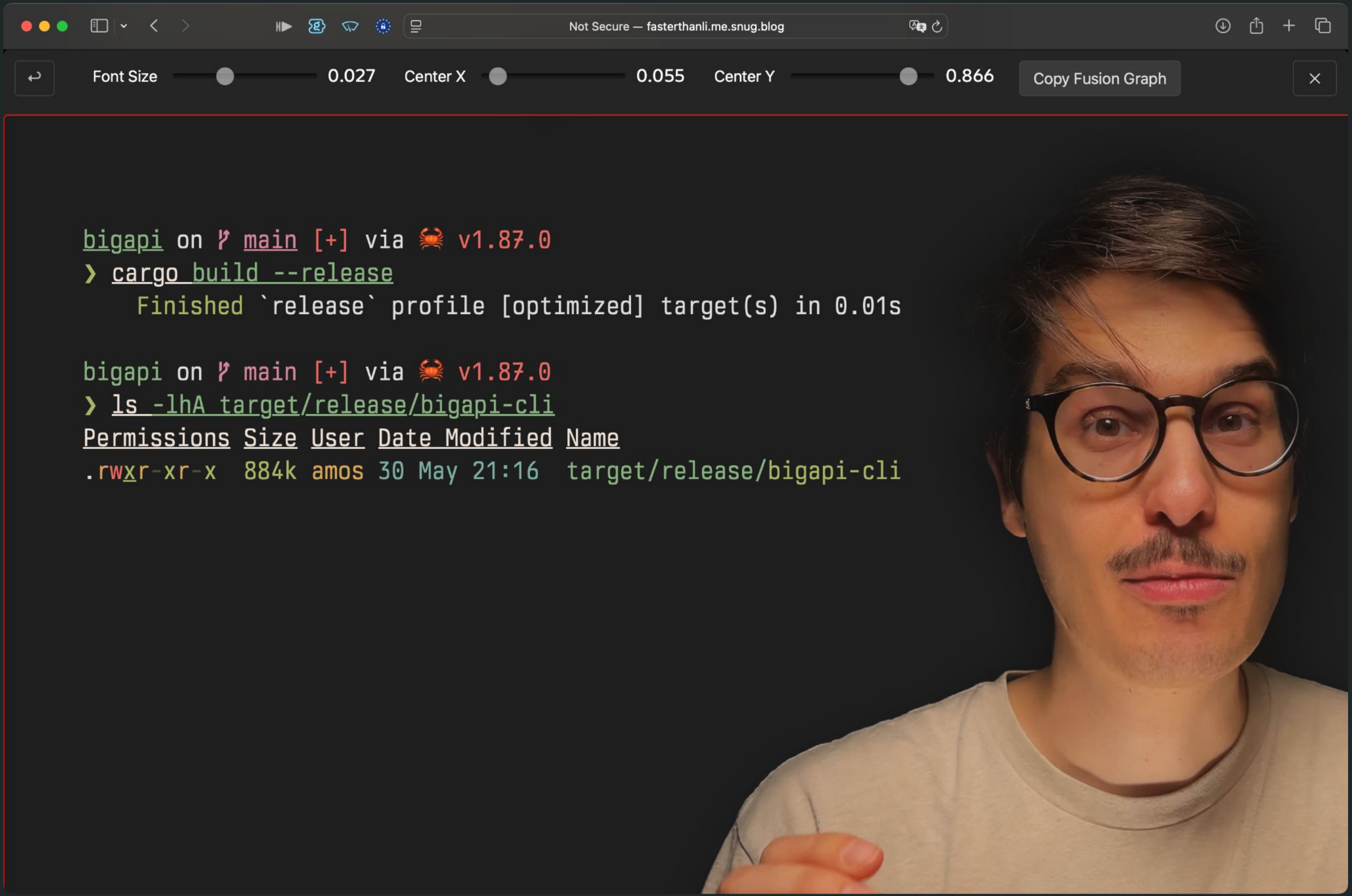
```go
func main() {
  log.SetFlags(log.Lshortfile)

  f, err := os.Open("main.go"
  if err != nil {
    log.Fatalf("%+v", err)
  }
  readSome(f)
}

func readSome(r io.Reader) {
  buf := make([]byte, 4)
  n, err := r.Read(buf)
  if err != nil {
    log.Printf("got
  } else {
    log.Printf("re
  }
}
```



```
tell application "System Events"
        tell application process "Safari"
                click menu item "Show JavaScript Console" of menu "Develop" of menu bar 1
                delay 1
        end tell
end tell
```

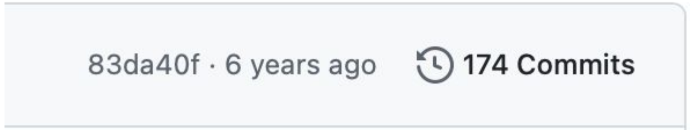

Safari
File
Edit
View
History
Bookmarks
Develop
Window
Help
Open Page With
User Agent
souffle
macOS 15.2
Service Workers
Web Extension Background Content
Enter Responsive Design Mode ^⌘R
Connect Web Inspector ⌥⇧⌘I
Show JavaScript Console ⌥⌘C
Show Page Source ⌥⌘U
Show Page Resources ⌥⌘A
Start Timeline Recording ⌥⇧⌘T
Start Element Selection ⇧⌘C
Empty Caches ⌥⌘E
Get Safari Technology Preview

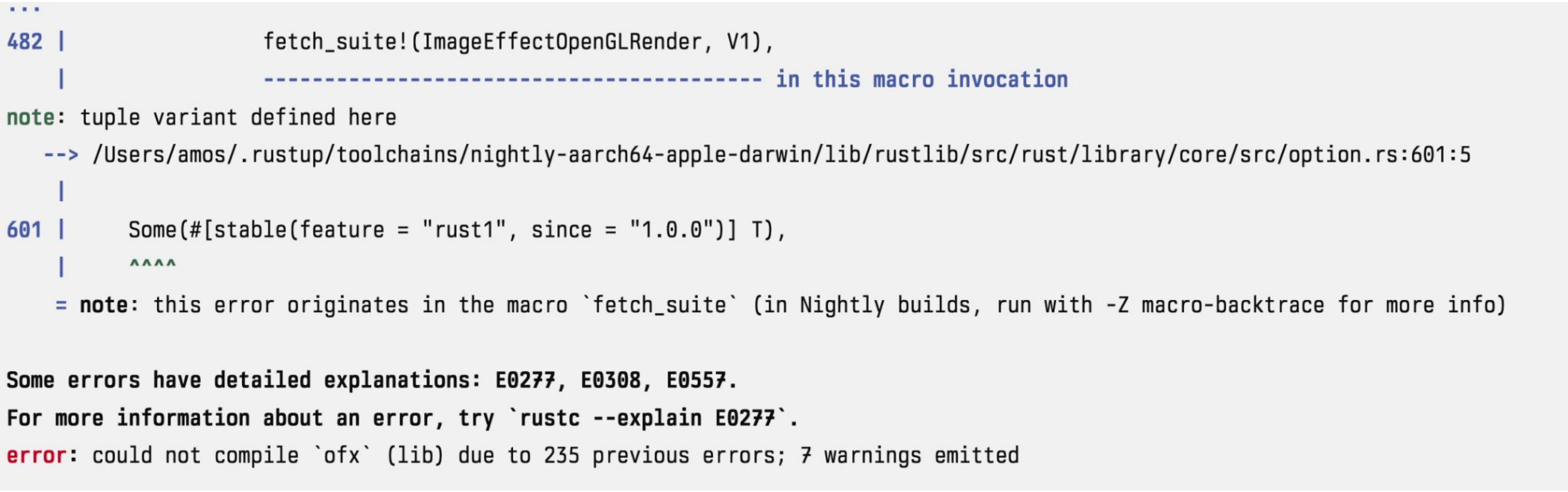
self-directed research



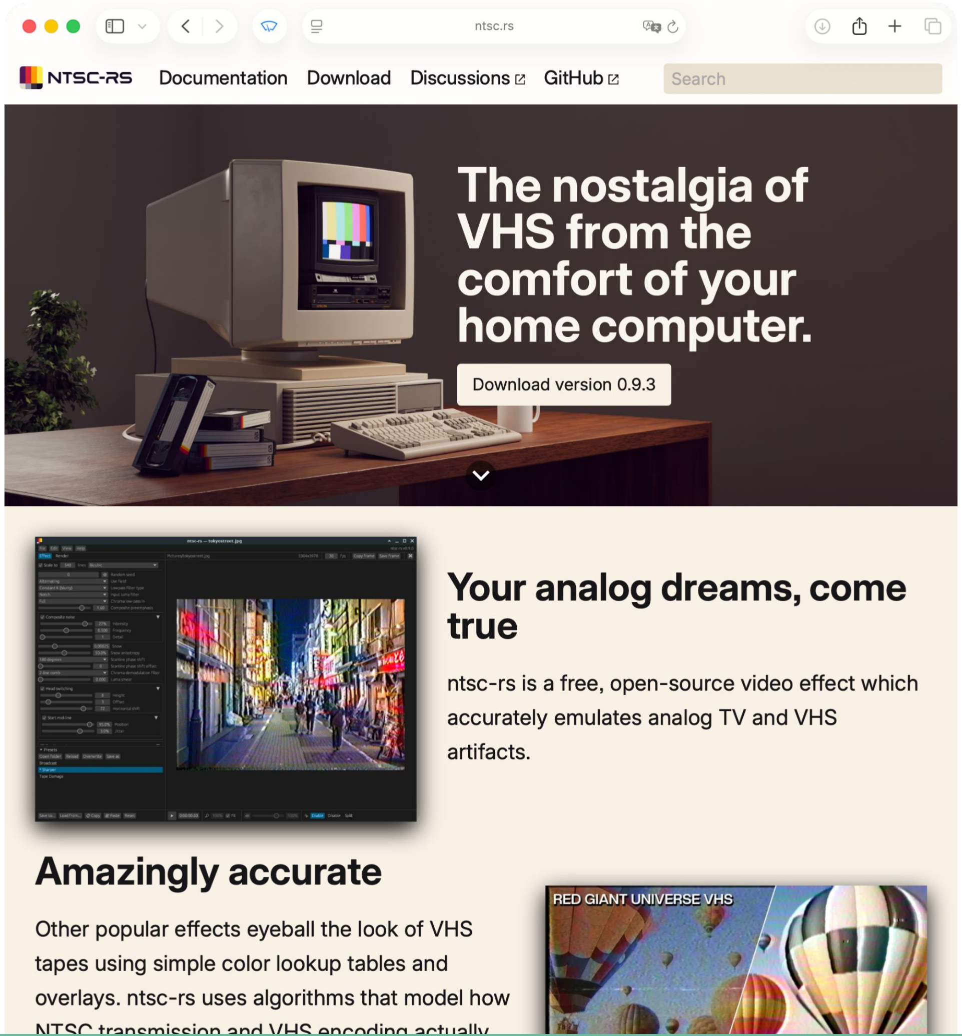
```
bigapi on  main [+] via 🦀 v1.87.0
❯ cargo build --release
    Finished `release` profile [optimized] target(s) in 0.01s

bigapi on  main [+] via 🦀 v1.87.0
❯ ls -lhA target/release/bigapi-cli
Permissions Size User Date Modified Name
.rwxr-xr-x  884k amos 30 May 21:16  target/release/bigapi-cli
```

It's time... for self-directed research


83da40f · 6 years ago 174 Commits


```
...
482 |                     fetch_suite!(ImageEffectOpenGLRender, V1),
    |                     ----------------------------------------- in this macro invocation
note: tuple variant defined here
   --> /Users/amos/.rustup/toolchains/nightly-aarch64-apple-darwin/lib/rustlib/src/rust/library/core/src/option.rs:601:5
    |
601 |     Some(#[stable(feature = "rust1", since = "1.0.0")] T),
    |     ^^^^
    = note: this error originates in the macro `fetch_suite` (in Nightly builds, run with -Z macro-backtrace for more info)

Some errors have detailed explanations: E0277, E0308, E0557.
For more information about an error, try `rustc --explain E0277`.
error: could not compile `ofx` (lib) due to 235 previous errors; 7 warnings emitted
```

self-directed research



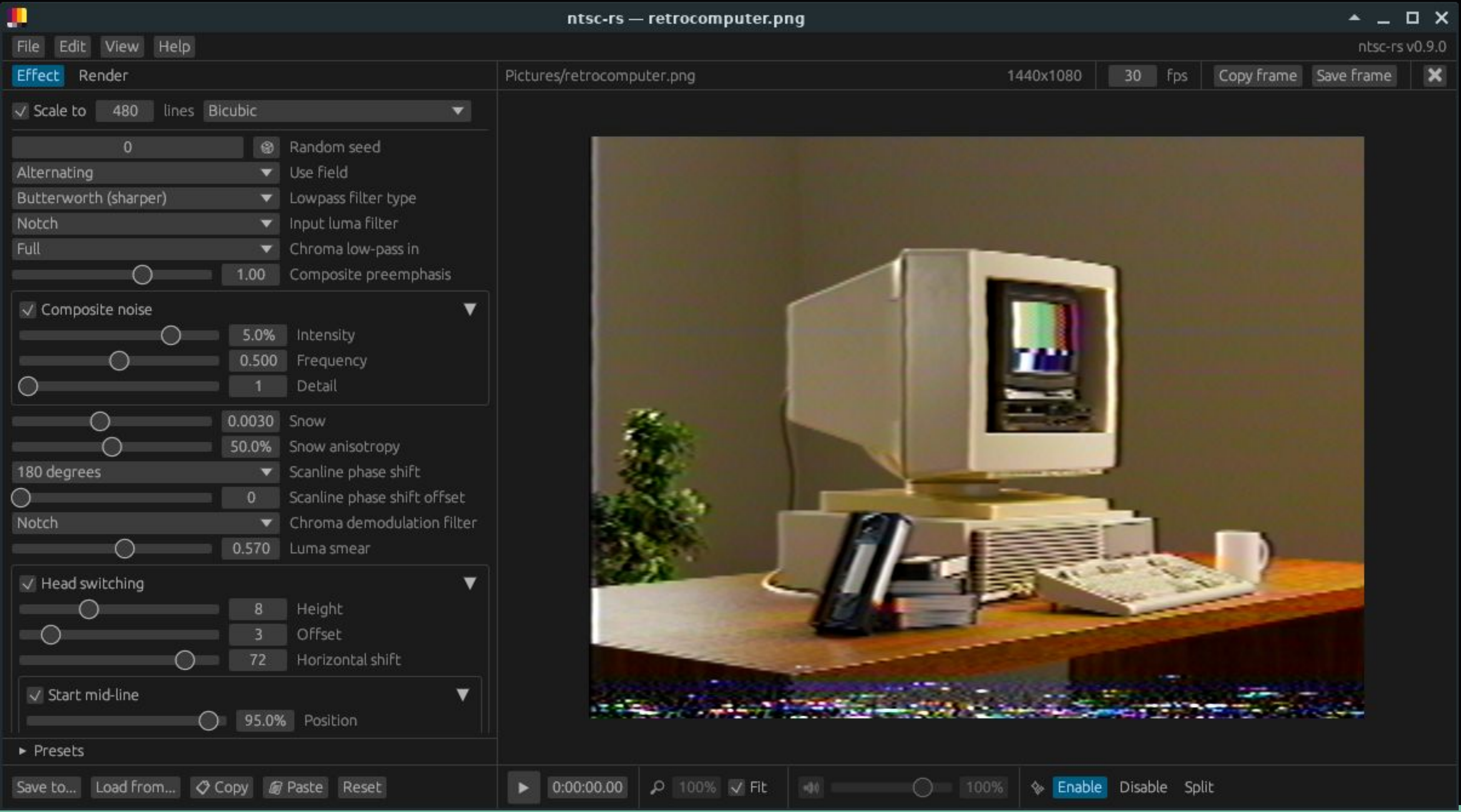
The nostalgia of VHS from the comfort of your home computer.

Your analog dreams, come true

ntsc-rs is a free, open-source video effect which accurately emulates analog TV and VHS artifacts.

Amazingly accurate

Other popular effects eyeball the look of VHS tapes using simple color lookup tables and overlays. ntsc-rs uses algorithms that model how NTSC transmission and VHS encoding actually


ntsc-rs — retrocomputer.png

File Edit View Help ntsc-rs v0.9.0

Effect Render

Pictures/retrocomputer.png 1440x1080 30 fps Copy frame Save frame

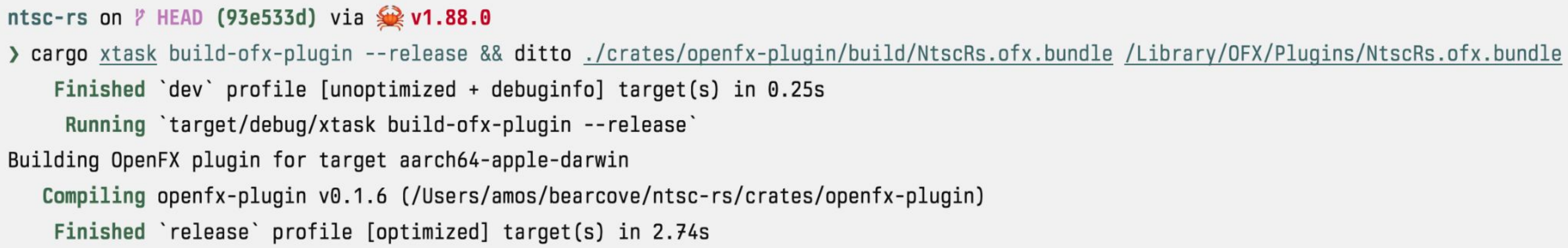
✓ Scale to 480 lines Bicubic

0	Random seed
Alternating	Use field
Butterworth (sharper)	Lowpass filter type
Notch	Input luma filter
Full	Chroma low-pass in
1.00	Composite preemphasis
✓ Composite noise	
5.0%	Intensity
0.500	Frequency
1	Detail
0.0030	Snow
50.0%	Snow anisotropy
180 degrees	Scanline phase shift
0	Scanline phase shift offset
Notch	Chroma demodulation filter
0.570	Luma smear
✓ Head switching	
8	Height
3	Offset
72	Horizontal shift
✓ Start mid-line	
95.0%	Position

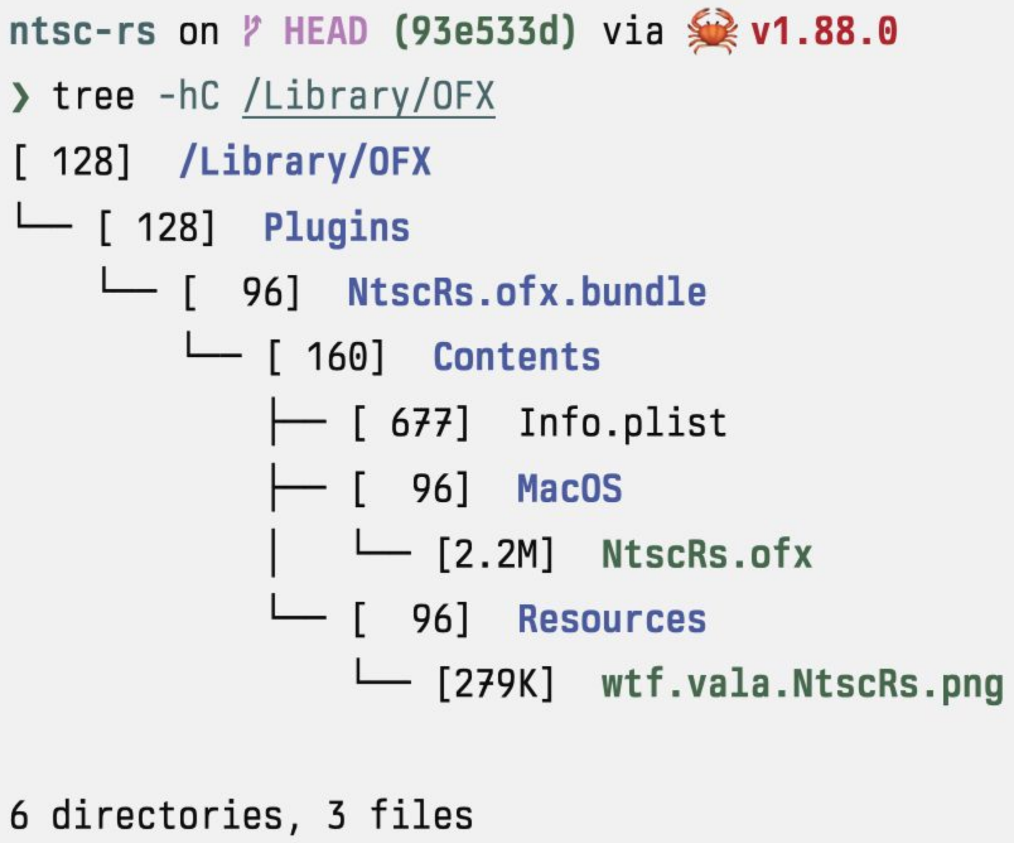
▸ Presets

Save to... Load from... Copy Paste Reset

0:00:00.00 100% ✓ Fit 100% Enable Disable Split



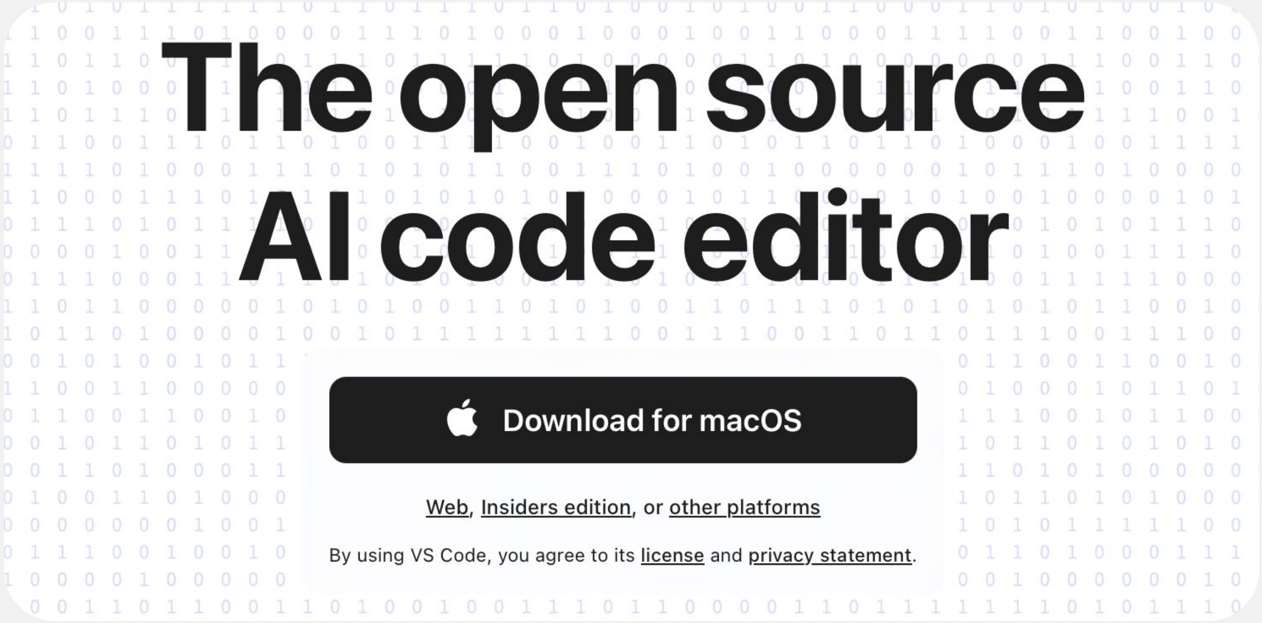
```
ntsc-rs on  HEAD (93e533d) via 🦀 v1.88.0
❯ cargo xtask build-ofx-plugin --release && ditto ./crates/openfx-plugin/build/NtscRs.ofx.bundle /Library/OFX/Plugins/NtscRs.ofx.bundle
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.25s
     Running `target/debug/xtask build-ofx-plugin --release`
Building OpenFX plugin for target aarch64-apple-darwin
   Compiling openfx-plugin v0.1.6 (/Users/amos/bearcove/ntsc-rs/crates/openfx-plugin)
    Finished `release` profile [optimized] target(s) in 2.74s
```



```
ntsc-rs on  HEAD (93e533d) via 🦀 v1.88.0
❯ tree -hC /Library/OFX
[ 128]  /Library/OFX
└── [ 128]  Plugins
    └── [  96]  NtscRs.ofx.bundle
        └── [ 160]  Contents
            ├── [ 677]  Info.plist
            ├── [  96]  MacOS
            │   └── [2.2M]  NtscRs.ofx
            └── [  96]  Resources
                └── [279K]  wtf.vala.NtscRs.png

6 directories, 3 files
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleDevelopmentRegion</key>
    <string>en</string>
    <key>CFBundlePackageType</key>
    <string>BNDL</string>
    <key>CFBundleIdentifier</key>
    <string>rs.ntsc.openfx</string>
    <key>CFBundleVersion</key>
    <string>0.1.6</string>
    <key>CFBundleShortVersionString</key>
    <string>0.1.6</string>
    <key>NSHumanReadableCopyright</key>
    <string>© 2023-2025 valadaptive</string>
```

```
// Combine the x86_64 and aarch64 builds into one using `lipo`, and output to the temp file we created
// above.
// TODO: Create the directories beforehand, output into that with lipo, and just rename it afterwards?
Command::new(program: "lipo") Command
    .args(&[
        OsString::from("-create"),
        OsString::from("-output"),
        dst_path.clone().into(),
        x86_64_path.into(),
        aarch64_path.into(),
    ]) &mut Command
    .status() Result<ExitStatus, Error>
    .expect_success()?;
```



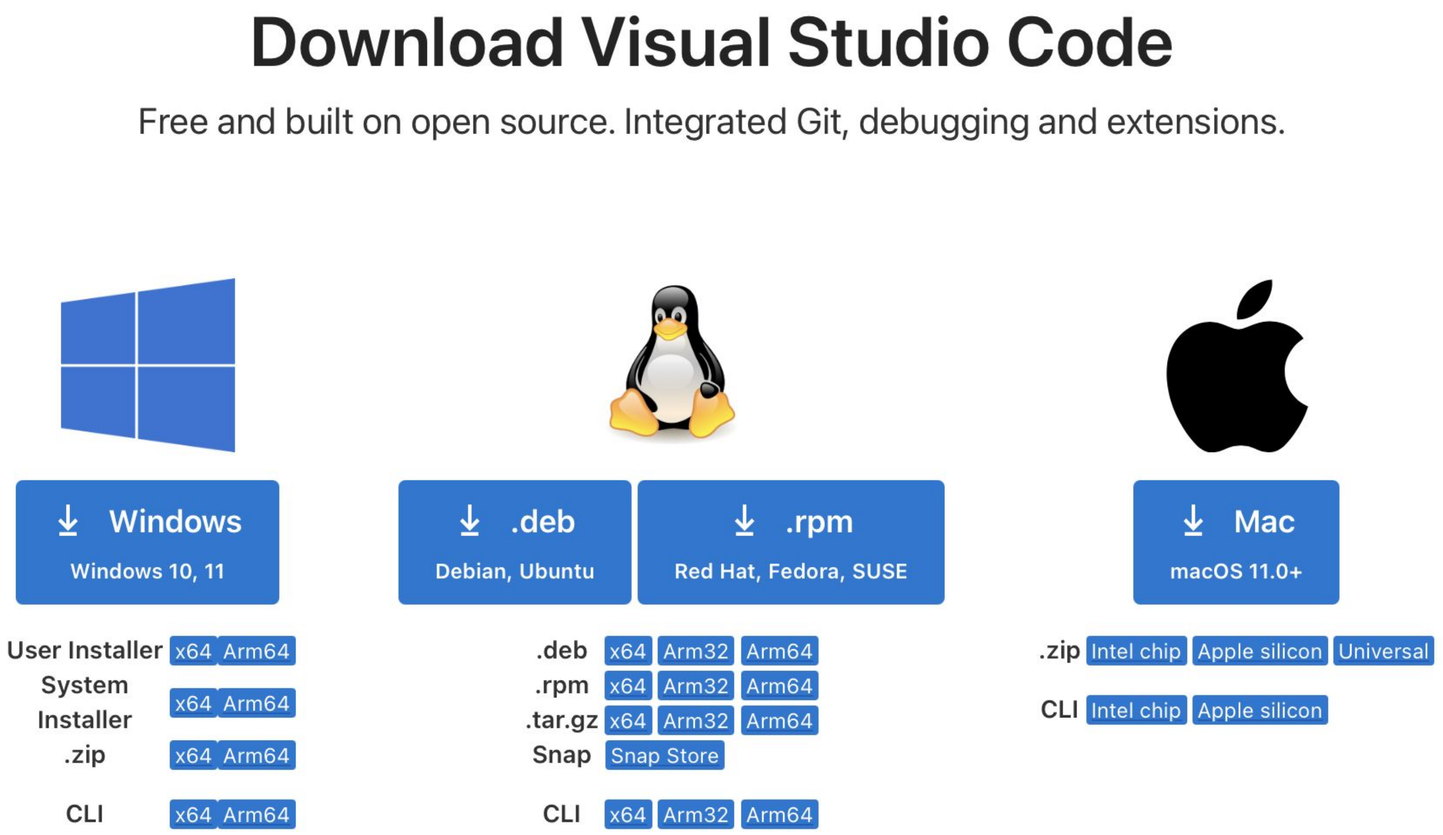
The open source
AI code editor

Download for macOS

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.

```
~/Downloads
❯ lipo -info 'Visual Studio Code.app/Contents/MacOS/Electron'
Architectures in the fat file: Visual Studio Code.app/Contents/MacOS/Electron are: x86_64 arm64
```



Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

Windows
Windows 10, 11

User Installer	x64	Arm64
System Installer	x64	Arm64
.zip	x64	Arm64
CLI	x64	Arm64

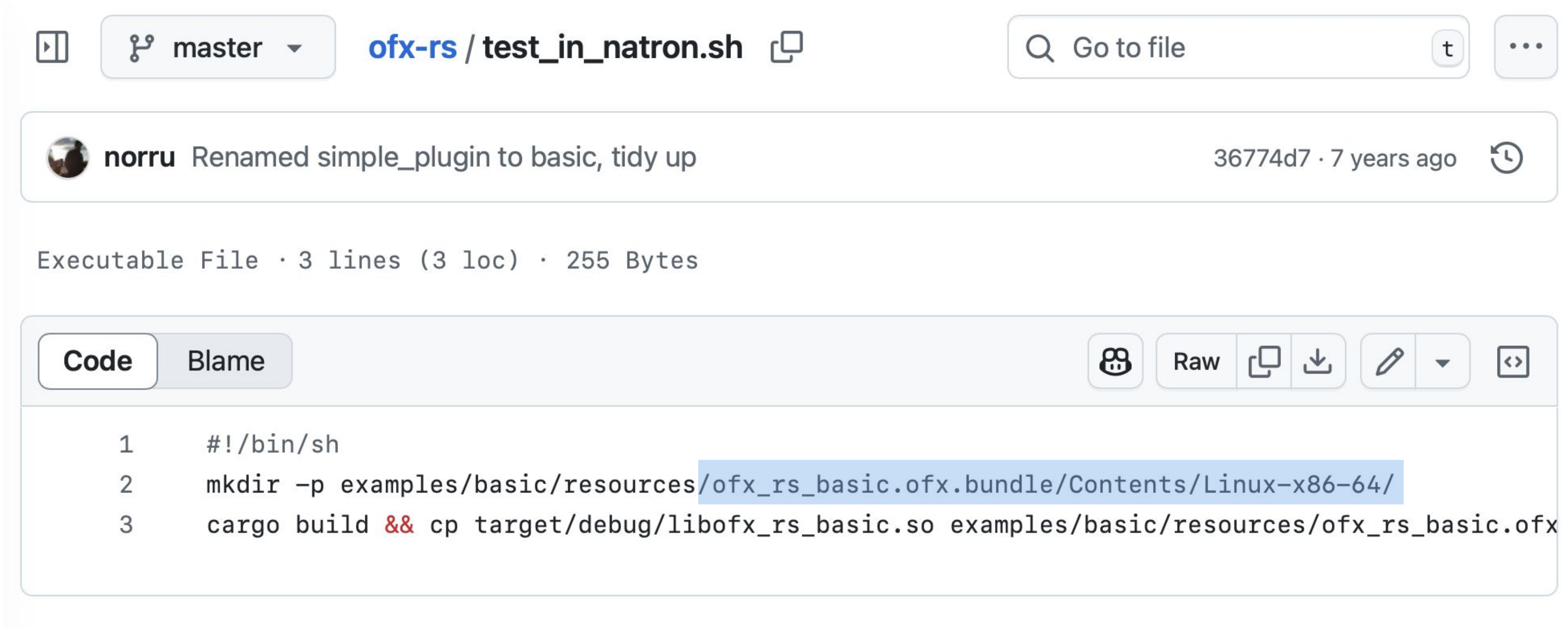
.deb
Debian, Ubuntu

.rpm
Red Hat, Fedora, SUSE

.deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64	Arm32	Arm64

Mac
macOS 11.0+

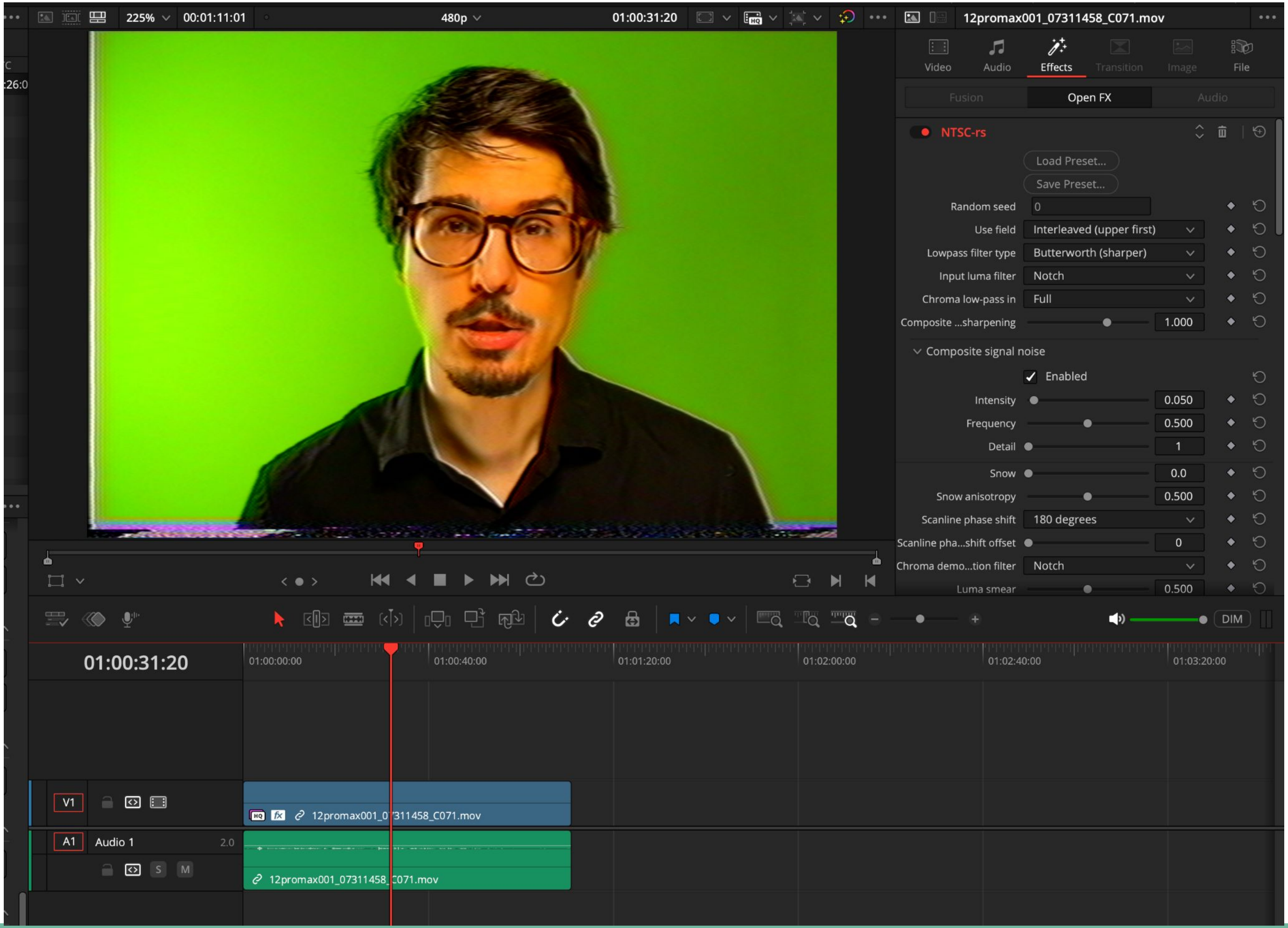
.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip	Apple silicon	



norru Renamed simple_plugin to basic, tidy up 36774d7 · 7 years ago

Executable File · 3 lines (3 loc) · 255 Bytes

```
#!/bin/sh
mkdir -p examples/basic/resources/ofx_rs_basic.ofx.bundle/Contents/Linux-x86-64/
cargo build && cp target/debug/libofx_rs_basic.so examples/basic/resources/ofx_rs_basic.ofx
```

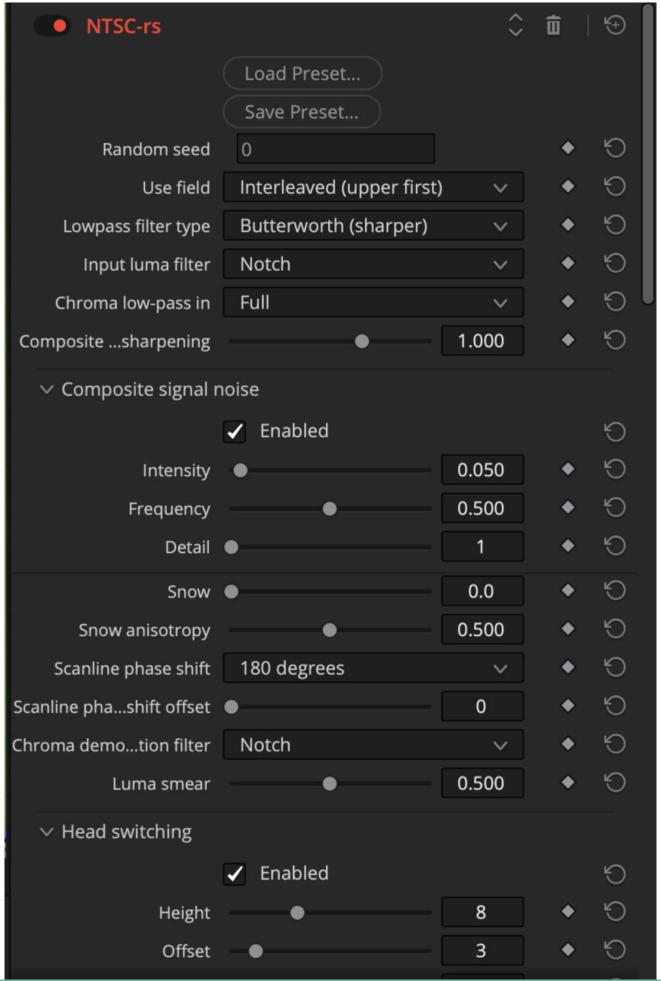


NTSC-rs

Random seed	0
Use field	Interleaved (upper first)
Lowpass filter type	Butterworth (sharper)
Input luma filter	Notch
Chroma low-pass in	Full
Composite ...sharpening	1.000

Composite signal noise

Enabled	✓
Intensity	0.050
Frequency	0.500
Detail	1
Snow	0.0
Snow anisotropy	0.500
Scanline phase shift	180 degrees
Scanline pha...shift offset	0
Chroma demo...tion filter	Notch
Luma smear	0.500


NTSC-rs

Load Preset...

Save Preset...

Random seed	0
Use field	Interleaved (upper first)
Lowpass filter type	Butterworth (sharper)
Input luma filter	Notch
Chroma low-pass in	Full
Composite ...sharpening	1.000

Composite signal noise

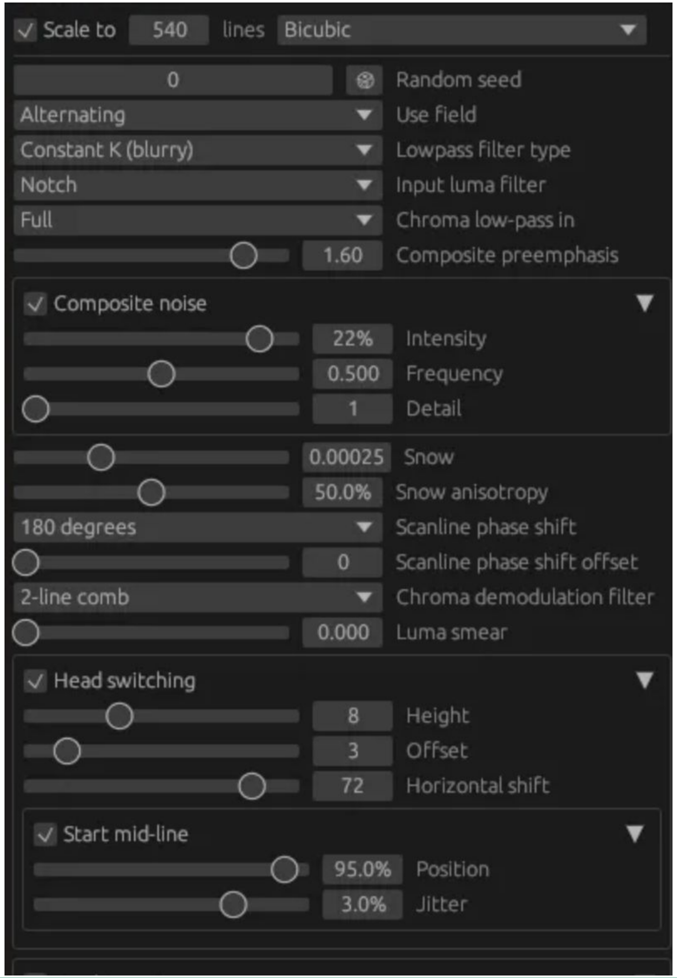
✓ Enabled

Intensity	0.050
Frequency	0.500
Detail	1
Snow	0.0
Snow anisotropy	0.500
Scanline phase shift	180 degrees
Scanline pha...shift offset	0
Chroma demo...tion filter	Notch
Luma smear	0.500

Head switching

✓ Enabled

Height	8
Offset	3


✓ Scale to 540 lines Bicubic

0	Random seed
Alternating	Use field
Constant K (blurry)	Lowpass filter type
Notch	Input luma filter
Full	Chroma low-pass in
1.60	Composite preemphasis

✓ Composite noise

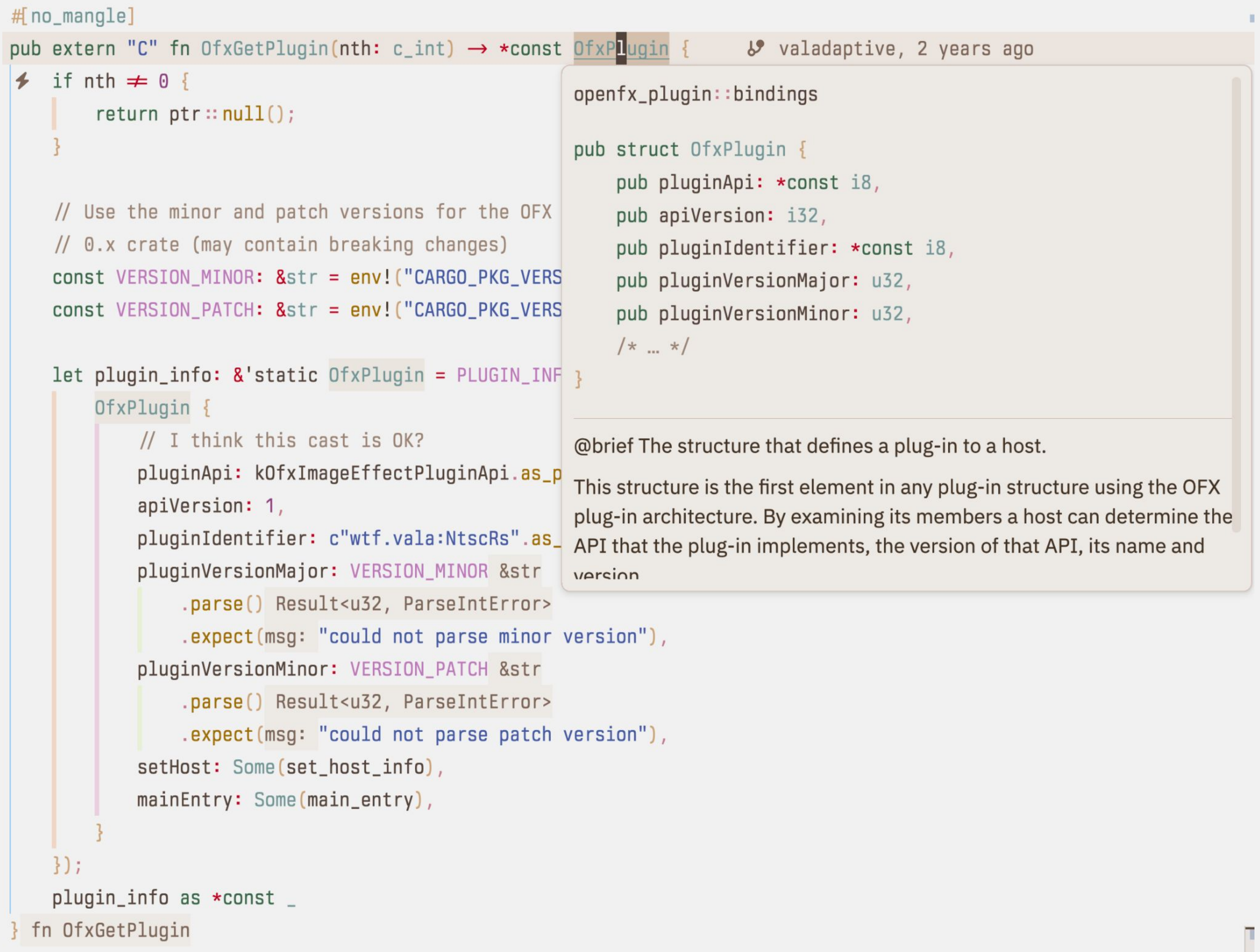
22%	Intensity
0.500	Frequency
1	Detail
0.00025	Snow
50.0%	Snow anisotropy
180 degrees	Scanline phase shift
0	Scanline phase shift offset
2-line comb	Chroma demodulation filter
0.000	Luma smear

✓ Head switching

8	Height
3	Offset
72	Horizontal shift

✓ Start mid-line

95.0%	Position
3.0%	Jitter



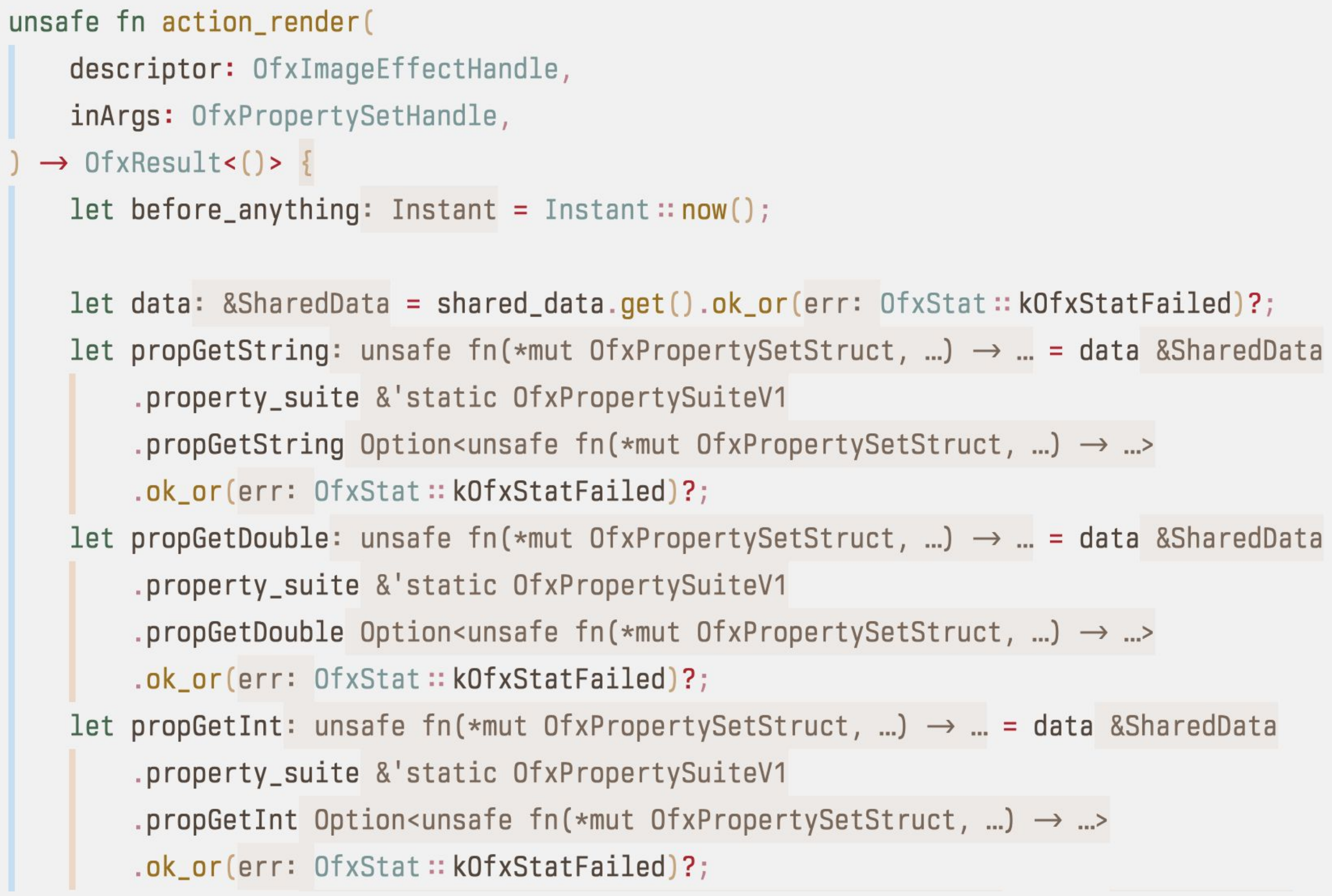
```
#[no_mangle]
pub extern "C" fn OfxGetPlugin(nth: c_int) → *const OfxPlugin {      valadaptive, 2 years ago
    if nth ≠ 0 {
        return ptr::null();
    }

    // Use the minor and patch versions for the OFX
    // 0.x crate (may contain breaking changes)
    const VERSION_MINOR: &str = env!("CARGO_PKG_VERS
    const VERSION_PATCH: &str = env!("CARGO_PKG_VERS

    let plugin_info: &'static OfxPlugin = PLUGIN_INF
        OfxPlugin {
            // I think this cast is OK?
            pluginApi: kOfxImageEffectPluginApi.as_p
            apiVersion: 1,
            pluginIdentifier: c"wtf.vala:NtscRs".as_
            pluginVersionMajor: VERSION_MINOR &str
                .parse() Result<u32, ParseIntError>
                .expect(msg: "could not parse minor version"),
            pluginVersionMinor: VERSION_PATCH &str
                .parse() Result<u32, ParseIntError>
                .expect(msg: "could not parse patch version"),
            setHost: Some(set_host_info),
            mainEntry: Some(main_entry),
        }
    });
    plugin_info as *const _
} fn OfxGetPlugin
```

```
openfx_plugin::bindings

pub struct OfxPlugin {
    pub pluginApi: *const i8,
    pub apiVersion: i32,
    pub pluginIdentifier: *const i8,
    pub pluginVersionMajor: u32,
    pub pluginVersionMinor: u32,
    /* … */
}
```

@brief The structure that defines a plug-in to a host.

This structure is the first element in any plug-in structure using the OFX plug-in architecture. By examining its members a host can determine the API that the plug-in implements, the version of that API, its name and version

```rust
unsafe extern "C" fn main_entry(
    action: *const c_char,
    handle: *const c_void,
    inArgs: OfxPropertySetHandle,
    outArgs: OfxPropertySetHandle,
) -> OfxStatus {
    let effect: *mut OfxImageEffectStruct = handle as OfxImageEffectHandle;
    let action: &CStr = CStr::from_ptr(action);

    // Needed so Resolve doesn't swallow the panic info
    std::panic::set_hook(Box::new(|info: &PanicHookInfo<'_>| {
        println!("{:?}", info);
    }));

    let return_status: Result<(), OfxStatus> = if action == kOfxActionLoad {
        action_load()
    } else if action == kOfxActionDescribe {
        action_describe(descriptor: effect)
    } else if action == kOfxImageEffectActionDescribeInContext {
        action_describe_in_context(descriptor: effect)
    } else if action == kOfxImageEffectActionGetRegionsOfInterest {
        action_get_regions_of_interest(descriptor: effect, inArgs, outArgs)
    } else if action == kOfxImageEffectActionGetClipPreferences {
        action_get_clip_preferences(outArgs)
    } else if action == kOfxActionInstanceChanged {
        action_instance_changed(descriptor: effect, inArgs)
    } else if action == kOfxImageEffectActionRender {
        action_render(descriptor: effect, inArgs)
```

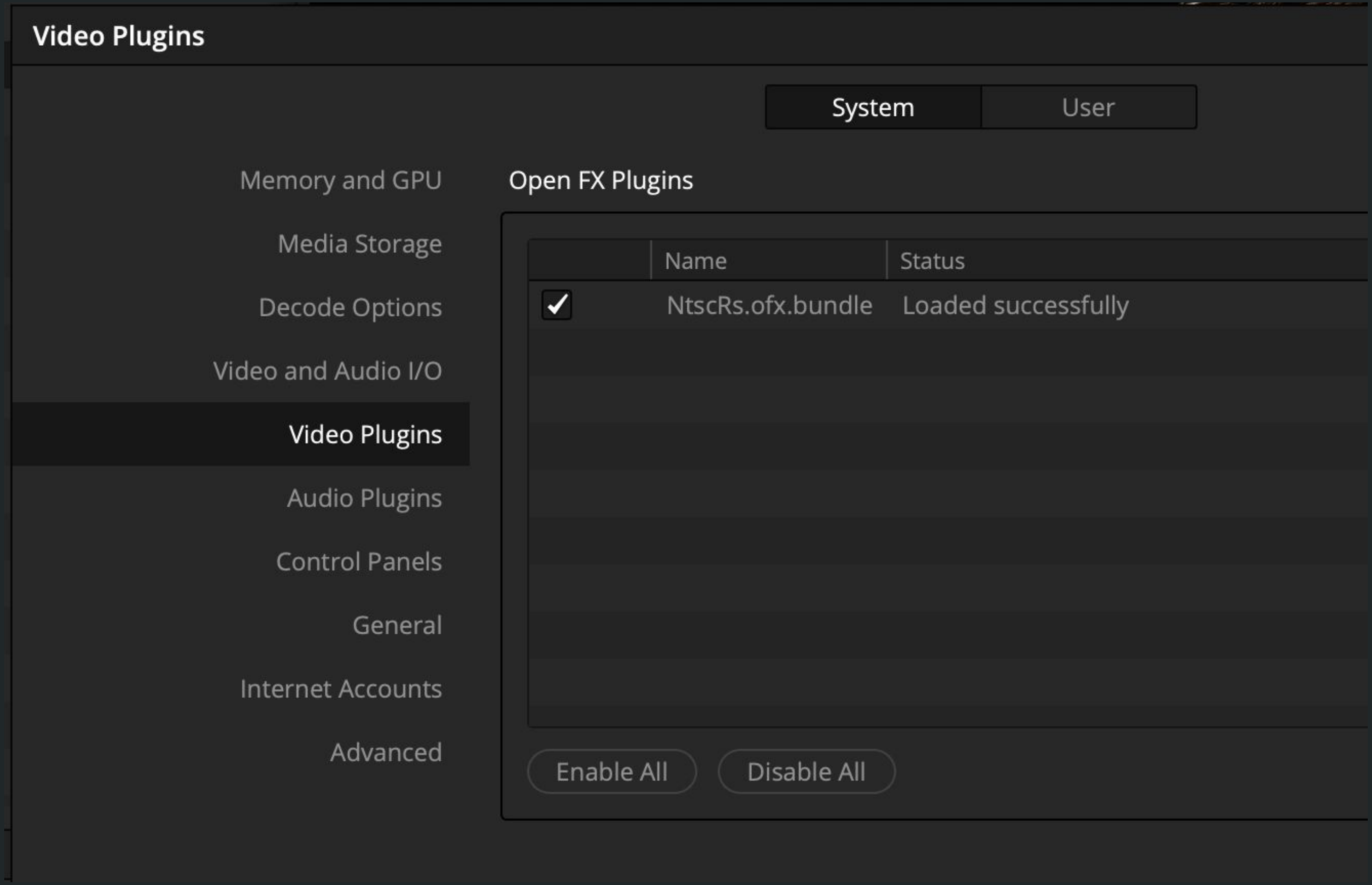


```
unsafe fn action_render(
    descriptor: OfxImageEffectHandle,
    inArgs: OfxPropertySetHandle,
) → OfxResult<()> {
    let before_anything: Instant = Instant::now();

    let data: &SharedData = shared_data.get().ok_or(err: OfxStat::kOfxStatFailed)?;
    let propGetString: unsafe fn(*mut OfxPropertySetStruct, …) → … = data &SharedData
        .property_suite &'static OfxPropertySuiteV1
        .propGetString Option<unsafe fn(*mut OfxPropertySetStruct, …) → …>
        .ok_or(err: OfxStat::kOfxStatFailed)?;
    let propGetDouble: unsafe fn(*mut OfxPropertySetStruct, …) → … = data &SharedData
        .property_suite &'static OfxPropertySuiteV1
        .propGetDouble Option<unsafe fn(*mut OfxPropertySetStruct, …) → …>
        .ok_or(err: OfxStat::kOfxStatFailed)?;
    let propGetInt: unsafe fn(*mut OfxPropertySetStruct, …) → … = data &SharedData
        .property_suite &'static OfxPropertySuiteV1
        .propGetInt Option<unsafe fn(*mut OfxPropertySetStruct, …) → …>
        .ok_or(err: OfxStat::kOfxStatFailed)?;
```

```
propGetDouble(inArgs, kOfxPropTime.as_ptr(), 0, &mut time).ofx_ok()?;
// I'm sure nothing bad will happen here as a result of propGetIntN writing past the pointer it was given
propGetIntN(
    inArgs,
    kOfxImageEffectPropRenderWindow.as_ptr(),
    4,
    ptr::addr_of_mut!(renderWindow) as *mut _,
) OfxStatus
.ofx_ok()?;

let mut outputClip: OfxImageClipHandle = ptr::null_mut();
clipGetHandle(
    descriptor,
    c"Output".as_ptr(),
    &mut outputClip,
    ptr::null_mut(),
)
.ofx_ok()?;
```

How is developing an OpenFX plug-in?

```rust
const DEBUG_LOG_PATH: &str = "/tmp/ofx-debug-log.txt";

pub(crate) fn append_debug_log(line: &str) -> std::io::Result<()> {
    use std::fs::OpenOptions;
    use std::io::Write;
    use std::sync::{Mutex, OnceLock};

    static FILE: OnceLock<Mutex<std::fs::File>> = OnceLock::new();

    let file_mutex: &Mutex<File> = FILE.get_or_init(|| {
        let file: File = OpenOptions::new() OpenOptions
            .create(true) &mut OpenOptions
            .append(true) &mut OpenOptions
            .open(DEBUG_LOG_PATH) Result<File, Error>
            .unwrap();
        Mutex::new(file)
    });

    let mut file: MutexGuard<'_, File> = file_mutex.lock().unwrap();
    writeln!(file, "{line}")?;
    Ok(())
}
```



Video Plugins

System | User

Memory and GPU
Media Storage
Decode Options
Video and Audio I/O
Video Plugins
Audio Plugins
Control Panels
General
Internet Accounts
Advanced

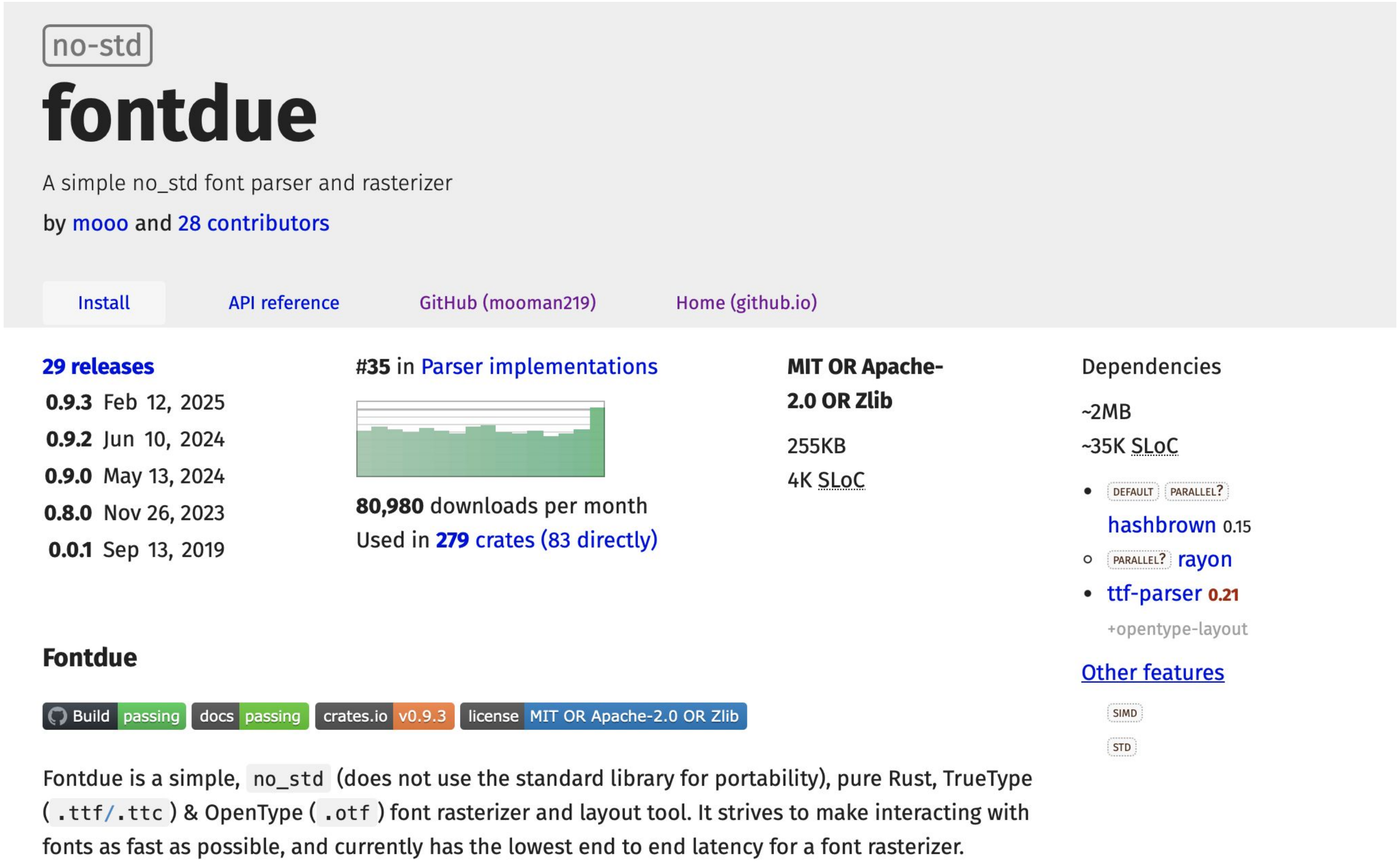
Open FX Plugins

	Name	Status
✓	NtscRs.ofx.bundle	Loaded successfully

Enable All | Disable All

```
print("Rendering test image...")
write = app1.createWriter("target/filtered_test_####.png")

source = app1.createNode("net.sf.openfx.CheckerBoardPlugin")
mask = app1.createNode("net.sf.openfx.Radial")

under_test = app1.createNode("net.itadinanta.ofx-rs.basic")
under_test.connectInput(0, source)
under_test.connectInput(1, mask)

under_test.getParam("scaleComponents").setValue(True)
under_test.getParam("scale").setValue(1.0)
under_test.getParam("scaleR").setValue(1.5)

write.connectInput(0, under_test)

app1.render(write, 1, 1)

quit()
```

What about the render?



no-std

fontdue

A simple no_std font parser and rasterizer

by mooo and 28 contributors

Install   API reference   GitHub (mooman219)   Home (github.io)

29 releases

0.9.3	Feb 12, 2025
0.9.2	Jun 10, 2024
0.9.0	May 13, 2024
0.8.0	Nov 26, 2023
0.0.1	Sep 13, 2019

#35 in Parser implementations

80,980 downloads per month
Used in **279** crates (83 directly)

MIT OR Apache-2.0 OR Zlib

255KB
4K SLoC

Dependencies

~2MB
~35K SLoC

- DEFAULT PARALLEL? hashbrown 0.15
  - PARALLEL? rayon
- ttf-parser **0.21**
  +opentype-layout

Other features

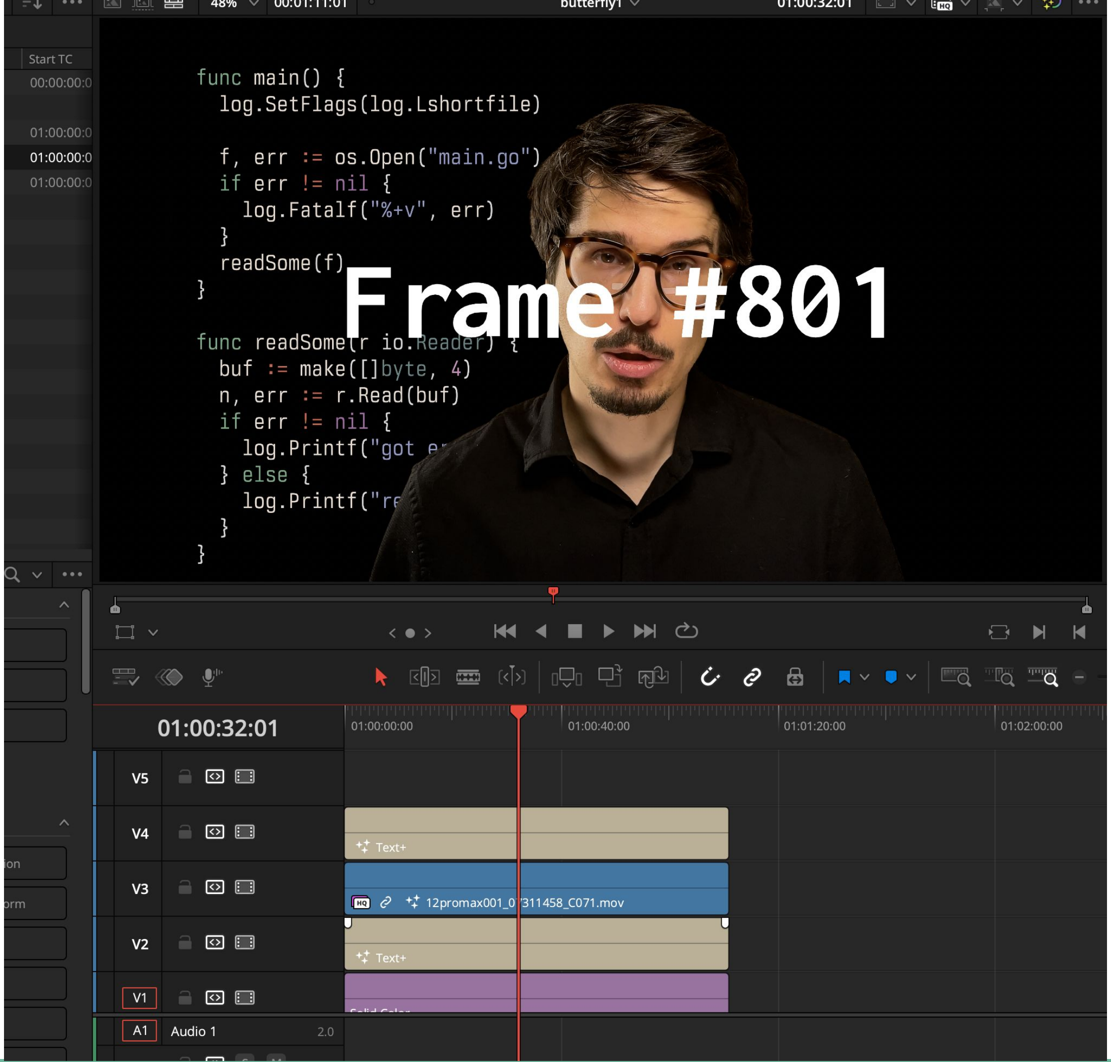
SIMD
STD

Fontdue

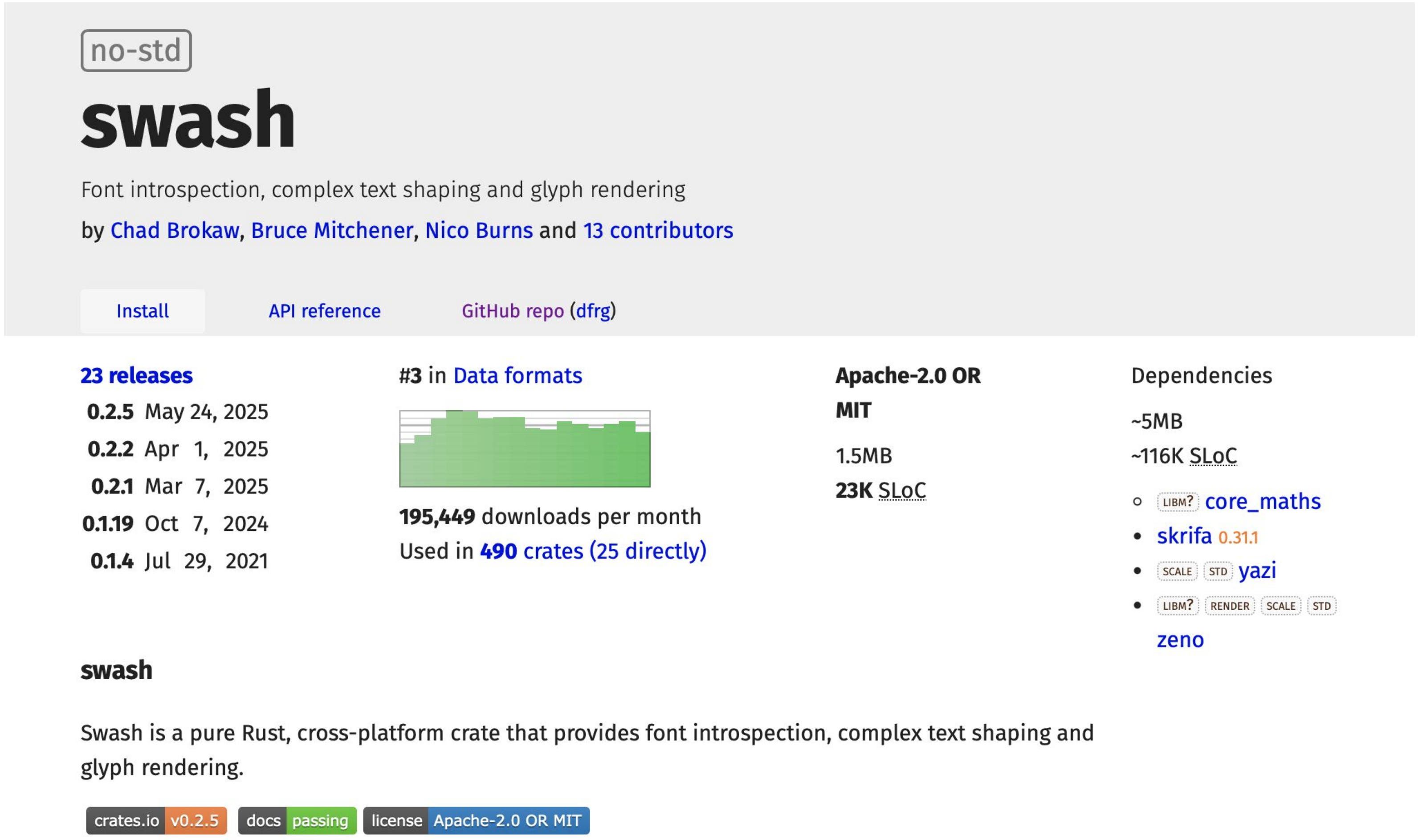
Build passing | docs passing | crates.io v0.9.3 | license MIT OR Apache-2.0 OR Zlib

Fontdue is a simple, `no_std` (does not use the standard library for portability), pure Rust, TrueType (`.ttf/.ttc`) & OpenType (`.otf`) font rasterizer and layout tool. It strives to make interacting with fonts as fast as possible, and currently has the lowest end to end latency for a font rasterizer.

self-directed research 🟩🟥🟦



```
func main() {
  log.SetFlags(log.Lshortfile)

  f, err := os.Open("main.go")
  if err != nil {
    log.Fatalf("%+v", err)
  }
  readSome(f)
}

func readSome(r io.Reader) {
  buf := make([]byte, 4)
  n, err := r.Read(buf)
  if err != nil {
    log.Printf("got er
  } else {
    log.Printf("re
  }
}
```



no-std

swash

Font introspection, complex text shaping and glyph rendering

by Chad Brokaw, Bruce Mitchener, Nico Burns and 13 contributors

Install | API reference | GitHub repo (dfrg)

23 releases

0.2.5	May 24, 2025
0.2.2	Apr 1, 2025
0.2.1	Mar 7, 2025
0.1.19	Oct 7, 2024
0.1.4	Jul 29, 2021

#3 in Data formats

195,449 downloads per month

Used in **490** crates (25 directly)

Apache-2.0 OR MIT

1.5MB

23K SLoC

Dependencies

~5MB

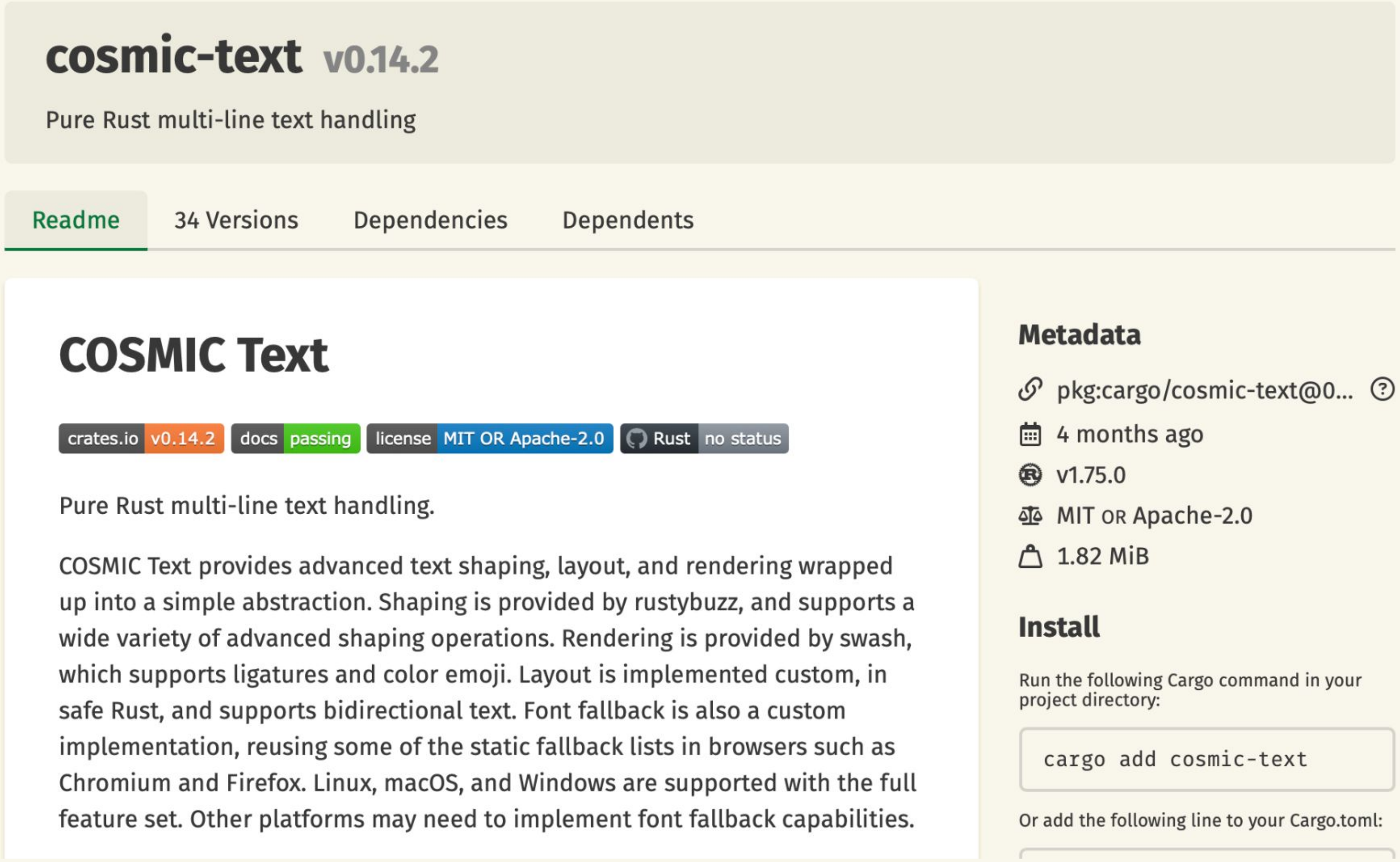
~116K SLoC

- LIBM? core_maths
- skrifa 0.31.1
- SCALE STD yazi
- LIBM? RENDER SCALE STD zeno

swash

Swash is a pure Rust, cross-platform crate that provides font introspection, complex text shaping and glyph rendering.

crates.io v0.2.5 | docs passing | license Apache-2.0 OR MIT



cosmic-text v0.14.2

Pure Rust multi-line text handling

Readme | 34 Versions | Dependencies | Dependents

COSMIC Text

crates.io v0.14.2 | docs passing | license MIT OR Apache-2.0 | Rust no status

Pure Rust multi-line text handling.

COSMIC Text provides advanced text shaping, layout, and rendering wrapped up into a simple abstraction. Shaping is provided by rustybuzz, and supports a wide variety of advanced shaping operations. Rendering is provided by swash, which supports ligatures and color emoji. Layout is implemented custom, in safe Rust, and supports bidirectional text. Font fallback is also a custom implementation, reusing some of the static fallback lists in browsers such as Chromium and Firefox. Linux, macOS, and Windows are supported with the full feature set. Other platforms may need to implement font fallback capabilities.

Metadata

pkg:cargo/cosmic-text@0...

4 months ago

v1.75.0

MIT OR Apache-2.0

1.82 MiB

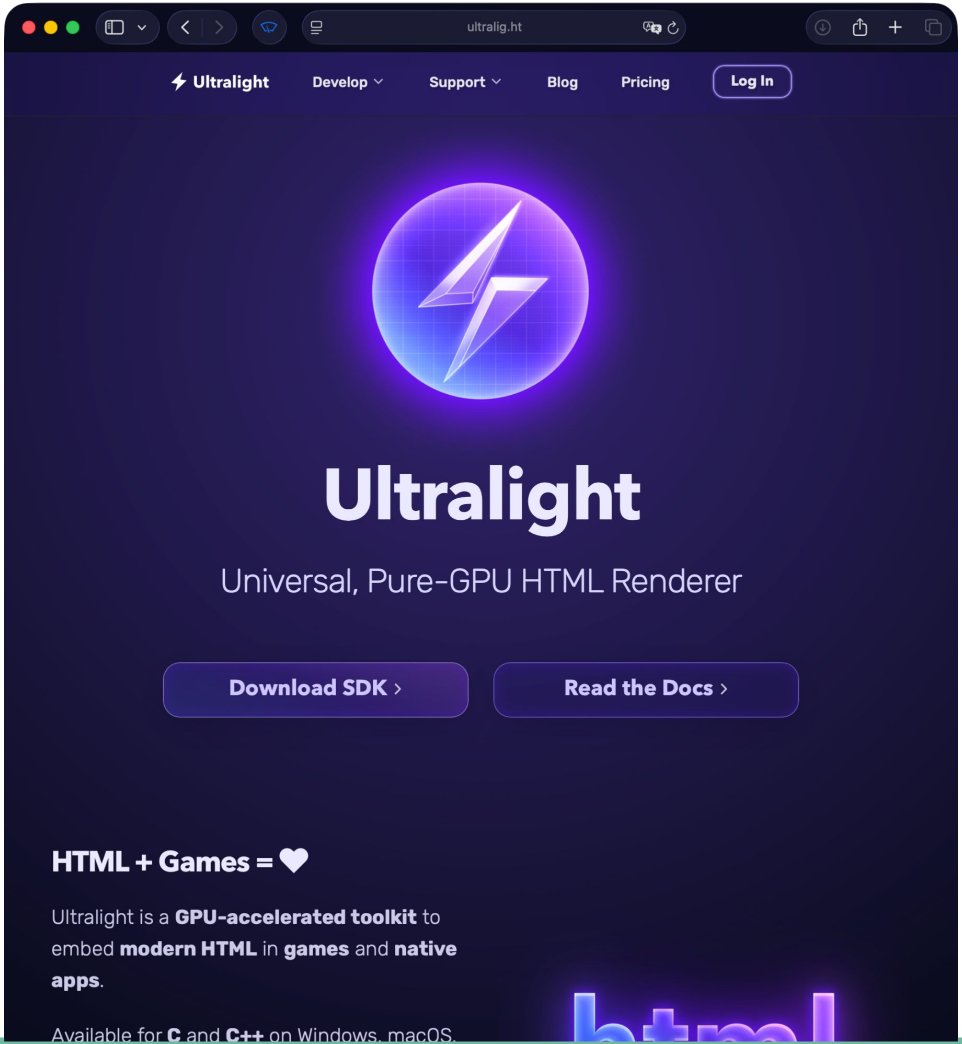
Install

Run the following Cargo command in your project directory:

```
cargo add cosmic-text
```

Or add the following line to your Cargo.toml:

What about browsers?



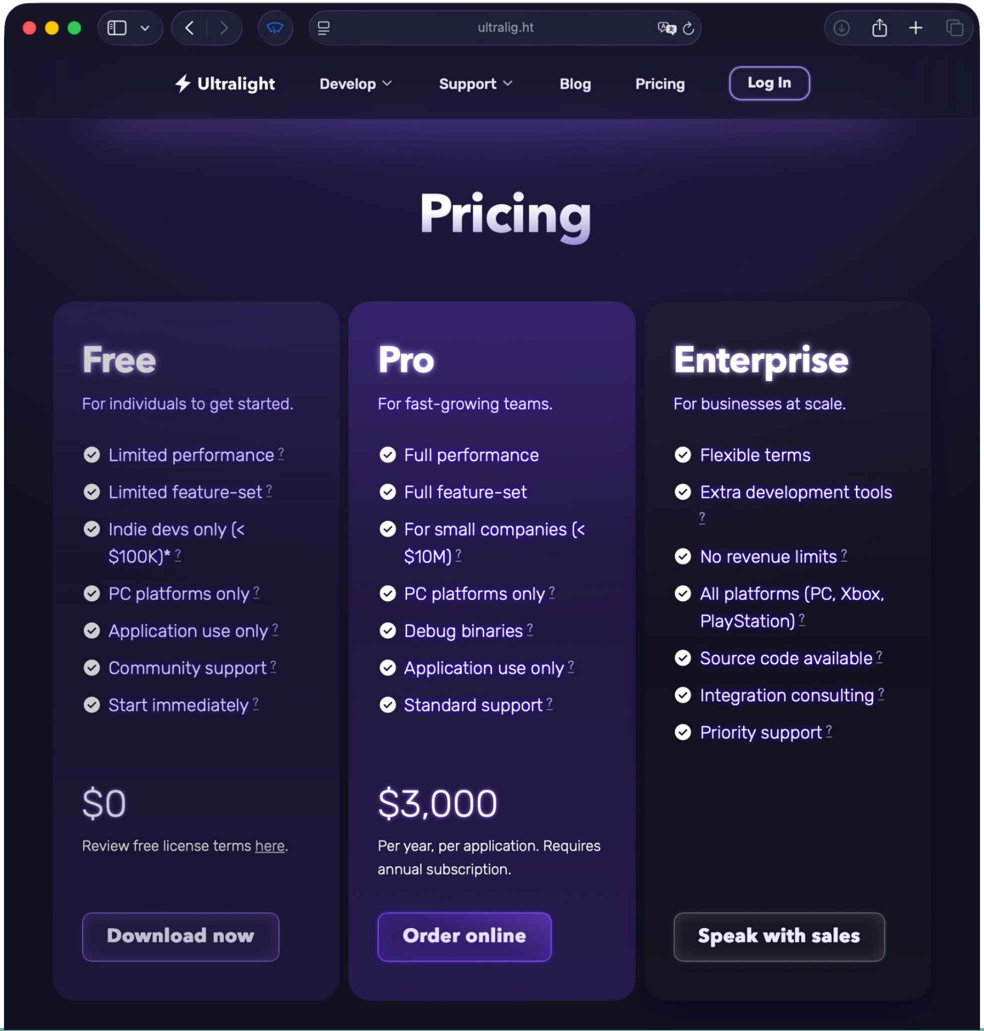
Ultralight

Universal, Pure-GPU HTML Renderer

HTML + Games = ❤

Ultralight is a **GPU-accelerated toolkit** to embed **modern HTML** in **games** and **native apps**.

Available for **C** and **C++** on Windows, macOS,



ultralig.ht

Ultralight Develop Support Blog Pricing Log In

Pricing

Free

For individuals to get started.

- Limited performance ?
- Limited feature-set ?
- Indie devs only (< $100K)* ?
- PC platforms only ?
- Application use only ?
- Community support ?
- Start immediately ?

$0

Review free license terms here.

Download now

Pro

For fast-growing teams.

- Full performance
- Full feature-set
- For small companies (< $10M) ?
- PC platforms only ?
- Debug binaries ?
- Application use only ?
- Standard support ?

$3,000

Per year, per application. Requires annual subscription.

Order online

Enterprise

For businesses at scale.

- Flexible terms
- Extra development tools ?
- No revenue limits ?
- All platforms (PC, Xbox, PlayStation) ?
- Source code available ?
- Integration consulting ?
- Priority support ?

Speak with sales

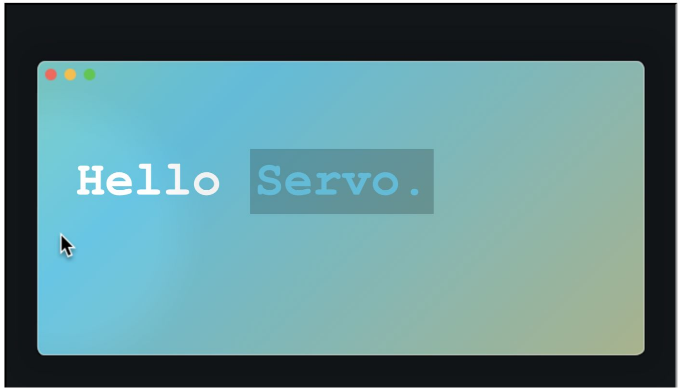
Servo?

2024-01-19

Tauri update: embedding prototype, offscreen rendering, multiple webviews, and more!

Overview of the embedding improvements we've landed as part of our collaboration with Tauri.

To integrate Servo with Tauri, we need to add **support for Servo in WRY**, the underlying webview library, and the developers of Tauri have created a proof of concept doing exactly that! While this is definitely not production-ready yet, you can play around with it by checking out the servo-wry-demo branch (permalink) and following the README.

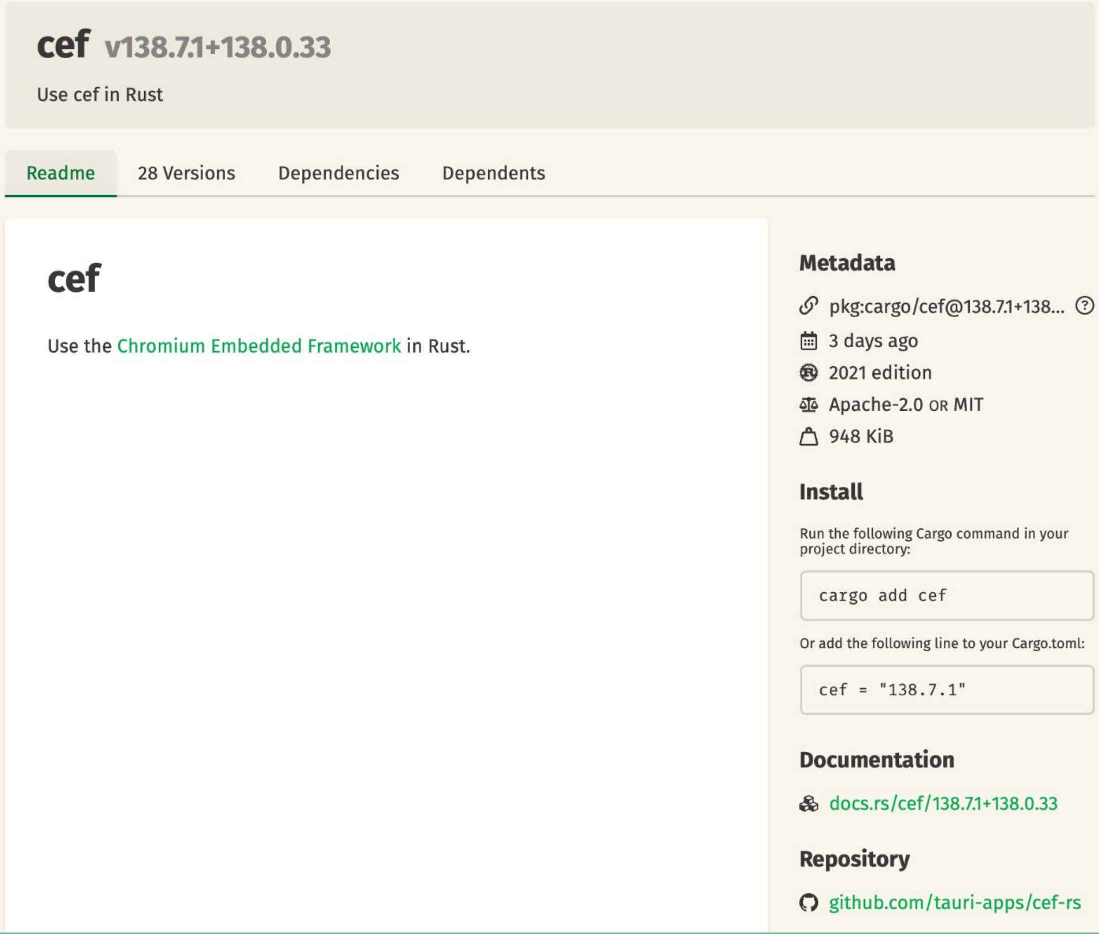


2025-02-19

This month in Servo: new webview API, relative colors, canvas buffs, and more!

Servo is becoming truly embeddable this year.

CEF?



cef v138.7.1+138.0.33

Use cef in Rust

Readme | 28 Versions | Dependencies | Dependents

cef

Use the Chromium Embedded Framework in Rust.

Metadata

pkg:cargo/cef@138.7.1+138...

3 days ago

2021 edition

Apache-2.0 OR MIT

948 KiB

Install

Run the following Cargo command in your project directory:

```
cargo add cef
```

Or add the following line to your Cargo.toml:

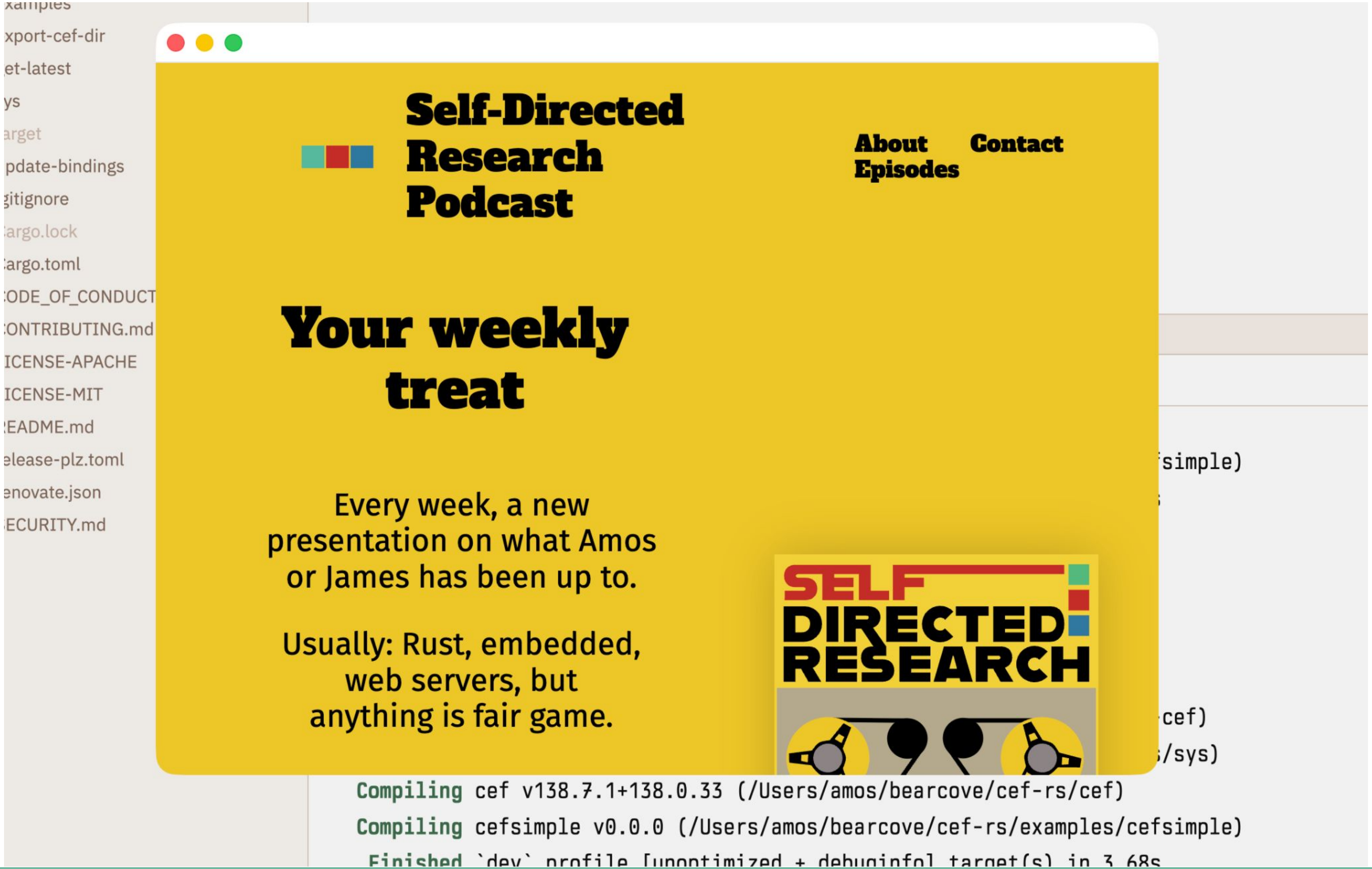
```
cef = "138.7.1"
```

Documentation

docs.rs/cef/138.7.1+138.0.33

Repository

github.com/tauri-apps/cef-rs



```
xamples
xport-cef-dir
et-latest
ys
arget
pdate-bindings
gitignore
argo.lock
argo.toml
ODE_OF_CONDUCT
ONTRIBUTING.md
ICENSE-APACHE
ICENSE-MIT
EADME.md
elease-plz.toml
enovate.json
ECURITY.md
```

Self-Directed Research Podcast

About **Contact** **Episodes**

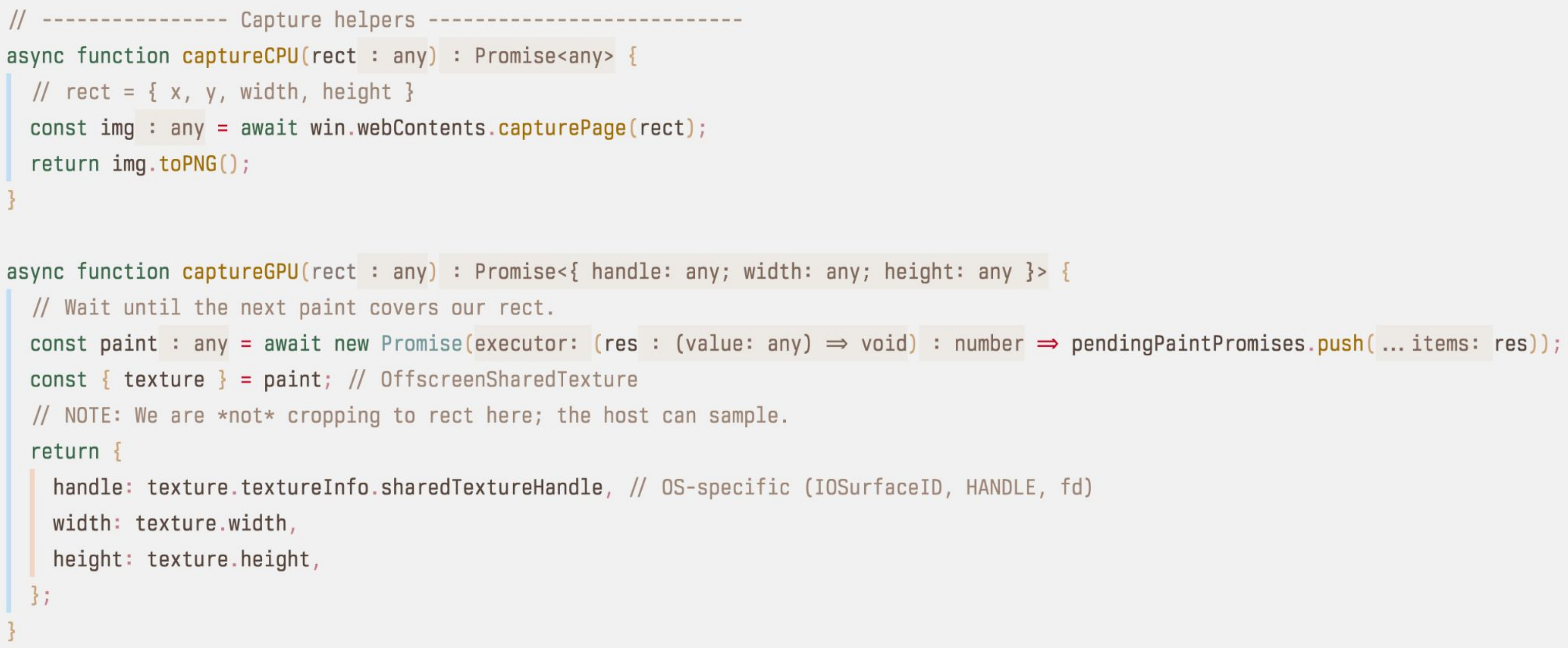
Your weekly treat

Every week, a new presentation on what Amos or James has been up to.

Usually: Rust, embedded, web servers, but anything is fair game.

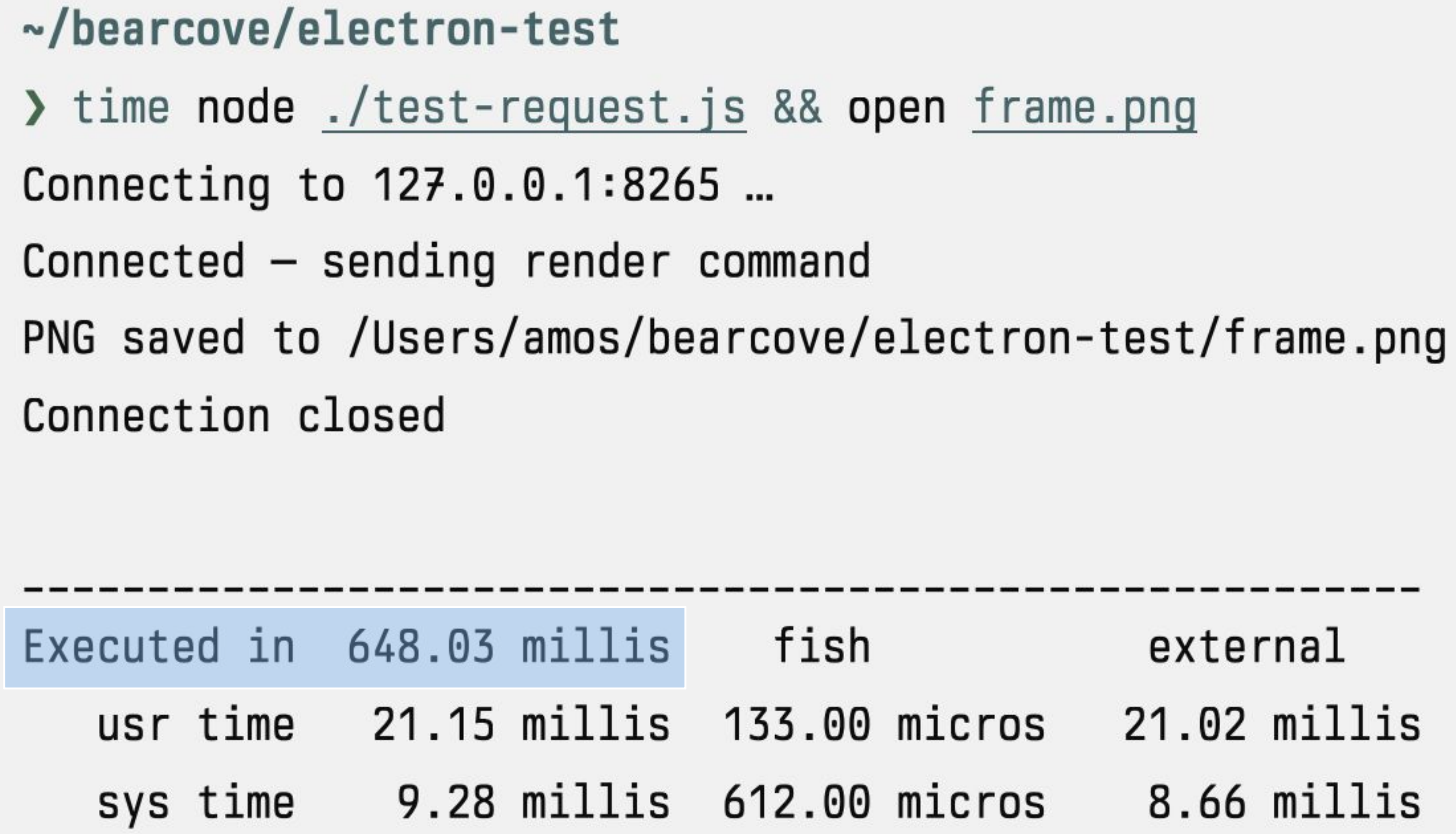
```
simple)

cef)
/sys)
 Compiling cef v138.7.1+138.0.33 (/Users/amos/bearcove/cef-rs/cef)
 Compiling cefsimple v0.0.0 (/Users/amos/bearcove/cef-rs/examples/cefsimple)
  Finished `dev` profile [unoptimized + debuginfo] target(s) in 3.68s
```

Electron?



```
// --------------- Capture helpers -------------------------
async function captureCPU(rect : any) : Promise<any> {
  // rect = { x, y, width, height }
  const img : any = await win.webContents.capturePage(rect);
  return img.toPNG();
}

async function captureGPU(rect : any) : Promise<{ handle: any; width: any; height: any }> {
  // Wait until the next paint covers our rect.
  const paint : any = await new Promise(executor: (res : (value: any) => void) : number => pendingPaintPromises.push(...items: res));
  const { texture } = paint; // OffscreenSharedTexture
  // NOTE: We are *not* cropping to rect here; the host can sample.
  return {
    handle: texture.textureInfo.sharedTextureHandle, // OS-specific (IOSurfaceID, HANDLE, fd)
    width: texture.width,
    height: texture.height,
  };
}
```

```
e on ᚠ main via 🦀 v1.88.0
❯ ls
binaries    Cargo.toml  clippy.toml  depot.json        Dockerfile  Justfile        README.md    repack.sh
Cargo.lock  CLAUDE.md   crates       docker-bake.hcl   docs        pnpm-lock.yaml  recipe.json  rust-toolchain.toml

home on ᚠ main via 🦀 v1.88.0
❯ gwS
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```



```
~/bearcove/electron-test
❯ time node ./test-request.js && open frame.png
Connecting to 127.0.0.1:8265 …
Connected — sending render command
PNG saved to /Users/amos/bearcove/electron-test/frame.png
Connection closed


________________________________________________________
Executed in  648.03 millis    fish           external
   usr time   21.15 millis  133.00 micros   21.02 millis
   sys time    9.28 millis  612.00 micros    8.66 millis
```

Next steps?