faustcode

Yann Orlarey, EMERAUDE

IFC 2026

2026-06-04

faustcode

A purely static, browser-native workbench for the Faust DSP language, typically deployed on GitHub Pages.

- orlarey.github.io/faustcode
- github.com/orlarey/faustcode
  - Edit, compile and run Faust entirely in the browser
  - Inspect diagrams, signals, tasks and live audio
  - Persist sessions in the browser's private filesystem
  - Expose the workflow to AI agents through **37 MCP tools**
  - Install its local MCP bridge directly from the webapp

No backend. No Docker. Just static files.

DSP View

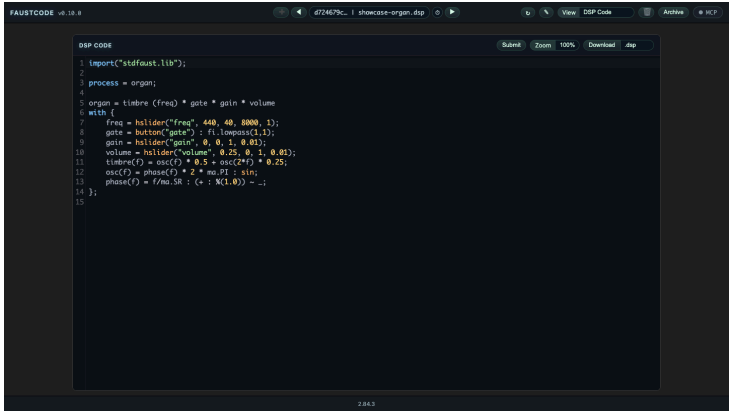
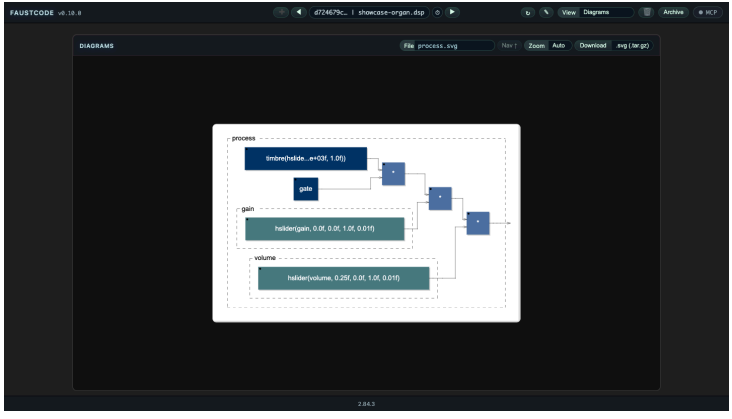
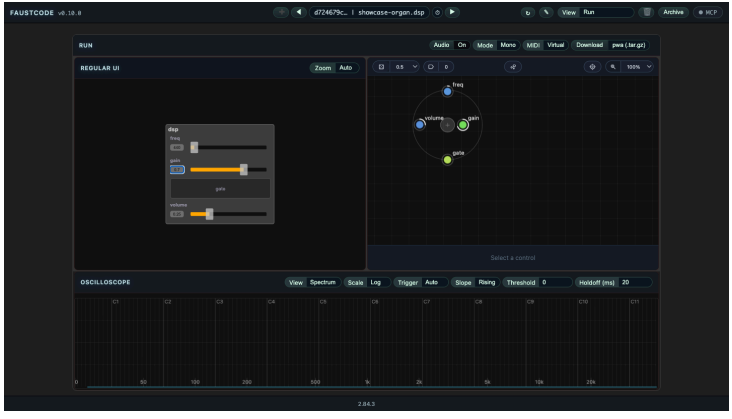

FAUSTCODE v0.10.8
d724679c... | showcase-organ.dsp
View
DSP Code
Archive
MCP
DSP CODE
Submit
Zoom
100%
Download
.dsp
import("stdfaust.lib");
process = organ;
organ = timbre (freq) * gate * gain * volume
with {
freq = hslider("freq", 440, 40, 8000, 1);
gate = button("gate") : fi.lowpass(1,1);
gain = hslider("gain", 0, 0, 1, 0.01);
volume = hslider("volume", 0.25, 0, 1, 0.01);
timbre(f) = osc(f) * 0.5 + osc(2*f) * 0.25;
osc(f) = phase(f) * 2 * ma.PI : sin;
phase(f) = f/ma.SR : (+ : %(1.0)) ~ _;
};
2.84.3

Diagram View

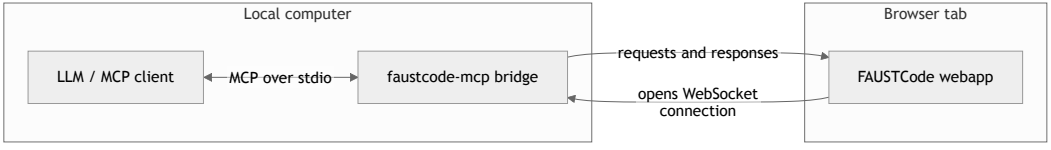


Run View

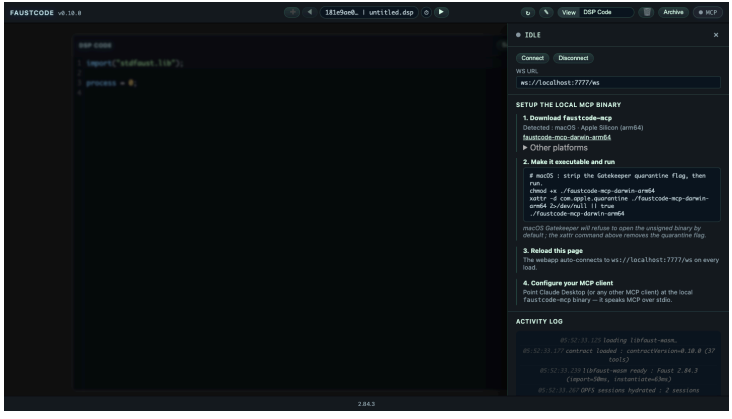

FAUSTCODE v0.10.8
d724679c... | showcase-organ.dsp
View
Run
Archive
MCP
RUN
Audio On
Mode Mono
MIDI Virtual
Download pwa (.tar.gz)
REGULAR UI
Zoom Auto
dsp
freq
gain
gate
volume
Select a control
OSCILLOSCOPE
View Spectrum
Scale Log
Trigger Auto
Slope Rising
Threshold 0
Holdoff (ms) 20
2.84.3

Need for a Local MCP Bridge

- Static web pages cannot expose an MCP stdio server
- Browsers cannot accept incoming WebSocket connections
- The browser opens an outgoing connection to `127.0.0.1:7777`
- The Go bridge translates MCP tool calls into WebSocket requests



Install Directly from FAUSTCode



Connect Claude Desktop

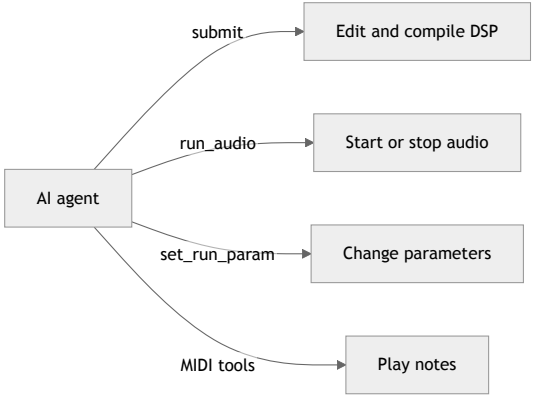
Add the local bridge to the Claude Desktop configuration:

```
{
  "mcpServers": {
    "faustcode": {
      "command": "/Users/yannorlarey/bin/faustcode-mcp"
    }
  }
}
```

Claude Desktop launches the bridge and communicates with it through MCP over `stdio`.

Keep a FAUSTCode browser tab open to complete the WebSocket connection.

What the Agent Can Do



Live Feedback: `get_spectrum`

Captures the DSP currently running in FAUSTCode.

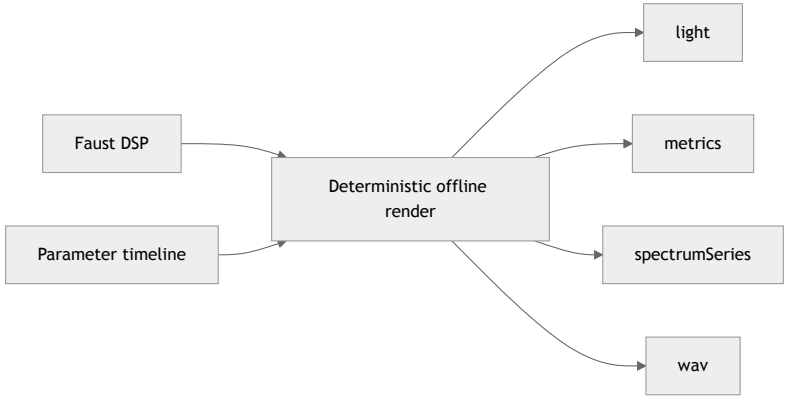
```
{"name":"get_spectrum","arguments":{}}
```

Returns one `spectrum_summary_v1`:

`f0 + peaks + noise bands + spectral features + audio quality`

Fast, current, non-deterministic. Not a continuous stream.

Offline Feedback: `render_audio`



The same DSP and timeline produce the same audio.

render_audio: light

One compact spectral snapshot near 80% of the render.

```
{"detail":"light","durationMs":2000}
```

Returns the same `spectrum_summary_v1` shape as `get_spectrum`.

Use for fast, reproducible iterations.

`render_audio: metrics`

Phased analysis of the complete sound:

```
attack → sustain → release
```

- envelope dynamics
- fundamental and harmonics
- HNR, roughness and spectral variation

Use for detailed timbre comparison.

`render_audio: spectrumSeries`

A time-indexed trajectory of compact spectral summaries.

```
SpectrumSeries ::= [(time, SpectrumSummary)]
```

Adaptive sampling keeps more frames where the spectrum changes.

Use to observe how the sound evolves.

`render_audio: wav`

Returns the rendered audio as a Float32 WAV file.

```
{"detail":"wav","durationMs":3000}
```

The MCP bridge writes the file locally and returns its path.

Use for listening or analysis by external audio tools.