SOCIETY FOR ARTS AND TECHNOLOGY

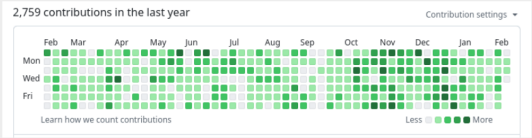
Dedicated to digital culture since 1996

Dr. Jean-Michaël Celerier

Pro
Directeur du développement technologique (SAT)
ossia.io
KDAB
Blue Yeti
Freelance & enseignement universitaire

Postdoc - MITACS Concordia x ossia.io
Thèse - Université de Bordeaux
“Authoring Interactive Media”

Carrière complète dans le logiciel libre 😎



water of Shape;

degrees of interoperability

Jean-Michaël Celerier • jmcelerier@sat.qc.ca

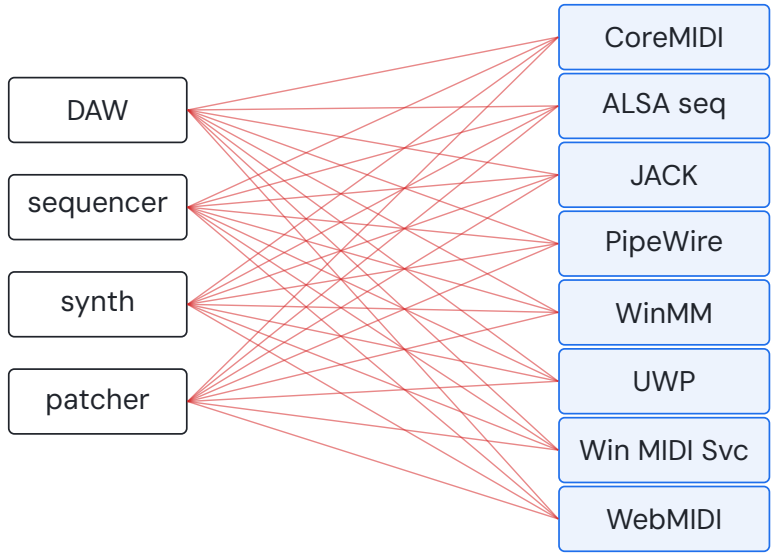
Société des Arts Technologiques

Linux Audio Conference 2026

Glue rot

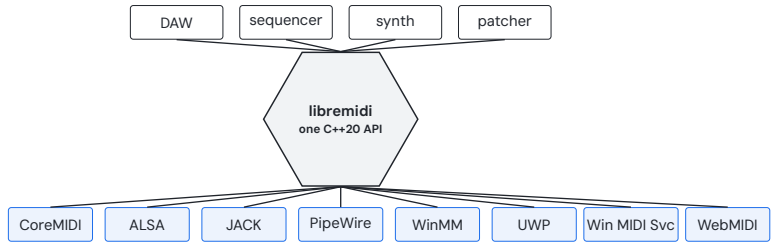
The bits are usually fine. The hand-written glue between the parts though?

$M \times N$: MIDI



Every MIDI app wired to every OS backend, by hand.

$M + N$: libremidi



One runtime API in the waist. MIDI 1 *and* MIDI 2 / UMP, every OS, written once; each app and each backend then meets it only once.

libremidi

Public

Sponsor Edit Pins Unwatch 24 Fork 78 Star 688

master 9 Branches 21 Tags Go to file T Add file <>

jcelerier semaphore: explicitly discard nodiscard try_acquire_for result 1c1c7ce · 2 weeks ago 737 Commits

.github	ci: bump clang-20 matrix entry to clang-22	2 weeks ago
.well-known	[core] Update manifest	2 years ago
bindings/python	python: fix various cases of callbacks not being freed	2 months ago
book	update book/src/cmake.md to match the tag that exists o...	2 weeks ago
cmake	winmidi: implement virtual port support. Fixes #194	2 weeks ago
examples	examples: move std includes before 'import libremidi' to f...	2 weeks ago
include/libremidi	semaphore: explicitly discard nodiscard try_acquire_for re...	2 weeks ago
src	backends: introduce shared pipewire layer	2 weeks ago
tests	python: fix various cases of callbacks not being freed	2 months ago
.clang-format	[ci] Many warning fixes	2 years ago
.clang-tidy	hygiene: clang-tidy cleanups	last year
.clangd	hygiene: clang-tidy cleanups	last year
.gitignore	android: implement android backend	11 months ago
AUTHORS	v5.4.0	5 months ago
CMakeLists.txt	v5.4.2	5 months ago
LICENSE.md	[license] Clarify the license contributions	3 years ago
README.md	v5.4.0	5 months ago
pyproject.toml	python: remove 3.15 from cibuildwheel	5 months ago

README License

A modern C++ MIDI 1 / MIDI 2 real-time & file I/O library. Supports Windows, macOS, Linux and WebMIDI.

midi webmidi uwp emscripten midi-api alsa jack coremidi jackaudio cpp20 midi2

Readme
View license
Activity
Custom properties
688 stars
24 watching
78 forks
Audit log
Report repository

Releases 10

v5.4.3 Latest
on jan 18

Sponsor this project

jcelerier Jean-Michaël Celerier

Sponsor

Deployments 33

github-pages 2 weeks ago
pypi 5 months ago

libremidi

A modern C++20/23 real-time MIDI 1 & 2 library.

A clean-room rewrite folding in RtMidi + ModernMIDI, with cmidi2 for UMP parsing.

- MIDI 1.0 *and* MIDI 2.0 / UMP behind one API
- Real-time *and* MIDI-file I/O; hot-plug observer
- Allocation-light RT path (vs RtMidi's SysEx allocations)
- Callback or manual-poll; per-backend configuration

~13 backends

CoreMIDI, ALSA (seq + raw), JACK, PipeWire, WinMM, UWP, *Windows MIDI Services*, Android, iOS, Emscripten/WebMIDI, keyboard, MIDI-over-network.

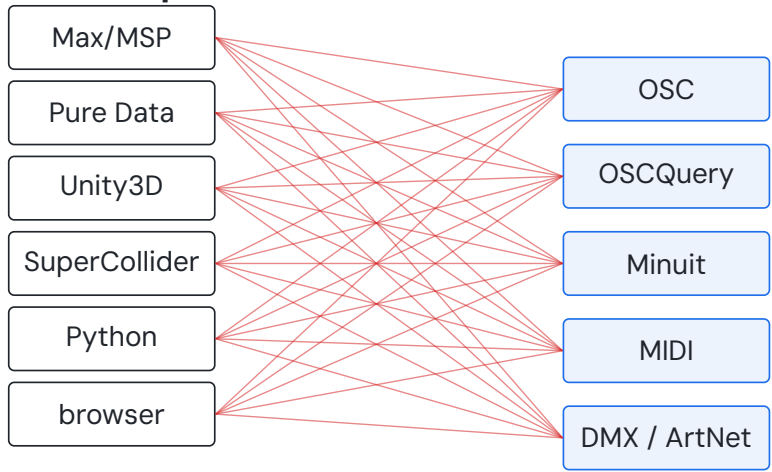
Tested everywhere

Win (x64 + arm64), Linux (gcc-13 ASAN/UBSAN, gcc-14, clang libc++), macOS, iOS, FreeBSD, Android NDK, Web, plus PyPI wheels.

Licence: new code BSD-2-Clause; built on RtMidi (MIT) + ModernMidi (BSD-2).

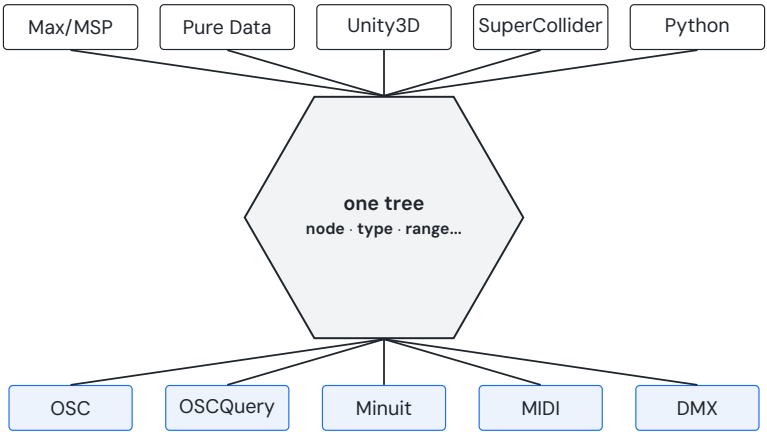
First to reach real-time MIDI 2 on macOS, Windows *and* Linux from one source. *libremidi*, SMC 2024.

$M \times N$: media arts protocols



Media-arts protocols are mostly run-time things: an OSC address space has no fixed schema, while OSCQuery at least lets a client discover one. Connecting every environment to every protocol by hand is again the same $M \times N$.

$M + N$: libossia


Max/MSP
Pure Data
Unity3D
SuperCollider
Python
one tree
node · type · range...
OSC
OSCQuery
Minuit
MIDI
DMX

libossia

Public

Sponsor Edit Pins Unwatch 25 Fork 35 Star 232

master 53 Branches 87 Tags Go to file T Add file

jcelerier cmake-modules: update to merged whole-archive lld fix f222c23 · last week 7,567 Commits

.cninja	[build] Add some more cninja configs	5 years ago
.github	ci: fix benchmark build	2 weeks ago
.vscode	[presets] Implement saveget / saveset / savebi attributes ...	4 years ago
.well-known	[core] Update manifest	2 years ago
3rdparty	kfr: update to 7.0.1	last week
ci	[ci] Continue cleanup	2 years ago
cmake	cmake-modules: update to merged whole-archive lld fix	last week
docs	Fix follow-up typos	4 years ago
examples	sockets: add UDP multicast support	2 weeks ago
minimal	esp32: finally supported properly	2 years ago
src	geometry: improve pcl support	2 weeks ago
tests	graph: better support for feedback	2 weeks ago
tools	[value] Use int32_t instead of int	3 years ago
.clang-format	[msvc] from_chars issues	2 years ago
.clang-tidy	Implement a custom parallel task-graph that does not us...	5 years ago
.gitignore	Adding the OSC_simple example, and adding .vscode to g...	2 years ago
.gitmodules	hash: migrate and uniformize around rapidhash	2 weeks ago
AUTHORS	code cleanup	10 years ago
CMakeLists.txt	ctest: try to understand runtime error	last year

A modern C++, cross-environment distributed object model for creative coding and interaction scoring

ossia.io

osc creative-coding midi c-plus-plus-14 floss open-sound-control ossia oscquery

Readme
LGPL-3.0, Unknown licenses found
Code of conduct
Contributing
Activity
Custom properties
232 stars
25 watching
35 forks
13 years old
Audit log

Releases 69

v2.0.0-rc6 Latest
on Jan 3, 2025

Sponsor this project

jcelerier Jean-Michaël Cele...

opencollective.com/ossia

Deployments 500+

libossia

A cross-environment distributed object model for creative coding.

Declare your app as a tree of nodes & parameters, with values, units and ranges, then expose and query it over the network. It is the engine under *ossia score*.

Protocols: OSC, OSCQuery, Minuit, MIDI, ArtNet (DMX), MQTT, libmapper, HTTP / WebSockets, serial, plus Joystick, Wiimote, Leap Motion, Phidgets, and Zeroconf discovery.

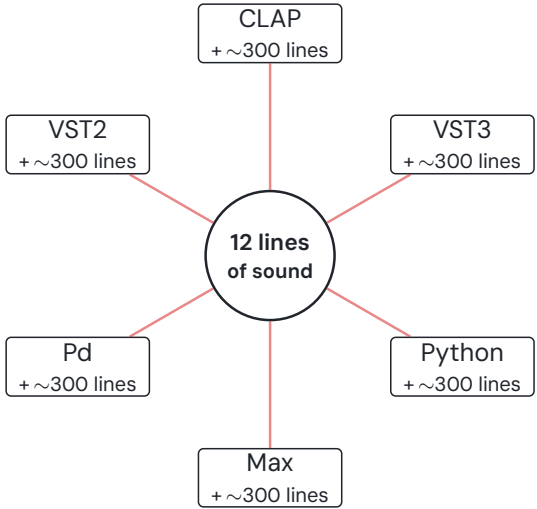
10 language bindings: PureData, Max/MSP, Python, C, C++, openFrameworks, Unity3D, QML, Faust, SuperCollider.

CI: Linux, macOS (x86_64 + arm64), Windows, Raspberry Pi (ARM cross), iOS.

LGPLv3 / CeCILL-C.

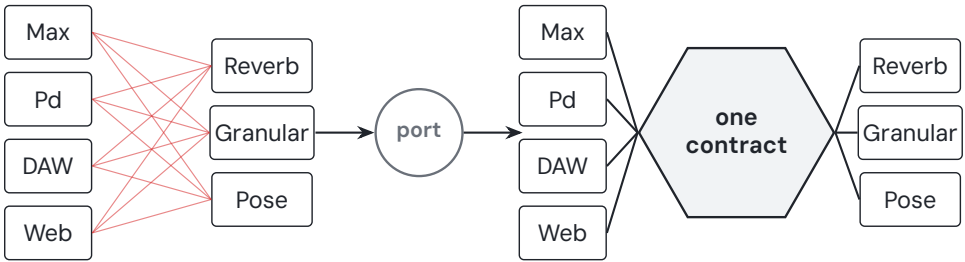
One object model, ten language bindings and a dozen-plus protocols, running from the desktop down to a Raspberry Pi.

$M \times N$: audio plug-ins / media objects



Twelve lines of DSP, a few hundred lines of glue, once per host. *Rage Against the Glue*.

$M + N$: Avendish



$M \times N$ **hand-written adapters**

$M + N$ **against one contract**

1984

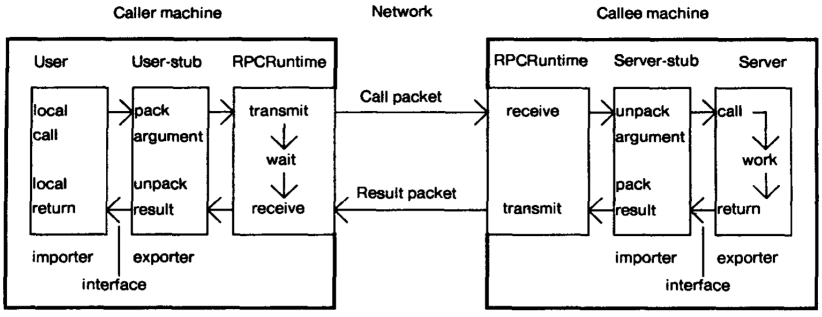


Fig. 1. The components of the system, and their interactions for a simple call.

invoked the procedure in the server directly. Indeed, if the user and server code

Birrell & Nelson, *Implementing Remote Procedure Calls*, 1984: the stub is code generated once from an interface description, so the caller writes a normal local call.

Every interop tool since is a stub generator. The contract moved (to the type), the binding

avendish

Public

Sponsor Edit Pins Unwatch 18 Fork 17 Starred 481

main 37 Branches 4 Tags Go to file T Add file

jcelerier and claude midi: fix make_command channel clamp and array-ba... 7eafe17 · 8 hours ago 897 Commits

.github	godot: headless GDExtension smoke tests, shared across ...	last week
.well-known	[core] Update manifest	2 years ago
book	mdbook: remove multilingual key	3 months ago
cmake	fuzz: add libFuzzer harness backend	8 hours ago
docs	examples: add a maxpat for testing jitter objects	8 months ago
examples	examples: port FilterGeometry to halp::dynamic_geometry	last week
include	midi: fix make_command channel clamp and array-backe...	8 hours ago
resources	Add a prototype of painter concept and painted / imaged...	4 years ago
src	[build] Some build fixes with _GLIBCXX_DEBUG	3 years ago
tests	bindings: update with changes in core API and concepts	4 months ago
.clang-format	[audio] Wip on passing various data such as tempo, bars,...	4 years ago
.gitignore	[pd] Handle variants as input in right-inlets	2 years ago
AvendishConfig.cmake	config: bootstrap ScoreExternalAddon when building aga...	last week
CMakeLists.txt	cmake: try to fix windows build	8 months ago
LICENSE	It's alive !	5 years ago
README.md	readme: enhance with new examples and backend info	8 months ago

README License

declarative polyamorous cross-system intermedia objects

audio c-plus-plus reflection multimedia declarative puredata vst maxmsp pd intermedia ossia

Readme
View license
Activity
Custom properties
481 stars
18 watching
17 forks
Audit log
Report repository

Releases 3

v3.0 Latest
on Nov 10, 2022

Sponsor this project

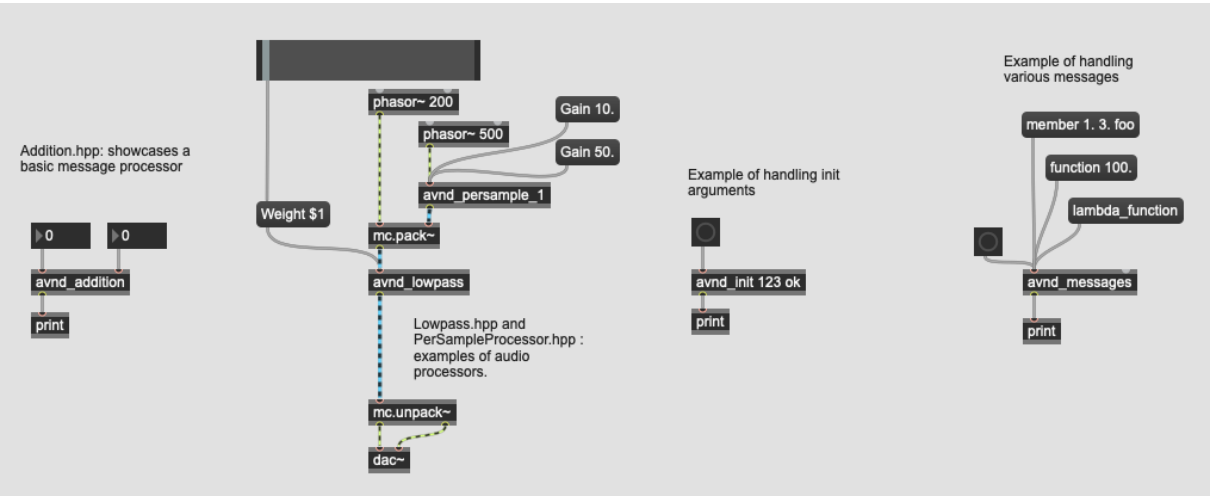
jcelerier Jean-Michaël Celerier

Sponsor

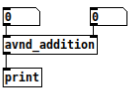
Deployments 113

github-pages 2 weeks ago

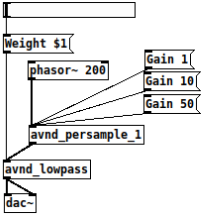
Contributors 8


Addition.hpp: showcases a basic message processor
0
0
avnd_addition
print
phasor~ 200
phasor~ 500
Gain 10.
Gain 50.
avnd_persample_1
Weight $1
mc.pack~
avnd_lowpass
Lowpass.hpp and PerSampleProcessor.hpp : examples of audio processors.
mc.unpack~
dac~
Example of handling init arguments
avnd_init 123 ok
print
Example of handling various messages
member 1. 3. foo
function 100.
lambda_function
avnd_messages
print

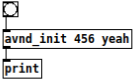
Addition.hpp: showcases a basic message processor



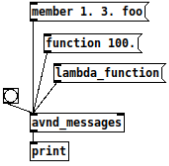
In pd, multi-channel must be handled by hand :-D
still, notice that Lowpass.hpp implements per-buffer
processing while PerSampleProcessor.hpp processes
per-sample.

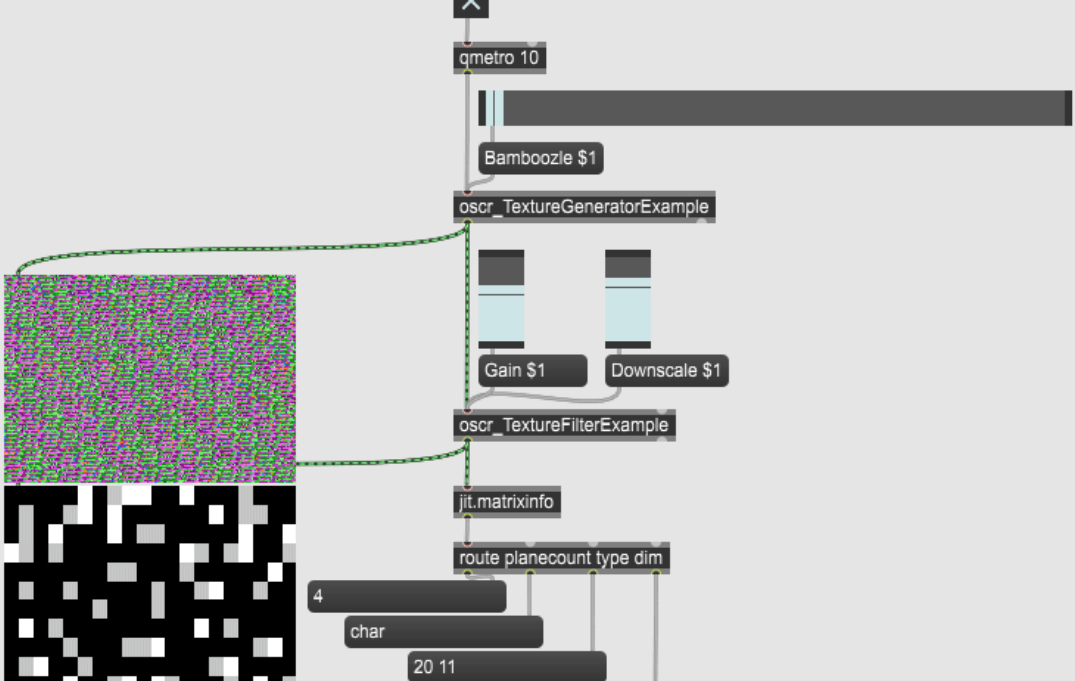


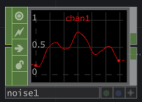
Example of handling init arguments (Init.hpp)

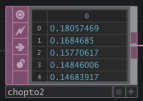


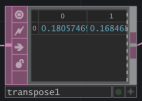
Example of handling various messages (Messages.hpp)

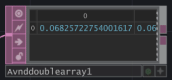


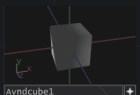

qmetro 10
Bamboozle $1
oscr_TextureGeneratorExample
Gain $1
Downscale $1
oscr_TextureFilterExample
jit.matrixinfo
route planecount type dim
4
char
20 11



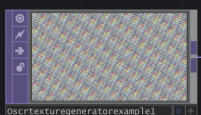


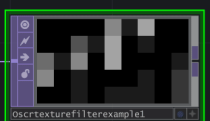




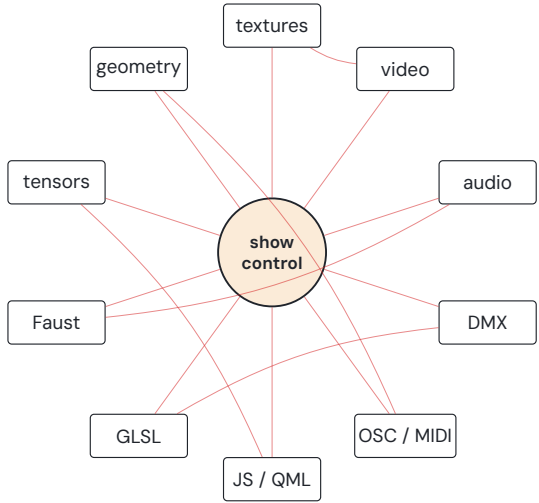


	0	1	2
0	index	P(0)	P(1)
1	0	-0.612	-0.612
2	1	0.612	-0.612
3	2	0.612	0.612
4	3	-0.612	0.612
5	4	-0.612	-0.612
6	5	0.612	-0.612
7	6	0.612	0.612
8	7	-0.612	0.612



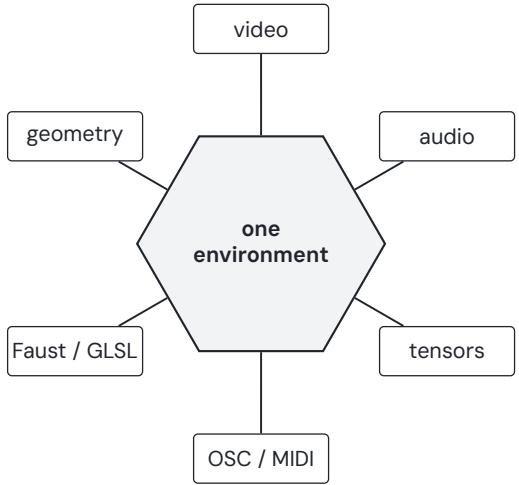


$M \times N$: data flows



How to combine all of it: signal processing, I/O protocols, every real-time media type (audio, video, textures, tensors, geometry), and several languages.

$M + N$ (stretching it a bit) : ossia score



Everything meets one contract, the typed and reflected object, and the host generates the rest. What we build in ossia then becomes reachable from every other environment as well.

Why all this?

Lazyness!

I'm a programmer: I want to do things *once*.

(And then it takes exactly fifteen years to get to that, just in time to be made moot by AI :))

score

Public

Sponsor Edit Pins Unwatch 56 Fork 123 Starred 2k

master 115 Branches 295 Tags Go to file T Add file

jcelerier presets: support hierarchic categories b24039a · 2 hours ago 13,035 Commits

.cninja	ci: more tries to disable kfr unity build	last week
.github	ci: replace the continuous release body instead of appen...	2 weeks ago
.well-known	[ci] Various fixes	2 years ago
3rdparty	presets: support hierarchic categories	2 hours ago
ci	sdk: use the resource dir of the compiler in use; ship Appl...	4 days ago
cmake	sdk: use the resource dir of the compiler in use; ship Appl...	4 days ago
docs	Fix various typos	4 years ago
src	presets: support hierarchic categories	2 hours ago
tests	[refactor] Standardize set / map containers, hash functio...	3 years ago
tools	[ci skip] fetch-sdk: unconditionally update cmake in sdk	last month
.clang-format	[avnd] minor fixes	4 years ago
.clang-tidy	[js] Improve GPU JS feature	2 years ago
.gitignore	add .claude to gitignore	3 weeks ago
.gitmodules	llfio: update to a more recent version	3 weeks ago
AUTHORS	[cleanup] Remove unused files and old CI stuff	3 years ago
CMakeLists.txt	windows: fix debug symbols file	last month
INSTALL.md	update ossia org name everywhere	6 years ago
LICENSE.txt	Update license with the current state of things	6 years ago
README.md	readme: minor improvements	4 months ago

ossia score, an interactive sequencer for the intermedia arts

ossia.io

art open-source qt osc creative-coding sequencer interactive midi digital-art osc-messages open-sound-control hacktoberfest intermedia media-art

Readme
View license
Code of conduct
Contributing
Activity
Custom properties
2k stars
56 watching
123 forks
13 years old
Audit log

Releases 279

v3.8.2 Latest
on Mar 27

Sponsor this project

jcelerier Jean-Michaël Cele...

opencollective.com/ossia


File Edit Object Play View Settings Help
DEVICE EXPLORER
Address Search
Address Value
Millumin
millumin
dlight
background [1, 0.889491, 0.926818, 0.231287]
circ
grandmaster 0
seq
go 0
fadeX1 0
fadeX2 0
X1 0
X2 0
sub
obs
scene
go
back
transOverrideType
fitToScreen
move
transOverrideDuration 0
movex 0
duplicateCurrentScene
side 0
movey 0
align 0
spin 0
Arturia MiniLab mkII
Artnet
H2000 Faze Machine
Smoke 0
Fan 0
Akai B-EYE K10
Linear CTO 0
Color Presets 0
Pan 0
Tilt 0
Maintenance 0
Reset 0
Zoom 0
Name Smoke
Address Artnet:/H2000 Faze Machine/Smoke
Value 0
Type Int
Min 0
Max 255
Values []
Unit none
Access <->
Bounding Clip
Repetition Filtered
Value Default 0
main-page /
Scenario1
Trigger the next part of the score when an external event happens
The white dots will send a cue with custom OSC, DMX, MIDI...
Execute some parts only when a condition is true
Sounds
and effects like Faust or VST
Freeverb
Gain
Bytebeat
State-machine-like and out-of-time scenes
As well as individual cues
Automations and tweening
Patch-based modulation and Javascript live-coding
Mapping (float)
Micromap
Javascript
vitalium
helm-synth
LFO
Units and dataspaces
PureData integration
example-synthesizer
Loops and unlimited nesting
pattern
Video and shaders
Crosshatch
Dither-Bayer
MIDI
Pattern sequencer
Arpeggiator
History
Sync.tree41
Event_swan54
State.tend47
State (State.tend47)
Label
Comments
Tree List
Address Value
Millumin
millumin
action
launchOrStopColumn 0
launchColumn 0
stopColumn 0
launchOrStopColumnWithName 0
layer
opacity
instance 100
x
instance 0
y
instance 0
rotation
instance 0
scale
instance 0
media
time
instance 0
timeInSeconds
instance 0
speed
instance 0
00:00:00.000
120.0
vol:-6.0 dB

ossia score

A sequencer for audio-visual artists, built to make interactive shows.

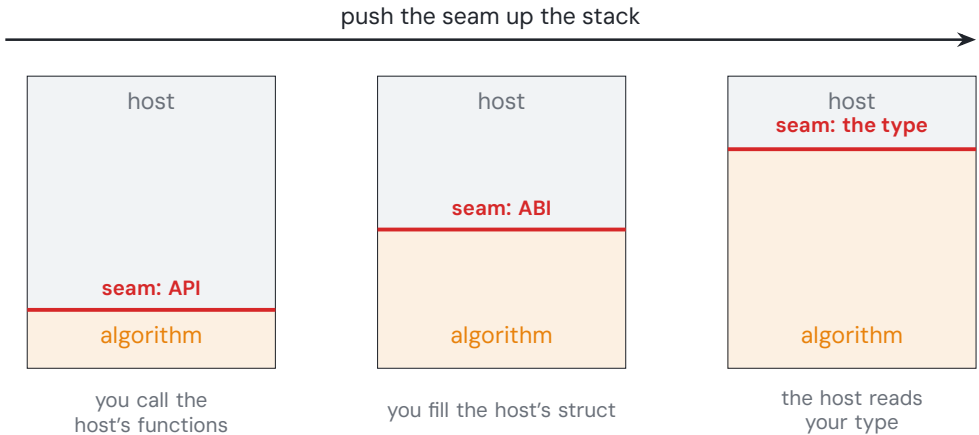
- Non-linear intermedia timeline: OSC, MIDI, DMX, sound, video
- Live-code *inside* the timeline: JS, ISF shaders, Faust, Pd, C++
- GPU video pipeline; loads any audio/video format
- Audio plug-in host: VST, VST3, LV2, JSFX
- Sonify data: CSV, HDF5, …
- Joysticks, Wiimotes, Leap, LSL, BLE; Spout / Syphon / NDI, Pipewire incoming

Runs everywhere
Windows, macOS (Intel + Apple Silicon), Linux (x86_64 + aarch64), FreeBSD, WebAssembly, and down to a *Raspberry Pi Zero 2*.

CI: Ubuntu/Debian (gcc, clang), macOS-15, Windows (MSVC + all MSYS2), FreeBSD, Emscripten; AppImage & Flatpak on x86_64 + arm64.

GPLv3.

The seam



Where amgp code meets the other party can sit at the call, at the binary, or at the type.

"The supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience."

Albert Einstein, *On the Method of Theoretical Physics*,
Herbert Spencer Lecture, Oxford, 1933

later flattened, on a thousand memes, to:

"Everything should be made as simple as possible, but not simpler."

(actually never quite said but we're FLUID)

In C, what is the *simplest* way to abstract an algorithm?

Say: a basic distortion, one controllable gain, on a single sample of audio.

Basic distortion simulation (ex. 1)

```
float behringer_metalzone_mt2(float input, float gain) {
  return std::tanh(input * gain);
}
```

Basic distortion simulation (ex. 2)

```
float behringer_multidisto(float input, float gain, int crush) {
  return (int)(crush * (std::tanh(input * gain))) / (float) crush;
}
```

What about a stateful algorithm?

```
float behringer_noise_reducer(float input, float gain) {
  // 1. Generate a random number
  static int seed = 4; // fair dice roll
  seed ^= seed << 13;
  seed ^= seed >> 17;
  seed ^= seed << 5;

  // 2. Add noise to the signal
  return input + (seed / (float)INT_MAX);
}
```

A lowpass filter in C++

```
struct behringer_lowpass {
  float prev = 0.f;

  float operator()(float input, float alpha) {
    prev += alpha * (input - prev);
    return prev;
  }
};
```

With state inside (ex. 4)

```
struct behringer_lowpass {
  float prev = 0.f;

  struct {
    float alpha = 0.5f;
  } inputs;

  float operator()(float input) {
    prev += alpha * (input - prev);
    return prev;
  }
};
```

With state inside (ex. 5)

```
[[=name("Lowpass")]]
struct behringer_lowpass {
  float prev = 0.5f;

  struct {
    [[=range(0.01, 2.0)]]
    float alpha;
  } inputs;

  float operator()(float input) {
    prev += alpha * (input - prev);
    return prev;
  }
};
```

One struct

```
struct Tremolo {
  struct {
    struct { float value = 4.f;  } rate;
    struct { float value = 0.5f; } depth;
  } inputs;

  void operator()(double* in, double* out, int n) {
    for (int i = 0; i < n; ++i) {
      phase += rate.value / 44100.;
      double ph = cos(2. * M_PI * phase);
      out[i] = in[i] * (1.0 - depth.value * 0.5 * (1.0 - ph));
    }
  }

  double phase = 0.0;
};
```

REACH 1 · LIQUID

Put it on the network

By hand

- OSC moves a value to an address: `/tremolo/rate 4.0`. Clean on the wire.
- But the receiver has to already know that address exists, its type, its range. So the map is kept by hand, on both ends, and the two never check each other.
- Rename `rate` (ex.: Laptop orchestras) and something across the room breaks silently.

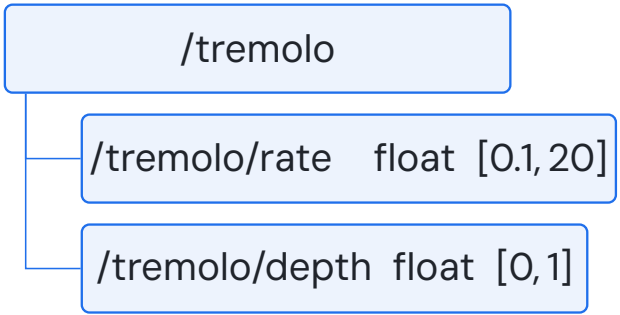
OSC carries the value but not its description, so the namespace ends up maintained by hand on both ends.

OSCQuery

- The device runs a small server. A client asks over HTTP and gets the whole tree: every node, type, range, unit. Subscribe over WebSocket; values stream.
- No hand-kept map on the client. One libossia implementation already talks to Max, Pd, Unity, SuperCollider, Faust, and a browser.

The contract is the address tree, discovered at runtime. At any moment it’s always possible to open a web browser to see what are the OSC addresses of the endpoint.

On the network



The struct never changed. “Something” else acted as the chair–computer glue.

MIDI 2.0

- MIDI 2.0 turns the cable into a conversation: capability inquiry, profiles, property exchange. `libremidi` parses the packets with zero allocation.
- 2026: native on macOS, Linux, and Windows at last. Same struct, a knob now discovers `rate` instead of being hand-mapped to it.

The controller and the device negotiate a contract instead of assuming one. *libremidi*, SMC 2024.

REACH 2 · LIQUID → ICE

Load it into a DAW

N plug-in ABIs, one algorithm

VST3

under proprietary control; MIDI arrives as parameter changes you reconstruct. Ugly duckling. COM ABI.

LV2

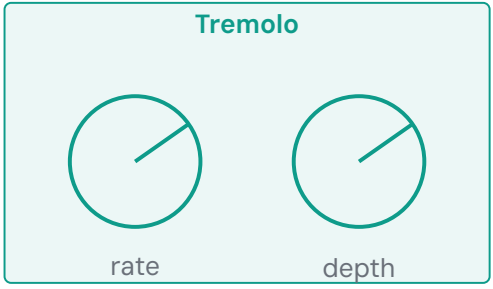
open; Turtle files, atom bus. Powerful, slow to evolve. Multiple concepts evolved over the years (event etc.).

CLAP

open C ABI, MIT, sample-accurate. The good one, and still one target of several.

An ABI is “the specification to which independently compiled binary entities must conform to be linked together and executed”: how to call functions, how data sits in memory, and where the metadata lives (Apple, *Swift ABI Stability Manifesto*). Three of them here, and even the best is one more target to generate rather than hand-write.

As a plug-in



Same struct, built as a `.clap`, dropped in the DAW. The two knobs are `rate` and `depth`.

The struct never changed. "Something" else acted as the chair-computer glue.

Avendish

- Write the algorithm as a plain C++ aggregate. Compile-time introspection reads it and generates the wrappers: VST3, CLAP, Pd, Max, Python, OSCQuery, GStreamer, Godot, ossia score, TouchDesigner, CSound (since yesterday) etc. The struct *is* the IDL, introspected in-language, with no preprocessor and no codegen.
- A volume control: **51 lines** vs **214** hand-written; the Steinberg VST3 gain example is **741**. Same binary size, same allocations, **101%** of the raw runtime (JUCE: 228%).
- Picked up by others: **Sygaldry** (West, PhD) runs it on bare-metal ESP32, a T-Stick in **15** instrument-specific lines instead of 1052; **BRT** ships one psychoacoustic model to three hosts unchanged.

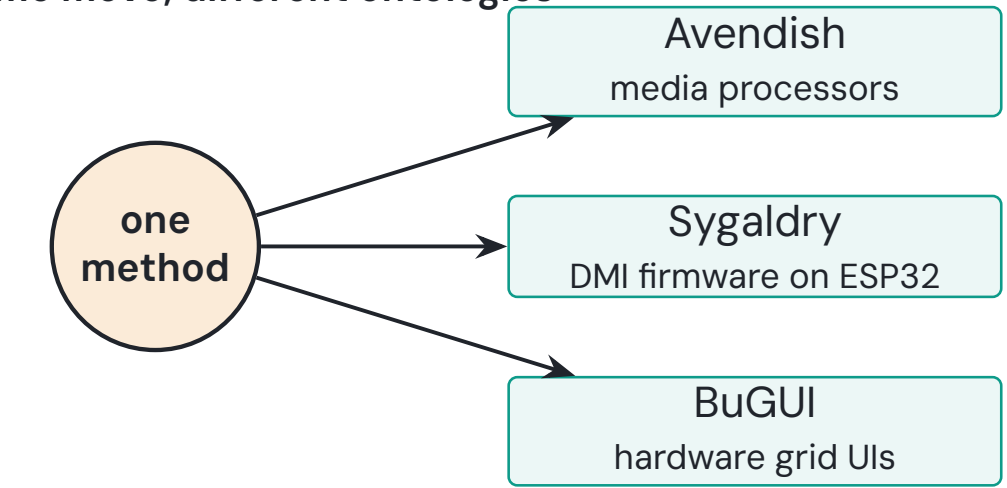
The network endpoint and the CLAP came from the same twelve lines. *Rage Against the Glue* (ICMC 2022); the full method in *Avendish: a modern C++ solution to the multimedia plug-in API problem*.

It travels

- The object depends on no host headers and no runtime, only the language. So it goes wherever a C++ compiler goes: a VST in a DAW, a Max object, a node in *score*, and the same source, unchanged, onto an **ESP32**, into **WebAssembly**, even **eBPF** in the kernel.
- Roughly **200+** such objects today. Every new ossi

Same source, no dependency to rot. That is also exactly what reproducible science needs: the experiment still runs, identically, years on, on hardware that did not exist when it was written.

Same move, different ontologies



Define a generic contract, and let many targets bind against it. The only thing that changes per domain is the *ontology*: processors, DMIs, grid widgets. *BuGUI* (Keller, Lazzarini, Delap, Celérier, LAC 2025) is this method one floor over, and *Sygaldry* (West) is it on bare metal.

Avendish, in one slide

A zero-cost, compile-time, reflection-based pure-C++ answer to the $M \times N$ glue problem.

Write one plain C++ class ("effects don't even include a header") and it generates the bindings, at **zero run-time cost** over equivalent C.

- Audio effects, synths, MIDI; CPU & GPU video; geometry; ML
- Thread-safe message buses, sample-accurate controls, polyphony
- Boost.PFR + concepts today; a C++26 `std::meta` branch

Generates: VST3, CLAP, Pd, Max/MSP, Max Jitter, TouchDesigner, GStreamer, Godot, Python, OSC/OSCQuery, ossia score, Qt/Nuklear UIs, GPU draw & compute, plus a JSON dump of any object.

200+ objects today. Success story: jit.ultraleap.

CI: Linux (clang-20 libc++, gcc-14 + GStreamer), macOS arm64 (Xcode 16.4), Windows (MSVC 2026 + MSYS2).

GPLv3 (+ permissive concept headers).

One struct, a dozen-plus ecosystems, no runtime tax. *Rage Against the Glue* (ICMC 2022).

One struct, every backend, for real

From the one `Tremolo` struct, `avnd_make_all` produced these, with no host headers, one compile each, on this laptop:

```
clap/avnd_tremolo.clap      # CLAP
vintage/avnd_tremolo.so     # VST2
pd/avnd_tremolo             # Pure Data
python/pyavnd_tremolo.so    # Python
gstreamer/avnd_tremolo.so   # GStreamer
godot/avnd_tremolo(.so,_fx)# Godot
standalone/Tremolo          # app
dump/Tremolo  ->  JSON
```

```
"name": "Rate",
"parameter": {
  "range": {"min":0.1,"max":20,"init":4},
  "value_type": "float",
  "widget": "slider"
}
```

the JSON dump: the struct *is* the schema, read straight back out.

Ten artifacts from one source (VST3 / Max / TouchDesigner / OSCQuery follow with their SDKs). `rate` and `depth` are still the only two fields.

Developer experience

- **Fast to compile:** it is just a struct; no SDK headers dragged in per target.
- **One introspection, many uses:** the reflection that emits a CLAP also dumps the object as JSON, so `.maxref.xml` generation, fuzz harnesses and unit tests come for free.
- **Declarative, so hard to crash:** wrong code fails to compile, not on stage.

And because it is this small, we can write one together, live, at the end.

REACH 3 · ICE

Hand it to another language

Speaks in tongues

- *score* already hosts a spread of languages, from the very specific to the very general: **Faust**, **GLSL**, **ExprTK**, **JSFX**, **Bytebeat**, `jq`, and **JavaScript/QML**, **C++**. Each part of a problem gets the language that fits it.
- And this has a theory: one program, two semantics, soundly joined at a boundary (Matthews & Findler, 2007; *Multi-Language Probabilistic Programming*, OOPSLA 2025, where the interoperating program is *an order of magnitude* more efficient than the single-language baseline, Pyro).

Pick the best language for each task, and still meet at one contract / interface / boundary.

Why not just Faust?

- A DSL like Faust is superb at what it is *for*, and that is exactly the point: it is for DSP. Many host APIs are object-oriented or procedural; they pass pointers and arbitrary data types, parse XML, drive OpenGL, talk to a proprietary device.
- Faust does not, and should not, have to do any of that. The binding layer needs a language that already speaks objects, pointers, and the messy real world.

A DSL wins inside its domain and stops at its edge. The glue problem lives at those edges, so the binding layer itself has to be a general language.

Why C++

- Performance, without a ceiling. The very reflection that reads the struct can also feed a JIT: cppyy on Cling/Clang speeds Python up by **2–20**×, because it inherits Clang's whole optimisation pipeline for free.
- Stacking reverbs and distortions in a live set, you do not want to feel the language pushing back.

Benchmark Case	Cppyy JIT time (s)	Numba JIT time (s)	Hot run time (s)	Python run time (s)	Speedup
Function w/o args	1.72e-03	3.33e-01	3.58e-06	1.73e-05	4.84×
Overloaded functions	1.05e-03	1.35e-01	4.51e-06	3.47e-05	7.70×
Templated free functions	8.92e-04	1.45e-01	3.48e-06	7.18e-05	20.66×
Class data members	1.43e-06	1.33e-01	5.87e-06	1.82e-05	3.10×
Class methods	2.16e-03	1.39e-01	6.06e-06	1.43e-05	2.36×

cppyy through Numba vs pure Python: 2.36× to 20.66× (Kundu, Vassilev & Lavrijsen, arXiv:2304.02712, Tab. 1).

C++ is the first place we get both at once: late, reflective flexibility and native speed. Every other corner of the dial makes you give one of them up.

Schema-first

Forty years of "write the interface once, generate the bindings." All of them ask for a separate schema file kept in sync by hand. **We can do better!!!!!**

The struct is the schema

```
template <typename T>
consteval auto describe() {
  for (auto m : nonstatic_data_members_of(^^T))   // ^^  reflects
    emit_parameter(identifier_of(m), type_of(m)); // [: :]  splices
}
describe<Tremolo>();   // -> rate:float, depth:float ... at compile time
```

- Reflection (P2996) is in C++26, voted in at Sofia, June 2025; ISO standard March 2026. `^^` reads the type; `[: :]` writes code back.
- Annotations are in too (P3394): `[[=meta::…]]` lets the author tag a field, so the binding reads intent, not just layout. The Avendish C++26 branch runs on this.
- No separate schema, no `protoc`. The compiler reads the type and writes the bindings. (Synthesising whole new *functions*, by token injection, P3294, is the next step, landing in C++29.)

Three targets, one compile

```
generate_oscquery<Tremolo>();    // -> /tremolo/rate, /tremolo/depth
generate_clap_params<Tremolo>();// -> two CLAP parameters
generate_python<Tremolo>();      // -> import tremolo; t.rate = 4.0
```

One `c++ -std=c++26`. Out come the network node, the plug-in, and the Python module, all from `rate` and `depth`.

The struct never changed. “Something” else acted as the chair-computer glue.

The compiler reads the struct

Real output, GCC 16, `-std=c++26 -freflection` (`code/example-2`, `code/example-5`):

```
$ ./distort                  # CLI synthesized from the function's parameters
usage: ./distort <input:float> <gain:float> <crush:int>

$ ./annotations 5.0          # reads the [[=name]] and [[=range]] annotations
object: "Lowpass"
  alpha: range [0.01, 2]
alpha 5 out of range -> clamped to 2
```

Parameter names, types and ranges are all recovered by the compiler from the declaration, with no schema file. Built on `nonstatic_data_members_of` and `annotations_of`, and runnable today on GCC 16.

Declarative, therefore safe

- Reflection only reads structs and variable definitions: compiler primitives, nothing imperative. So the contract is *declarative*; and declarative is safe by construction: invalid code does not run, it does not even pass the syntax check.
- Compare the other way: a hand-written Pd or Max external, full of pointers and registration calls, checked by nothing until it crashes on stage.

Who here has never crashed while building a Pd external? Leaning on the language's own static semantics moves whole classes of bug from run time to compile time, which is to say from the stage back to the keyboard.

One VM, many languages

```c
#include <truffle.h>

typedef struct {
  int value;
} S;

S *obj = //...
Export("obj", obj);
```

Listing 4. This C snipped exports the variable `obj` which points to a `struct` of type S.

```javascript
var obj = Truffle.import("obj");
var a = obj.value;
```

Listing 5. This JavaScript snipped imports a variable `obj` and accesses its `value` member.

GraalVM does it at the other end of the dial: C exports a struct, JavaScript reads `obj.value` with no marshalling, and the JIT optimises straight through, so that “language boundaries are completely transparent to the compiler” (Grimmer et al., TOPLAS 2018).

Convergence

WebAssembly's WIT reaches the same conclusion from its corner: the contract is the code, the runtime writes the glue. Carbon builds a whole new language to avoid this loop. That is a lot to pay.

Everyone is doing it

UniFFI (Mozilla): one Rust source → Kotlin / Swift / Python:

```
#[uniffi::export]
fn add(a: u32, b: u32) -> u32 { a + b }     // bindings generated, no glue
```

Diplomat: one annotated Rust module → C / C++ / JS / Dart:

```
#[diplomat::bridge]
mod ffi { struct Tremolo { /* rate, depth */ } }   // one source, many targets
```

C++26 reflection: the struct *is* the `.proto`:

```
struct Point { int x; int y; };
// nonstatic_data_members_of(^^Point) -> a proto3 (de)serializer, no protoc
```

Rust's binding generators and C++26 reflection are the same idea from two directions: one declarative source of truth, with the machine writing the glue, and the separate IDL file then disappears.

REACH 4 · VAPOR

Let something use it that does not know its types

LSP, then MCP

- LSP (2016) fixed editor × language: the language server became the one contract. That is M times N to M plus N, and it is why your editor handles languages its authors never touched.
- MCP does it for app × tool. A server describes its tools; the model discovers them and writes the call. Donated to the Linux Foundation, Dec 2025.

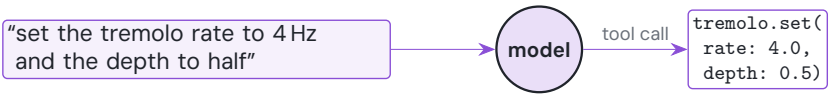
The same decoupling, one layer up. Turns the contract into a description, and the thing writing the glue is a model.

ossia score, as MCP tools

- **score-mcp** runs an MCP server inside ossia score (on `cpp-mcp`). It serves the live document as **resources** (`score://document/graph`, `score://devices/{name}/tree`, `score://introspection/processes/{Key}`) and as **tools** to browse, search and edit, each tagged with MCP read-only / destructive / idempotent hints.
- A model drives it: Claude over MCP, *or* score's own in-app LLM panel dispatching the same tools in-process: *"add a reverb to cue 3, dim the dome."*
- The same introspected tree that sits behind OSCQuery, projected once more: the LSP→MCP move, on a live patch.

The contract that drove the network node and the plug-in drives the model too: the document's own tree, re-served as MCP resources and tools.

In plain language



Plain language becomes a tool call against the same two parameters. The tool schema came from the struct too.

The struct never changed. “Something” else acted as the chair–computer glue.

Why all this ? Free access to creation.

Everyone, whatever their skill or their means, should be allowed to make things.

Compile the model in

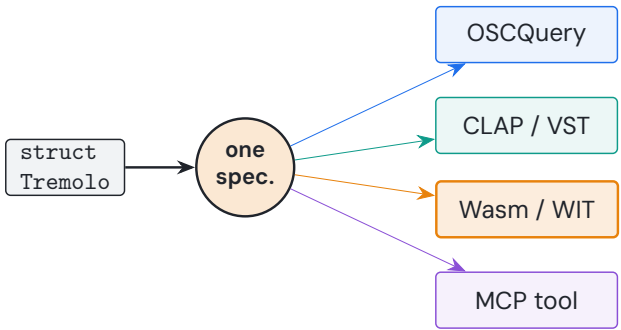
- The Inference Zoo extends Avendish to model inference. The same reflection that read the Tremolo reads a model's inputs and outputs and wraps `onnxruntime` as one portable C++ object.
- BlazePose, DepthAnything2, a small Qwen: each becomes a node in ossia score, no Python environment to rot, deterministic on stage. Likewise for diffusion models.

Leverage our abstractions to built resilient media object implementations.

Review

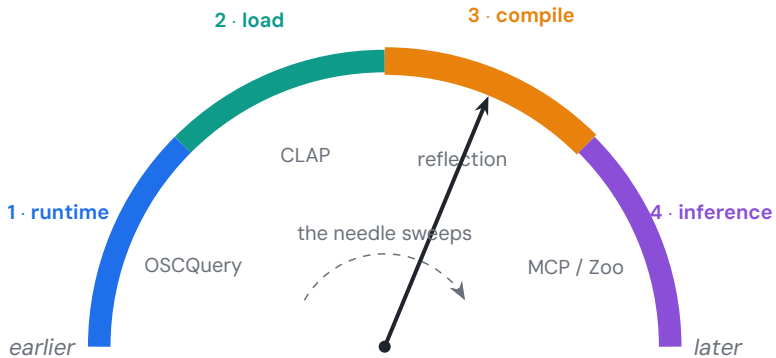
reach	wanted	who wrote the glue
1	it on the network	the protocol stack
2	it in a DAW	the loader (Avendish-generated glue)
3	it in another language	the compiler
4	it to follow plain language	the model

One move



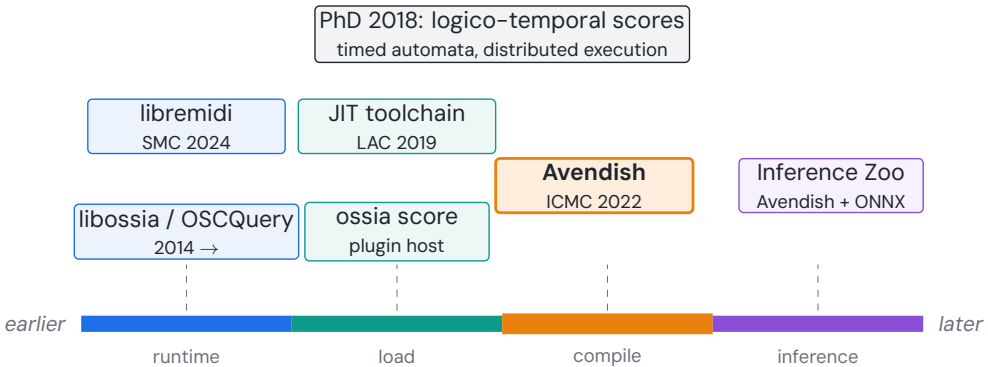
The only thing that changed each time was *when* the glue got made.

Binding time



when is the contract resolved?

One at a time


PhD 2018: logico-temporal scores
timed automata, distributed execution
libremidi
SMC 2024
JIT toolchain
LAC 2019
Avendish
ICMC 2022
Inference Zoo
Avendish + ONNX
libossia / OSCQuery
2014 →
ossia score
plugin host
earlier
later
runtime
load
compile
inference

ACT V

How fluid must we be?

The world won’t hold still

- Reflection fixes the shape at compile time, on purpose. Avendish says so in as many words: the number of ports “cannot vary at run-time.”

It’snot “how early can we bind?” but “how late do we still need to bend?”

LV2 is an ontology

- LV2 describes its ports in RDF/Turtle, a real ontology, ranging from a trivial float port to deep semantic class trees.
- An ontology is “an explicit specification of a conceptualization … that agents commit to” (Gruber, 1993).

Precision vs freedom

- Too precise a *host* API forecloses the UI. GStreamer caps, PipeWire's graph, LV2's `.ttl`: the tighter the port spec, the less room left to invent the experience that makes someone pick *your* tool over another (remember the seam going up in the stack).
- Yet the more precise the *algorithm author* is, the better the generated binding. Let the author use the data type that fits the algorithm, rather than the one the host forces them to coerce into.

Put the precision at the source, where meaning (beautiful DSP) lives. Keep the vessel (the host) loose, where the UI breathes.

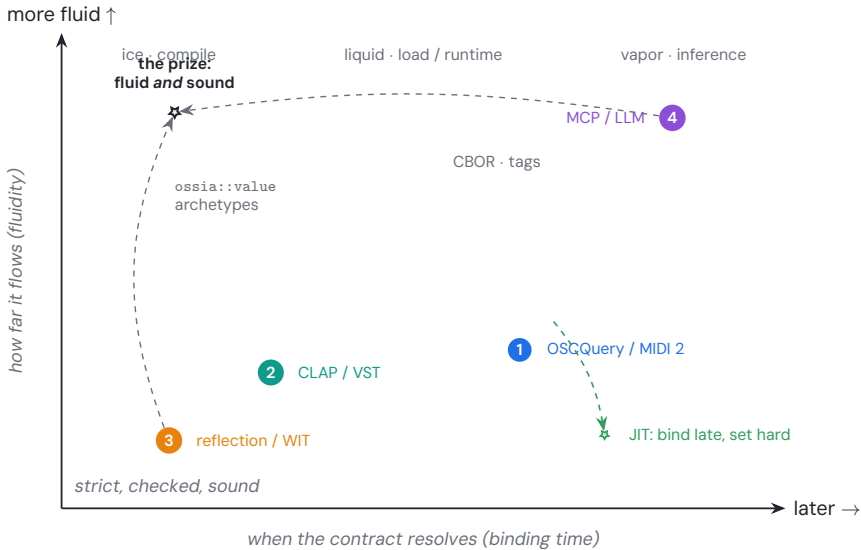
If a tool erases a distinction…

If a new tool can erase a distinction,
the distinction was never real.

static vs dynamic, erased by the JIT.
schema vs code, erased by reflection.
strict vs fluid types, erased by archetypes.
schema vs schemaless, erased by CBOR's tags.

Water has no native shape. The vessels were the only difference, and the dial dissolves them.

Cloud Atlas


more fluid ↑
ice · compile
liquid · load / runtime
vapor · inference
the prize:
fluid and sound
MCP / LLM
4
CBOR · tags
ossia::value
archetypes
how far it flows (fluidity)
1
OSCQuery / MIDI 2
2
CLAP / VST
3
reflection / WIT
JIT: bind late, set hard
strict, checked, sound
later →
when the contract resolves (binding time)

Deep structure

Every contract is a type

- An OSCQuery tree, a WIT world, `std::meta::info`, an MCP schema, a session type: one kind of object, a type description.
- The hard part is always the boundary, what happens to a value crossing between two type systems (Matthews & Findler, “the lump,” POPL 2007).

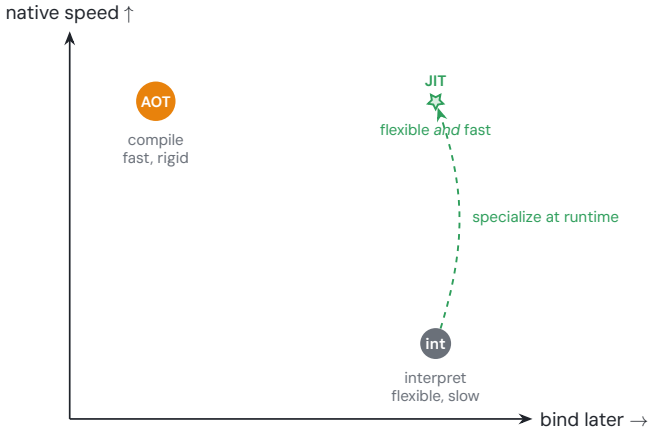
Interoperability is **type-system reconciliation across a boundary**. (Wegner, 1996) : interoperability a first class problem.

A specializer

- A binding generator takes a contract and a peer and writes the adapter. That operation has a name: partial evaluation.
- Futamura, 1971: specialise an interpreter to a program and a compiled program falls out. Reflection stages the marshalling away at compile time; the model does it at inference time.

`specialize(marshaller, contract)` ⇒ `glue, with none left at runtime`

JIT



Run the specializer at runtime. Bind as late as an interpreter, then compile to native speed: the one corner of the dial where you keep both. (JIT toolchain, LAC 2019.)

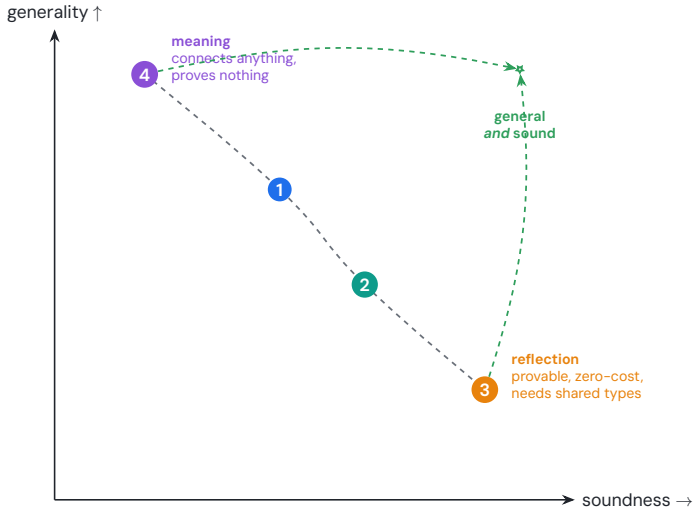
JIT, live: hot-swap while it plays

An Avendish object, recompiled *in-process* (Clang front-end → LLVM ORC `LLJIT`) and swapped on the running audio thread, with no `.so`, no subprocess and no dropout (`code/avendish-jit/…/nary-add`):

```
[selftest] in-process JIT Add<3>: 1+2+3 = 6
>> reconfigured to n=4  (42 ms; audio ran blocks 1303..1577 uninterrupted)
>> reconfigured to n=8  (41 ms; ...)
>> reconfigured to n=1  (42 ms; ...)
Done: audio ran 6030 blocks through three live recompiles.
```

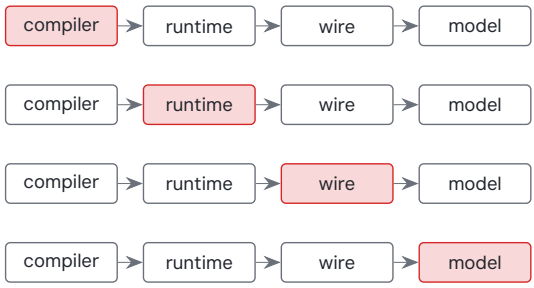
Bind as late as an interpreter, but run at native speed (JIT toolchain, LAC 2019).

Generality against soundness



What's the pareto frontier of all this?

Cost is conserved

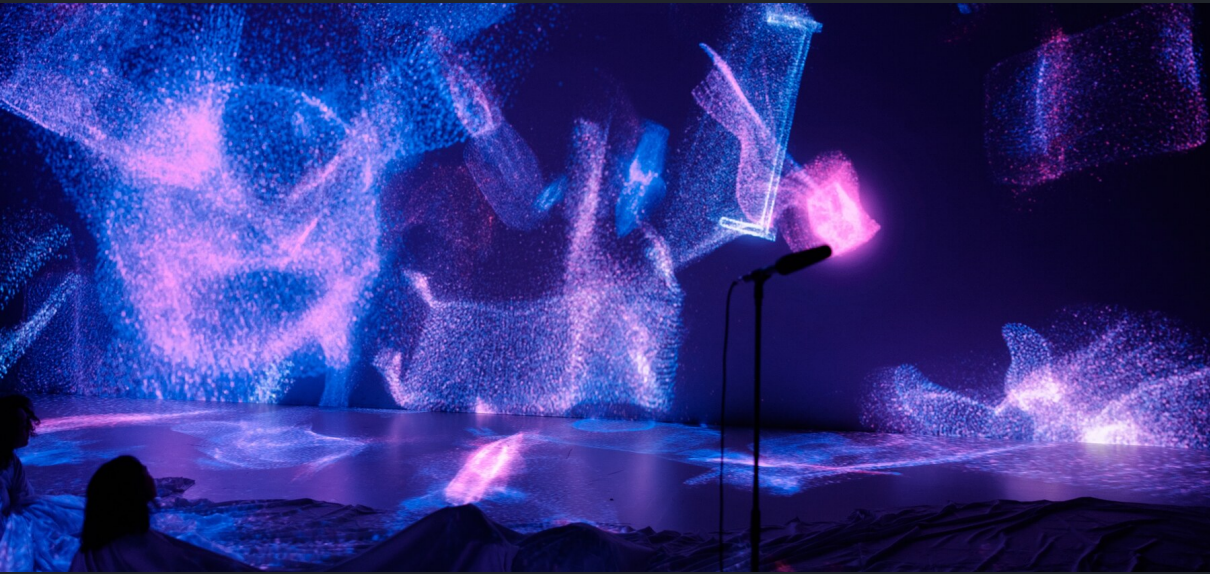


reflection: paid in the compiler, zero at runtime

GraalVM: paid by the runtime, JIT across the boundary

zero-copy (Arrow, Cap'n Proto): paid on the wire format

LLM: paid by the model, in tokens and latency



DreamLLM-3D, Pinyao Liu & Keon Ju Maverick Lee

Where does the model fit?

In the live loop

Artists already do this. *ReVerie* (Liu & Lee, at the SAT) feeds live sensor data straight to a model *as CSV* and asks it to interpret, on stage, every show.

- Maximum generality, zero ceremony
- Risk paid *at every inference*: non-determinism, latency, nothing to audit

At creation time

Use the model to *write the code*, the algorithm and the binding, then ship the code, not the model.

- Output is reviewable, testable, deterministic
- Risk paid *once*, at the keyboard, not on stage

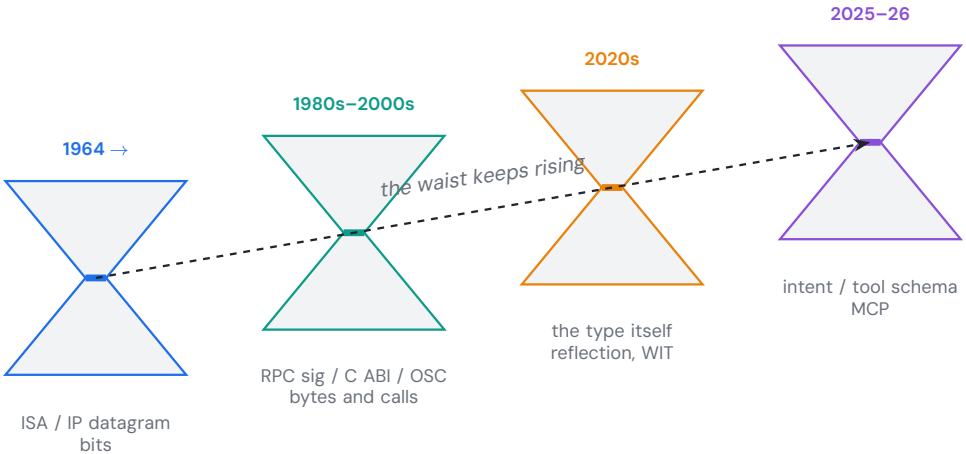
Move the model earlier, from every inference to software-creation time, and you trade a little fluency for an artifact you can audit and keep. Obvious for us! But not how artists are approaching it.

The model wrote the binding

- In the workshop yesterday, I asked an LLM to add a **Csound backend** to Avendish, a whole new target in the $M + N$ picture.
- **One prompt, two clarifying questions, $\sim$25 minutes. $\sim$300k tokens.** Code works.

The fuzziest specializer, used at the safest binding time: let the model write the glue once, then make sure the glue is good rather than the model on stage.

The rising waist



Beck's hourglass: a thin agreed layer, many parts below, many uses above. For sixty years it has climbed, from bits toward meaning.

What is the waist of media arts?

- Every hourglass picks its waist by a trade-off: the thinner and more checkable it is, the more it supports but the less it can say; the richer and more expressive, the fewer things implement it. Performance versus expressiveness, drawn as a layer.
- So what is *ours*? Not bytes (too low to mean anything), not a fixed plug-in ABI (too narrow). The bet of this talk: the waist of media arts is the **typed, reflected object**: expressive enough to carry meaning, thin enough to generate from.
- And it is written *as a set of C++ concepts*. RDF / LV2 keeps the ontology in a separate `.ttl` you hand-sync with the plug-in code; here the ontology lives *in the host language itself*. **C++17** `if constexpr` was the turning point: "do this *if* the type has this property," checked at compile time.

Choosing the waist is choosing where to sit on the performance–expressiveness curve. Ours is a set of concepts: the ontology is defined in the C++ code itself, and enforced at compile-time.

No single best shape

- What is efficient in one environment is wrong in another. VCV Rack hands you *one sample at a time*; a VST hands you a *buffer*. The same algorithm wants a different shape in each.
- There is often no single best representation, *unless* a strong enough compiler can specialise one description into each shape. Which is exactly why we push the contract up to the type.
- The waist cannot be optimal everywhere at once. A powerful compiler is what lets one declarative description stay efficient in each host.

Our waist: the wrappers

- The compile-time waist of Avendish is its *wrappers*: small concept-matchers that recognise a behaviour across backends: the same "matcher" question MetaFFI (Cherny-Shahar, 2024) asks of data types, answered here at compile time, at no run-time cost.
- Two examples that came out: a **generalized value type** (one universal value, many representations) and a **polymorphic audio wrapper** that adapts a single DSP loop to per-sample, per-channel, or per-bus hosts.

The wrapper is where $M + N$ actually happens: write the behaviour once; a concept-matcher binds it to each host's shape. That layer is our hourglass waist, in code.

`ossia::value`

- `ossia::value` is one universal value: a recursive sum of float, string, list, map. Two with different member order are different C++ types, same value.
- Conformance is by concept, consistency, and coercion, not by name. That is an archetype: prototypes, not classes.
- Which is Rosch and Wittgenstein: human categories are family resemblances. The Semantic Web tried to write meaning in logic and lost to a JSON schema. That was correct.

The fuzziest contract in the talk has the deepest precedent, and it has been in our codebase for years.

A law?

An interoperability mechanism is a specializer:

it takes a shared contract-type and a peer, and writes the boundary adapter.

A state space of interoperability? binding time × checkability × contract expressiveness.

Frontier: make the choice of point parametric, and the most general point sound. Is this where LLMs could shine, by defining a custom interoperability mechanism live, depending on the exact situation?

All we do is pass data around

- Strip everything back and the job is one thing: *move data across a boundary*. This whole talk is the ways we do it, and how modern language technology makes every one of them better, without costing performance:

less loss at the bounds · more performance · simpler code

more bugs caught at compile time · better fuzzing · work that still runs in ten years · scientific reproducibility

One paper offered to have your cake and eat it too (Patterson & Ahmed, *Linking Types*, SNAPL 2017). With the contract pushed down to the type, we propose to eat the whole bakery.

My relation to the model

When I came home last year, my parents told me:
"You know, ChatGPT is part of the family now."

Sadly, strong family bonds like that do not convert into OpenAI shares.

More recently, my mother:
"I'm working out with Gemini how it could get a seat at the UN."

A libre worry, and a counterfactual

- Flatpak banning LLM contributions: does that favor proprietary software over free software?
- A question for this room: if Richard Stallman, stuck with that printer in the 1970s, could have just *asked a model to write the driver*, would he ever have written the GPL?

The tool that dissolves the small frustration can also dissolve the impulse that built the commons. Worth guarding on purpose.

At a glance

- **168** cloned libraries (audio, video, image, data, ML)
- **56,430** unique object candidates (distinct *name* per library)
- **3,724** matched to a known algorithm class across **34** categories
- Goal: measure *how much the field re-implements the same algorithms*

Domain	# objects
Audio / DSP	2354
Video / image / CV	915
Control / data	455
Total classified	**3724**

Method: one object = a uniquely-named source/shader/patch unit in a library, bucketed by a keyword taxonomy; a per-library domain prevents audio/video keyword bleed. Headers and cross-platform build copies are de-duplicated.

The most re-implemented algorithms

By raw count (# objects)

Category	#
Filter/EQ	590
Analysis/MIR	299
Oscillator/Source	264
Distortion/Saturation	239
Geometry/Transform	207
Compressor/Dynamics	164
Delay/Echo	143
Math/Map/Logic	131
Sequencer/Clock	121
Keying/Composite	111
Edge/Feature(CV)	108

By spread (# libraries)

Category	libs
Filter/EQ	64
Analysis/MIR	57
Oscillator/Source	55
Distortion/Saturation	50
Math/Map/Logic	45
Compressor/Dynamics	44
Delay/Echo	40
Envelope/Mod	39
Random/Chaos	39
Gain/Util	37
Reverb	36

Filters/EQ are the universal primitive: 590 implementations across 64 libraries.

Spotlight

Algorithm class	# objects	# libraries
Distortion / saturation	239	50
Delay / echo	143	40
Reverb	104	36
Video transition / wipe	50	7
Gaussian blur (named)	11	7
Blur (all kinds)	66	16

- **Distortion** and **delay/echo** are the most duplicated *effects* (∼239 and 143 copies).
- **Video transitions** concentrate in a few libraries (ISF, gl-transitions, MLT, frei0r).
- **Gaussian blur** as a *named* object is rare (11); "blur" in general appears 66 times, since the algorithm hides inside generic convolution/kernel code.

Audio / DSP categories

Category	#obj	#lib	Top libraries
Reverb	104	36	lmms 9, guitarix 8, WaveDigitalFilters 6
Delay/Echo	143	40	guitarix 24, pd-else 21, piloslib 13
Chorus	58	21	airwindows 15, pd-else 7, guitarix 6
Flanger	32	13	guitarix 8, audio-effects 5, pd-else 5
Phaser	39	12	pd-else 7, guitarix 6, BespokeSynth 4
Distortion/Saturation	239	50	guitarix 23, pd-else 19, piloslib 15
Compressor/Dynamics	164	44	pd-else 22, guitarix 13, lmms 10
Tremolo/Vibrato	53	14	guitarix 10, audio-effects 9, pd-else 8
Ring/AM/FM mod	26	11	airwindows 5, guitarix 4, audio-effects 4
Pitch/Time	80	23	pd-else 21, clam 12, vb-objects 6
Filter/EQ	590	64	airwindows 93, pd-else 85, eurorack-blocks 71
Oscillator/Source	264	55	pd-else 69, eurorack-mutable 22, surge 18
Synthesis/Phys.model	106	29	pd-else 32, eurorack-mutable 15, stk 7
Spatial/Pan	64	20	Spatial_Audio_Framework 11, JamomaMax 11, airwindows 6
Analysis/MIR	299	57	pd-else 31, essentia 23, eDSP 20
Gain/Util	93	37	pd-else 23, JamomaMax 12, lmms 7

Video / image / CV categories

Category	#obj	#lib	Top libraries
Gaussian blur	11	7	nap 3, ofxVisualProgramming 2, openfx-misc 2
Blur (other)	55	16	ISF-Files 11, Gem 8, opencv 6
Sharpen	20	10	openfx-misc 4, FFmpeg 3, ISF-Files 3
Color/Grade	95	18	FFmpeg 11, Lab 11, ISF-Files 11
Invert/Negate	16	13	vuo 2, ISF-Files 2, Gem 2
Transition/Wipe	50	7	ISF-Files 15, mlt 14, gl-transitions 9
Fade	9	5	ISF-Files 4, gl-transitions 2, CImg 1
Keying/Composite	111	22	vuo 19, libvips 16, ISF-Files 12
Glitch/Distort(vid)	50	13	ISF-Files 19, vuo 6, Lab 5
Geometry/Transform	207	22	vuo 37, opencv 35, ISF-Files 26
Edge/Feature(CV)	108	19	opencv 39, dlib 12, ISF-Files 8
Noise/Grain/Denoise	82	19	FFmpeg 21, opencv 9, vuo 9
Stylize	54	14	ISF-Files 14, JamomaMax 6, vuo 6
Generator(visual)	47	11	FragM 24, openfx-misc 6, FFmpeg 3

Control / data categories

Category	#obj	#lib	Top libraries
Sequencer/Clock	121	29	voxglitch 30, pd-else 20, BespokeSynth 17
Envelope/Mod	107	39	pd-else 24, piloslib 19, BogaudioModules 5
Random/Chaos	96	39	dlib 14, pd-else 13, ISF-Files 8
Math/Map/Logic	131	45	pd-else 17, vuo 11, Gem 7

What the duplication tells us

- A **small core of ~15 algorithm classes** (filter, oscillator, distortion, delay, reverb, compressor, envelope, geometry, blur, color) accounts for most of the redundancy.
- The **long tail** (vocoder, PSOLA, ambisonics, optical-flow, slit-scan, Euclidean sequencer) is implemented only once or twice, which is the real value of a Rosetta collection.
- **Filters/EQ** span 64 libraries: the strongest candidate for a single canonical Avendish abstraction.
- Audio is 63% of classified objects, so the collection is still *audio-skewed*; video transitions and CV operators are comparatively under-covered.

Full per-object data: `objects_classified.csv` (3724 rows).

YOUR TURN

Let’s build a plug-in

Thank you

jmcelerier@sat.qc.ca

References

The through-line

- *Rage Against the Glue*, ICMC 2022; *Avendish: a modern C++ solution to the multimedia plug-in API problem* (⟨*Programming*⟩)
- *An Inference Zoo for Real-Time Media Arts*
- *libremidi*, SMC 2024
- *ossia score 3*, TENOR 2022
- *A Cross-Platform Toolchain for JIT-Compilation*, LAC 2019
- PhD: *Authoring interactive media*, 2018

Foundations

- Wegner, *Interoperability*, CSUR 1996
- Birrell & Nelson, RPC, 1984
- Beck, *On the Hourglass Model*, CACM 2019
- Honda, Yoshida & Carbone, MPST, POPL 2008
- Matthews & Findler, multi-language semantics, 2007
- Liskov & Wing 1994; Reynolds 1983
- Lenzerini, data integration, PODS 2002
- Futamura, partial evaluation, 1971
- Wright & Freed, OSC, 1997
- P2996 reflection + P3394 annotations, C++26 (2025)