

Aurora / Technical stack

Version 0.1.0 / Release handoff / 14 October 2026

Aurora is a native, free, open-source audio analysis and reactive visual plug-in. It combines sample-derived meters with six GPU scenes, 24 complete appearance presets and detailed display controls. This document describes the code and binaries in this release, the tools used to produce them, and the limits of the measurements and validation.

Release	Included implementation
Formats	macOS Audio Unit effect (.component) and VST3 effect (.vst3).
Architectures	Universal binary: Apple Silicon arm64 and Intel x86_64.
Minimum OS in shipped binaries	macOS 13.3. Both architectures in both plug-ins declare 13.3 in their Mach-O build metadata.
Native framework	JUCE 8.0.6, pinned to commit 51a8a6d7aeae7326956d747737ccf1575e61e209.
Languages	C++17 for plug-in, analysis and native UI; GLSL for GPU shaders. Objective-C++ and platform code are supplied through JUCE.
Default appearance	Pearl Cloud scene, Flight Deck material theme, Phosphor palette, Studio layout.
Audio behavior	Mono or stereo, matched input/output layout; host samples are left unchanged; zero reported algorithmic latency.
Local operation	No account, online activation, runtime AI, audio upload, network requests or recording to disk in Aurora.

What is delivered

The macOS download contains both plug-in bundles and release documentation. The corresponding-source download contains Aurora source, tests, build scripts, technical records, the small editable Pearl Cloud design project and the exact vendored JUCE source with its notices. The development preview and QA applications are tools for synthetic signals; they are not standalone system-audio monitoring products.

Platform scope

This release does not include Windows, Linux, AAX, VST2 or a standalone audio capture application. A universal binary includes Intel machine code, but Intel runtime compatibility has not yet been tested. Actual Ableton playback of this revision and other GPU configurations remain unverified; the completed checks are listed later in this document.

The CMake source contains an intended deployment default of macOS 11.0. The verified release was built with an empty deployment-target cache entry and macOS 13.3 SDK, so its actual minimum is 13.3. This handoff states the binary requirement rather than advertising the unverified intended default.

Native runtime & host integration

The plug-in is compiled software hosted by a DAW.

Layer	Technology and responsibility
DAW interface	JUCE AudioProcessor and AudioProcessorEditor; JUCE Audio Unit and VST3 wrappers. Product Aurora, manufacturer Soft Club, bundle ID audio.softclub.aurora.
Native interface	JUCE Components, Graphics, Paths, Images, controls and Viewports. Native 2D drawing renders the waveform, spectrum, spectrogram, stereo display and levels.
GPU renderer	JUCE OpenGLContext requests desktop OpenGL 3.2. GLSL shaders, vertex/index buffers, floating-point textures and framebuffer passes produce the creative scenes.
State and files	JUCE ValueTree and XML for host state and .aurora appearance files; a local preset directory. No database or server.
Standard library	C++ arrays, vectors, algorithms, atomics, mutexes and numeric functions. The independent analysis engine has no third-party DSP dependency.
System frameworks	Apple frameworks supply windowing, graphics, plug-in hosting and system integration. They are supplied by macOS, not shipped inside Aurora.

JUCE modules and system linkage

The CMake target links juce_audio_utils and juce_opengl plus JUCE recommended configuration flags. JUCE resolves their supporting modules, including audio processors, audio basics, audio formats/devices, GUI, graphics, events, data structures and core utilities.

The verified link command includes AudioUnit, AudioToolbox, CoreAudio, CoreAudioKit, CoreMIDI, Cocoa, Foundation, IOKit, Security, Accelerate, QuartzCore, OpenGL, WebKit and DiscRecording, with weak Metal and MetalKit linkage. Some arrive transitively from JUCE. This list does not imply Aurora captures MIDI, uses a browser UI, burns discs or renders its scenes with Metal.

Build-time feature switches

```
JUCE_WEB_BROWSER=0
JUCE_USE_CURL=0
JUCE_VST3_CAN_REPLACE_VST2=0
FORMATS AU VST3
IS_SYNTH FALSE
NEEDS_MIDI_INPUT FALSE
NEEDS_MIDI_OUTPUT FALSE
```

The UI uses local system typography and native widgets. There is no Electron, React web front end, embedded website, CDN, telemetry service, cloud database, machine-learning model or external shader service at plug-in runtime. A hosted download page is separate from the plug-in and requires no Aurora backend.

Audio & analysis architecture

Real-time audio, worker analysis and graphics have separate responsibilities.

Stage	Data and scheduling
1 / Host callback	processBlock receives floating-point PCM and copies it into a preallocated single-producer/single-consumer FIFO. It reads host buffers without writing them.
2 / Bounded queue	Two fixed channel arrays have capacity 32,768 frames. JUCE AbstractFifo coordinates indices. A full queue drops analysis frames instead of waiting.
3 / Worker	A JUCE thread drains up to 1,024 frames at a time into fixed arrays and runs AudioAnalyzer. While empty, it waits 4 ms before polling again.
4 / Snapshots	The analyzer publishes meter/FFT snapshots and continuous waveform history. Snapshot/reset access uses a mutex outside the audio callback; snapshots own their copied data.
5 / Display	The editor consumes the latest available results. Visible, running native meters request repaint at 60 Hz; the scene has selectable 15/30/60 Hz targets, default 30.
6 / GPU thread	The scene renderer receives a copied feature state, runs its required shader passes and presents to the editor's own OpenGL context.

Audio callback contract

The callback performs no buffer allocation, FFT, rendering, mutex locking or operating-system event notification. It does not alter, normalize or delay the DAW's samples. The processor reports zero latency and accepts only matching mono or stereo buses. A null right input in the standalone analyzer duplicates left visually; mono loudness counts one channel rather than two.

Discontinuities and bounded memory

FIFO overflow is counted and marks a discontinuity. The worker discards its pending backlog and resets analysis windows so missing samples are not joined into false continuous measurements. Reset or format changes also invalidate old history. The audio path remains untouched. Incoming PCM is not retained indefinitely or written to a file; fixed rings and histograms keep working storage bounded with session duration.

Timing limits

Analysis is asynchronous. Offline exports can advance faster than the worker and drop analysis data, while the exported audio remains unchanged. The 80 ms waveform display buffer affects the visual timeline only. Freeze holds the display, while audio and analysis continue. Actual frame delivery depends on the host, system load, GPU and display scale; repaint targets are not guaranteed frame rates.

Meter definitions & accuracy

Measured data and creative visual controls are kept separate.

Meter	Definition
Sample peak	Maximum absolute sample in the latest complete 50 ms window, converted to dBFS. No intersample reconstruction.
Peak hold	Maximum absolute sample since reset. Counts all delivered samples independently of FFT cadence, UI polling or display freeze.
RMS	Unweighted mean-square amplitude over the same 50 ms window. A full-scale sine is about -3.01 dBFS.
Correlation	$\text{Sum}(L \times R) / \sqrt{\text{sum}(L^2) \times \text{sum}(R^2)}$ over 50 ms. In-phase is +1; inverted polarity is -1; silent/empty-channel windows report 0. DC is not removed.
Spectrum	4,096-point periodic Hann window, 1,024-sample hop, 2,049 bins including DC and Nyquist. Stereo powers are averaged before dB conversion, preserving anti-phase energy.
Spectral calibration	Bin frequency is $\text{index} \times \text{sample rate} / 4,096$. Coherent-gain correction makes a bin-centred full-scale sine on both channels read 0 dBFS. Off-bin tones spread over bins.
Momentary / short-term	Two-stage K weighting; summed channel power and -0.691 offset. Windows are 400 ms and 3 s, updated every 100 ms after the first full window.
Integrated LUFS estimate	Overlapping 400 ms blocks; -70 LUFS absolute gate and relative gate 10 LU below the absolute-gated mean. A fixed 0.1 LU histogram stores actual summed energies and counts.

What "estimate" means

Integrated gate decisions use each occupied histogram bucket's mean energy. Blocks near the relative threshold may be included or excluded together, although the final average uses actual energies rather than rounded bucket centres. There is no fixed session-duration expiry. This implementation is not a certified BS.1770 / EBU R 128 meter and has not passed a full standards compliance suite.

The analysis accepts 8-384 kHz sample rates, recalculating filter coefficients per rate. Time windows round to whole samples. Invalid formats are ignored and nonfinite analysis samples become zero; host audio is still not rewritten. Level fields use a finite -120 dB floor, also used before enough loudness samples exist.

Excluded measurements

Aurora does not currently measure true peak, loudness range or calibrated analog VU/PPM. The low/mid/high controls used for animation, visual sensitivity, waveform colours, automatic stereo display gain and scene transients are creative encodings. They do not change calibrated meter values and are not beat or tempo detectors.

Waveform & instrument rendering

The visual detail is derived from stored sample statistics.

Continuous waveform history

The worker accumulates every sanitized sample into buckets of $\text{round}(\text{sample rate} / 1,000)$ samples. Each channel retains signed minima/maxima, the exact first sample positions of those extrema, double-precision squared-sample energy, filtered-band energy and 65 signed amplitude-occupancy counters. The ring retains at least 24.25 seconds. An unfinished bucket is published as a replaceable tail, so stopping between bucket boundaries does not erase final samples.

At the actual display pixel density, the editor groups only contiguous buckets on sample-time boundaries. Energies and histogram counts are summed before deriving RMS and colours. Missing intervals stay blank. This preserves measured extrema and avoids averaging already-normalized colours. The histogram is quantized, does not preserve sample order and cannot reconstruct individual PCM samples.

Display feature	Implementation
Materials	Silk emphasizes measured amplitude occupancy; Prism emphasizes filtered-band RMS contours; Trace emphasizes fine RMS/band lines. All share the same signed envelope.
Colour	Bands uses configurable low/mid/high colours. Its 12 dB/oct Butterworth splits are 250 Hz and $\min(4 \text{ kHz}, 0.45 \times \text{sample rate})$. Palette maps RMS from -60 to 0 dBFS through three RGB stops.
Brightness	Density receives local three-bin smoothing, per-column maximum normalization and square-root contrast. This is artistic shading of measured occupancy, not calibrated probability.
Time / amplitude	1, 3, 6, 12 or 24 s; custom 0.25-24 s. Shared 1x/2x/4x magnification and display gain produce explicit fixed axes. Raw full-scale flags still require actual samples reaching ± 1 .
Cache / scrolling	A circular pixel cache reuses stable columns. A fractional sample clock uses an 80 ms display buffer and bounded correction, never scrolls beyond received audio, holds on stop and reanchors on resume.
Scope	Separate triggered close-up: up to 768 original samples from the latest 1,024-sample snapshot. Shared, labelled adaptive amplitude range; history materials and magnification do not apply.

Other meter surfaces

The spectrogram stores a 480×256 cache of UI-sampled FFT history, up to 24 seconds, and recolours retained dB data through a lookup table. It is approximate wall-time history, not a lossless spectrogram recording. The stereo display accumulates mid/side sample density into a 256-square cache, with configurable persistence and display gain. Spectrum traces and trails project measured data onto a logarithmic axis and clip at Nyquist.

Creative scenes & GPU pipeline

Procedural shaders react to copied audio features.

The six scenes are Folded Light, Liquid Aperture, Pearl Cloud, Night Transit, Diffraction Veil and Orbit Loom. Scene selection, camera orbit, zoom, three-stop palette, theme, exposure, bloom, saturation, speed and sensitivity are editable. VisualReactivity separates artistic response from numeric metering with adaptive normalization, 18 ms attack, 160 ms release and a 180 ms transient envelope; missing snapshots release after 250 ms.

Pearl Cloud: persistent fluid and layered material

PearlFlow uses floating-point ping-pong textures for velocity, pressure and pigment. Its bounded fixed steps perform velocity advection, curl/confinement and audio forcing, divergence, 12 pressure iterations, pressure projection and pigment transport. Limited MacCormack correction preserves fine dye structure without introducing new extrema. The production pigment canvas has a fixed 848:512 aspect; the solver's long side is 256 cells and the dye long side is 768 texels.

PearlFlowMaterial integrates three curved optical strata over 128 samples per view ray. Density, relief, transmission and highlights create the pearlescent volume; a support mask contains it. Colour channels stay palette-independent so colour edits can update the appearance immediately. Silence and Freeze stop simulation advance. A zero flow-speed control also holds motion. The fallback shader is the earlier deterministic design study.

Shared rendering and performance choices

Only Pearl Cloud initializes and advances the fluid solver. The other five scenes use their own procedural geometry/material shaders. All scenes receive a floating-point finishing pass with quarter-resolution bloom; finishing is not fed back into the simulation. Camera framing considers rotated bounds, while intentional zoom can crop the image.

Uniform locations are cached after linking. Unchanged frozen scenes avoid redraws. Covered or absent scene panels skip updates, and Customise parks the existing OpenGL surface outside the visible area to preserve its context and resources. Closing the editor releases its GPU context. Palette, spectrum and stereo caching reduce repeated CPU work without reducing measured waveform resolution or Pearl Cloud's material sampling.

Optional Diffusion Studio authoring

Diffusion Studio v0.211.0 was used to edit and render the earlier Pearl Cloud shader study. Its project is TypeScript/TSX plus GLSL, with solid-js and @diffusionstudio/jsx supplied by the editor. There is no local npm dependency installation for that project. Studio is optional and is not linked, bundled or required by Aurora. Despite its name, no diffusion model or runtime AI is part of the plug-in.

design/pearl-cloud/pearl-cloud.glsl generates shader.ts for Studio and Source/PearlCloudShader.h for the native fallback. scripts/sync-pearl-cloud.py checks those embeds. The current persistent solver and material live in PearlFlow.h and PearlFlowMaterial.h; the earlier Studio renders do not demonstrate that newer engine's performance.

Appearance, presets & persistence

Presentation settings never rewrite audio measurements.

Aurora supplies 24 complete factory looks, 12 named palette choices plus Custom, four material themes and four layouts: Studio, Meter strip, Waveform and Custom. The appearance model adds 96 validated custom settings to the core selectors, colours, camera and intensity. A complete factory look sets all 112 appearance XML attributes explicitly.

Area	Editable controls
Colours / interface	Arbitrary RGB sliders and six-digit hex values; three gradient stops and midpoint; interface background, text, grid, panel and clip-marker colours; independent module palettes.
Waveform	Material, colour mode and band colours; channels; custom time window; gain, density, RMS body, crest, band detail, fill, gradient, grid and labels.
Spectrum / spectrogram	Frequency and dBFS limits, line/fill, trails, contrast, time span, direction and grid.
Stereo / levels	Stereo persistence, brightness, rotation and gain; levels floor, warning threshold, RMS shading, peak holds, segments and loudness visibility.
Scene / layout	Scene, camera, motion, response, finish and frame-rate target; gaps, proportions, one/two/three columns, module visibility and order.
Presets	Factory collection, previous/next navigation, modified-state indicator, local saved library, import/export, section reset and whole-appearance reset.

Storage contracts

Host state is a JUCE ValueTree serialized to XML and copied to the host's binary state block. Its root is `AuroraMetersState`, version 1; it includes appearance and Freeze. Imported appearance presets use a separate appearance-only contract. Finite values and limits are validated, RGB values are masked to 24 bits and module positions remain a permutation. Missing older properties receive defined defaults.

The local macOS library is `~/Library/Aurora/Presets/`, one `.aurora` XML file per saved appearance. Presets contain no recorded audio, FFT history, loudness integration or measurement history. Loading/importing a preset or resetting appearance preserves current measurements and Freeze. The main Reset button separately clears measurement state.

UI behavior

Customise is a native dropdown within the editor, with scrollable content in short layouts. Toggle or Escape closes it. Header layout selection closes it and reveals the chosen meters. Controls synchronize only when their model changes, preventing timer refreshes from erasing an asynchronously delivered menu selection. Camera drag, wheel/pinch zoom and reset remain available for the 3D scene.

Build tools & reproducible source

Tools used to author and package Aurora are separate from runtime dependencies.

Tool	Role / verified version
Apple toolchain	Apple Clang 14.0.3.14030022, Command Line Tools compiler and macOS 13.3 SDK produced the verified release. C++17 is required.
CMake	3.31.6 installed locally from PyPI; project minimum is 3.22. Generates plug-in and test targets through JUCE CMake helpers.
Ninja	Pinned Python distribution ninja==1.11.1.4. Executes generated builds.
Python 3	Creates the local tool environment, synchronizes shader embeds and writes ZIP archives with zipfile. Not required to run the plug-in.
Shell / Git	Bash build, signing and install scripts; Git fetches JUCE 8.0.6 when the vendored tree is absent.
Apple release utilities	codesign signs/verifies bundles; lipo checks architectures; otool checks linked libraries and deployment metadata; auval validates Audio Units; ditto copies bundles.
QA / capture tools	CTest and original native test runners. Optional Swift video encoder uses AVFoundation, CoreGraphics, CoreVideo, ImageIO and Accelerate to encode captured PNGs.
Authoring assistance	OpenAI Codex assisted implementation, research, tests and packaging in this workspace. It is a development tool, not an Aurora dependency or service.
Optional visual editor	Diffusion Studio 0.211.0 and its CLI; only needed to open the included editable design study.

Standard macOS build

Install Apple Command Line Tools and Python 3, extract the source archive, then run from aurora-meters:

```
xcode-select --install
./scripts/build.sh
```

build.sh obtains JUCE 8.0.6 if needed, creates .tools, installs pinned CMake/Ninja, configures Release for arm64;x86_64, builds with four jobs, runs CTest and ad hoc signs macOS bundles. With the included JUCE tree it does not need a fresh JUCE clone; the first tooling install still requires internet. Nothing is automatically installed into a DAW.

For a release with an explicit deployment target, pass -DCMAKE_OSX_DEPLOYMENT_TARGET=13.3 when configuring. Do not infer a binary's minimum OS from the default in CMakeLists.txt; inspect its Mach-O metadata. Changing the target is a new build that requires validation.

```
.tools/bin/cmake -S . -B build -G Ninja \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_OSX_ARCHITECTURES="arm64;x86_64" \
  -DCMAKE_OSX_DEPLOYMENT_TARGET=13.3
.tools/bin/cmake --build build --parallel 4
.tools/bin/ctest --test-dir build --output-on-failure
```

Validation & measured performance

Evidence is scoped to the fixtures and hardware actually tested.

The 14 October 2026 revision built universal AU, VST3 and QA applications. Apple's Audio Unit validator passed. Signed installed executables matched the tested build hashes. The full machine-readable record is docs/inline-controls-validation.json; the readable sequence of revisions is docs/VALIDATION.md.

Check	Completed result
Headless CTest	8 suites passed in 10.89 s: analysis, visual_framing, visual_reactivity, waveform_clock, waveform_projection, waveform_ribbon, processor and editor_performance.
Editor cache/raster	91 checks, including pixel equivalence against retained reference routines, relevant invalidation, unrelated-setting cache reuse and incremental spectrogram consistency.
Native customization	520 checks, 53 captures, zero failures. Exercises asynchronous menu/timer races, header/embedded layouts, compact heights, scrolling, toggle, Escape and same-window behavior.
Factory collection	1,044 checks, 42 captures and 17 actual GPU captures. All 24 looks; settings/history preservation and repeated same-context dropdown lifecycle.
Audio Unit	auval -v auxf Auro Aucl passed. Universal architecture and strict code-signature verification were also checked.

Isolated CPU comparison

Median stereo raster work fell from 14.561 ms to 0.462 ms. Full spectrogram recolouring fell from 58.895 ms to 10.740 ms. These compare retained test-only baseline routines with the production renderer on identical synthetic data. Output pixels match exactly. They are timings of those routines, not whole-editor, DAW or audio-thread timings.

Isolated GPU comparison

On Apple M2 Max at 1,440 x 900 physical pixels, 30 synchronized frames per scene showed 64-75% lower median rendering cost for the five non-Pearl scenes: 2.357-2.900 ms before and 0.712-0.881 ms after. The baseline deliberately reinserts the previous unused fluid workload into the current renderer; it is not an old-binary comparison. GL checks passed, inactive fluid pixels stayed unchanged and Pearl Cloud still evolved. No Pearl Cloud speedup is claimed.

What these tests do not establish

Fixtures use synthetic PCM and open no audio device. Results do not establish Ableton frame rate, real-time audio guarantees, Intel runtime compatibility, every host/version/GPU combination, notarization or full ITU/EBU certification. Prior waveform tests also cover exact extrema, energy grouping, partial tails, clipping, gaps, fixed axes, freeze/resume and cache correctness. The native GUI and GPU runners require a graphical desktop and are separate from headless CTest. This handoff document is generated with Python 3 and ReportLab; pypdf checks its extracted text and Poppler renders the pages for visual inspection. These document tools are optional and not plug-in dependencies.

Distribution, signing & installation

The release can be served as ordinary downloadable files.

Release artifacts

The public binary archive is Aurora-0.1.0-macOS-universal.zip and the corresponding-source archive is Aurora-0.1.0-source.zip. Keep the matching source and dependency notices alongside the binary when publishing the release. A website can link these files directly; GitHub Releases can host them as assets. Aurora needs no server code, package manager, runtime database or account service. This handoff does not create or publish a website.

Signature status

The bundles are locally ad hoc signed with `codesign --sign -` and pass strict signature verification. They are not Developer ID signed or notarized. Local validation does not establish acceptance of a browser-downloaded, quarantined copy on another Mac. This is a material release property, not a notarized installer.

DAW installation locations

```
~/Library/Audio/Plug-Ins/VST3/Aurora.vst3  
~/Library/Audio/Plug-Ins/Components/Aurora.component
```

Copy the VST3 and/or Audio Unit bundle into the corresponding user Library folder, then enable that format's system plug-in location and rescan in the host. Look for Aurora under Soft Club and place it on a track or master bus. Actual host labels and menus depend on the host version. The source repository also provides `scripts/install-macos.sh`, which installs a locally built release and backs up an existing Aurora bundle.

Source packaging and shader consistency

```
python3 scripts/sync-pearl-cloud.py --native --check  
python3 scripts/package.py
```

`scripts/package-release.py` wraps the validated base archives, documentation and screenshots into Aurora-0.1.0-website-kit.zip. This standard-library Python packager creates the public 0.1.0 downloads, preserves executable modes and writes checksums, with the matching source and dependency notices. It does not publish to GitHub or a website.

The original `package.py` generates the base Aurora-0.1-macOS-universal.zip and Aurora-0.1-source.zip inputs. It first checks that the editable GLSL and generated Studio/native embeds match. It includes Aurora source, Tests, scripts, docs, CMakeLists.txt, README, LICENSE, THIRD_PARTY.md and the JUCE tree. The small Studio project is explicitly enumerated. Git metadata, build outputs, tool environments, `node_modules`, caches, backups and working captures are excluded from the source ZIP. The binary ZIP carries the AU/VST3 bundles and project notices.

Releasing a later revision

Build and test the new source, verify architecture and actual minimum OS, sign with the intended release identity, validate AU and representative hosts, and regenerate matching binary/source archives. Use fresh checksums for the final files. Developer ID signing and notarization would be additional release work; neither is implied by the current ad hoc signature.

Licenses & provenance

This inventory records project notices; the included license texts remain authoritative.

Component / source	License and role
Aurora original code	Copyright 2026 Aurora contributors; GNU Affero General Public License v3.0 or later. Full text in LICENSE; project notice in THIRD_PARTY.md.
JUCE 8.0.6	Pinned commit 51a8a6d7aeae7326956d747737ccf1575e61e209. JUCE modules offer AGPLv3 or commercial licensing; Aurora uses the AGPLv3 route. See third_party/JUCE/LICENSE.md.
VST3 SDK in pinned JUCE	The vendored SDK's notice lists the proprietary Steinberg / GPLv3 options. Aurora's dependency record identifies the open-source GPLv3 option. Retain the exact bundled notices.
Apple AudioUnitSDK	Apache 2.0 notice in JUCE's Audio Unit client source. System frameworks remain macOS dependencies.
Pearl Flow solver reference	Finite-difference GLSL informed by/adapted from Pavel Dobryakov's WebGL Fluid Simulation; MIT, copyright 2017 Pavel Dobryakov. Full MIT notice retained in Source/PearlFlow.h.
Original fluid additions	Aurora's audio forcing, ribbon sources, lifecycle, optical material and post-processing. No upstream assets or preset pack are bundled.
Bundled JUCE dependencies	Examples include FLAC, Ogg/Vorbis, zlib, PNG, JPEG, HarfBuzz, SheenBidi and OpenGL helper code. Exact licenses/notices are retained in the vendored JUCE tree. Presence in that tree is not a claim that every module is active.
Build tools	CMake, Ninja, Python, Git, Apple tools and optional Diffusion Studio are authoring/build tools; their applications and environments are not linked into Aurora or shipped as plug-in runtime dependencies.

Research references

Ghost and MiniMeters were studied as functional and visual references. No vendor native source, binaries, graphics, presets or branding were copied into Aurora. Public open-source visualizer research also examined projectM, Butterchurn, Pulse, Prism and OpenMeters. Their renderers or preset collections are not incorporated. The solver adaptation listed above is the explicitly retained third-party implementation reference.

Research links, inspected revisions and distinctions between observed demos and source inspection are recorded in docs/REFERENCE-RESEARCH.md and docs/VISUAL-RESEARCH.md. The earlier Soft Club and later Flight Deck directions are original treatments; the release does not bundle Cowboy Bebop imagery, logos or media.

Source and notices stay together

The source archive includes the exact JUCE source used for the release, its dependency notices and Aurora's build scripts. Read LICENSE, THIRD_PARTY.md and the relevant vendored files for their actual terms. This document is an implementation inventory and does not replace those texts.

Source map & further reading

Paths below are relative to the extracted aurora-meters source folder.

Path	Responsibility
CMakeLists.txt	Plug-in formats, C++17, JUCE module linkage, preview and test targets.
Source/PluginProcessor.{h,cpp}	Audio passthrough, FIFO, analysis worker, host state, preset import/export and defaults.
Source/Analysis.{h,cpp}	Independent DSP: FFT, peaks, RMS, correlation, K weighting, gated loudness, waveform statistics and history.
Source/PluginEditor.{h,cpp}	Native meter drawing, display caches, controls, dropdown, layouts and editor timing.
Source/Customization.h; CustomizationPanel.cpp	96 validated appearance settings; native section controls, colour entry, preset library and layout controls.
Source/FactoryPresets.cpp; FactoryPresetBrowser.cpp	24 fully specified factory appearances and illustrated native gallery.
Source/Palette.h; SpectralProjection.h	Palette definitions and frequency projection used consistently for drawing and readouts.
Source/WaveformClock.h; WaveformProjection.h; WaveformRibbon.h	Fractional sample timeline, precise grouping and measured-material pixel caching.
Source/VisualSurface.{h,cpp}; VisualFraming.h; VisualReactivity.h	GPU bridge, scenes/camera, fitted bounds and independent artistic music response.
Source/PearlFlow.h; PearlFlowMaterial.h	Persistent solver, retained MIT notice and original layered ray-integrated material.
Source/SceneMaterial.h; VisualPostProcess.h; PearlCloudShader.h	Other scene materials, floating-point finishing/bloom and generated fallback shader.
design/pearl-cloud/	Editable optional Studio TSX/GLSL study, generated import and its capture record.
Tests/	Independent DSP and rendering tests, native UI/GPU runners, procedural music fixtures and silent preview.
scripts/	Build, sign, install, package, shader synchronization and optional Swift preview-video encoder.
docs/METERING.md	Detailed meter definitions, timing, display encodings and known accuracy limits.
docs/VALIDATION.md; inline-controls-validation.json	Human-readable verification history and exact current machine-readable results/hashes.
LICENSE; THIRD_PARTY.md; third_party/JUCE/	Aurora license, dependency summary and pinned framework source/notices.