LED Matrix Server — Help

A modern LED matrix server. Cross-platform Electron app, all the pixel maths in JavaScript (some effects WebGL-accelerated), six output protocols covering everything from venue-grade DMX to DIY Arduino strips.

Getting started

> LMS launches on the **Scene/Chase Builder** — the main work surface. From there:
>
> - **Settings** → `File` menu → `Settings` (or `Ctrl+,`).
> - **Show Mode** → click the **Show Mode** button in the Builder's centre column.
> - **Back to Builder** from any sub-page → `Esc`, the ◀ **Back to Builder** button, or the window **X**.
>
> Only the window X on the Builder itself actually quits the app — every other X returns you home.

Settings (File → Settings · Ctrl+,)

Everything that's set once and forgotten: matrix layout, render rate, audio source, and where pixels go. ◀ **Back to Builder** button top-left to return.

Tabbed layout

The Settings page is split into five tabs across the top:

- **Matrix** — dimensions, cabling pattern, pixel order, bit depth, start corner, the Open Patch Editor button, and the live wiring preview.
- **Protocol Settings** (was "Output" pre-46) — Output Streaming master gate + per-protocol cards. Each card carries only its *protocol-wide* options — Universe Economy, ArtSync, sACN mode/priority/Source name, DDP Economy, and the **Send-to …** toggle. **Per-row Target IP and Universe live in the Patch Editor** since the Patch is the canonical place to wire up destinations. TPM2.net, Serial DMX, Gledіator, and HDMI cards stay full-featured (whole-frame protocols not in the Patch's protocol dropdown, or non-protocol outputs). **Tab reordered to Matrix → Protocol Settings → Patch Editor → …** so the workflow reads top-to-bottom (set canvas, choose protocols, then patch rows).
- **Audio** — input source, gain, auto-start.
- **External Control** — Art-Net DMX IN, sACN IN, OSC, MIDI Clock + MTC.
- **General** — render rate, Performance Mode PIN, Project file auto-save, Tablet Remote port.

The last-active tab is remembered across launches via localStorage so you land back where you were.

Matrix

- **Width × Height** — your physical matrix dimensions. Up to 1024 × 1024.
- **Cabling pattern** — how the data cable snakes through your pixels: Rows L→R / Rows zig-zag / Cols T→B / Cols zig-zag. Pick the one matching your wiring. (Per-row Panel Footprints override this — see Output Patch below.)
- **Start corner** — where pixel 0 of the wire-order stream lives: Top-Left (default) / Top-Right / Bottom-Left / Bottom-Right. Cabling pattern still picks the snake style; this picks the origin. Lets matrices wired from any corner be patched without remapping.
- **Pixel order** — colour-channel order your LEDs expect. Grouped into:
  - **RGB (3 bytes/pixel)** — RGB / GRB (default for most WS2812) / BGR / RBG
  - **RGBW (4 bytes/pixel)** — RGBW / GRBW / WRGB / WGRB for white-channel strips (SK6812-RGBW, WS2814W). LMS computes W = min(R,G,B) and subtracts it from R/G/B for accurate colour, standard FastLED/WLED convention.
- **Pixel-order viz chips** — coloured chips under the dropdown show which channel is in which byte position (1 / 2 / 3 / 4). Live-updates on dropdown change.
- **Bit depth** — 8-bit (standard, 256 levels/channel) or **16-bit (smooth dimming, 65,536 levels/channel, 2× DMX channels)**. 16-bit emits MSB+LSB on adjacent channels through a γ=2.2 perceptual curve — gives way smoother low-end fades on dimmer packs / DMX drivers that support it. Doubles your channel/universe count.

- **Matrix summary line** at the bottom of the Matrix card now reads "… *DMX channels (min N universes) · patched M univ.*" The **min** count is the absolute floor assuming max-packing at 170 px/universe. The **patched** count is the actual unique-universe total from your Patch Editor configuration (Art-Net + sACN rows; DDP doesn't use universes). Real-world setups almost always have patched > min because panels split / variable-length universes / per-controller boundary breaks add overhead — the two numbers together make that gap obvious.
- **Apply** — commits and re-renders the preview. Pink dot = start of cable. Violet line = the path.
- **Open Patch Editor** → — opens the per-universe Output Patch table (see dedicated section below).

Render rate

Slider snaps to clean monitor-refresh divisors: **15 / 20 / 24 / 30 / 40 / 48 / 60 FPS**. Intermediate rates would judder because the browser's `requestAnimationFrame` tick can only pace whole multiples of the monitor refresh (16.67 ms on 60 Hz, 6.94 ms on 144 Hz) — picking 25 fps on a 60 Hz monitor effectively gets you 20, because the throttle has to wait until the elapsed time exceeds the target interval. The snap stops are the ones that pace evenly. **Live FPS** shows the measured rate so you can see whether your effect stack is keeping up.

Changing FPS mid-stream now resets all protocol-send throttle accumulators + Universe Economy bookkeeping so the next frame paces cleanly at the new rate — previous behaviour glitched the output for a few frames after a slider tweak.

Audio Input

Drives the audio-reactive effects (Spectrum Bars, Beat Pulse, Beat Rings, VU Meter, plus any effect with an **Audio Trigger** / **Audio reactive** checkbox).

- Pick a Source from the dropdown, hit **Start**. Live level meter shows it's working.
- To react to **music playing on your PC**, route the system audio through a loopback device:
  - **Windows:** install VB-Audio Cable (https://vb-audio.com/Cable/) — set the media app's output to "CABLE Input"; in LMS pick "CABLE Output" as the source.
  - **Mac:** install BlackHole (https://existential.audio/blackhole/) — same pattern.
- Microphone mode (no loopback) works for live-mic / DJ-on-stage setups.

Output Streaming

Dedicated card with the **master gate** for every network output protocol. Big green/red button — click to silence (or restore) every Art-Net / sACN / DDP / TPM2 / Serial DMX / Gledіator send at once. **HDMI is NOT affected by this gate**, so for HDMI-driven walls you can leave streaming off entirely and save the per-frame network-send CPU cost.

- **OUTPUT STREAMING — ON / OFF** — master toggle. Mirrors a 💡 bulb button on the Builder centre column AND a small toggle on the Patch Editor toolbar so you don't have to bounce back to Settings mid-show.
- **Remember streaming state across launches** — when ticked, the app boots into whatever streaming state was active when you last closed it. When unticked (default), it always starts with streaming ON.

- **Blackout on stop** — turning streaming OFF now fires **three** zero-frame sweeps spaced 50 ms apart (150 ms total) across every active output (patched rows + whole-frame protocols). Three sweeps because UDP is lossy — a single zero-frame can be dropped on a congested switch and leave pixels stuck on, which is the worst possible failure mode for "Stop". Three sweeps nearly guarantees delivery to every receiver. Well under perceptual instant.
- **Lock during streaming:** while streaming is ON, the Matrix shape + every output protocol's config params freeze (yellow 🔒 hint in the affected card headers). Per-protocol "Send to …" toggles stay live so you can mute individual protocols mid-show; External Control IN is unaffected. Turn streaming OFF to edit.
- **Send-to toggles now gate patched rows too (beta.46):** previously the per-protocol Send-to toggles only affected whole-frame mode and were ignored in the Patch path. Now they act as a true per-protocol mute everywhere — flip "Send to Art-Net" off, every patched Art-Net row stops sending. If a patched protocol's Send-to is OFF, an **amber warning banner** appears at the top of the Patch Editor: "*Protocol muted by Send-to toggle — Art-Net (3 rows). Toggle Send to… back on to resume sending.*" So nothing silently goes dark.

Settings persistence

Every Settings field is remembered across launches. That covers:

- Matrix dimensions, cabling, start corner, pixel order, bit depth.
- Art-Net, sACN, DDP, TPM2 — host / IP / universe / priority / source name / mode / economy AND each protocol's **Send to …** toggle. Send-to toggles auto-Apply on restore so the configure() call fires before the first frame.
- HDMI — display, canvas mode, scale, X / Y, bg colour, always-on-top, click-through, custom W / H. Saved on EVERY field edit (not just Apply / Open), so HDMI-only rigs don't lose config when saving before clicking Open.
- NIC selectors (Output NIC, Input NIC).
- Audio input mode, device, gain, auto-start.
- Render rate target FPS, Performance Mode PIN, Lock on next launch, Auto-save project on close, Tablet Remote port + Auto-start.
- Last-active Settings tab.
- All Output Patch rows (including Panel Footprints) — these save with the project file.

Tablet Remote

Small card next to Output Streaming. Sets the listen **port** (default 8080) and an **Auto-start when entering Show Mode** checkbox (ticked by default for fresh projects). Live state, QR scan, and Stop-Server controls live on the Show Mode side panel — see the dedicated Tablet Remote section below.

Protocol Settings

Protocol-wide config — defaults that apply across all rows of each protocol (ArtSync, sACN Source Name + multi/unicast + priority, Economy flags, Gledіator port, etc.). Tick the "Send to …" box on each card to enable that protocol. **Multiple outputs can run simultaneously** (e.g. Art-Net to a network node + HDMI to a Colorlight wall + Gledіator to a DIY Arduino strip). Per-row Target IP / universe / pixel mapping lives in the **Patch Editor**.

Protocol status pills on each card now stack as two lines — top line "CONFIGURED" / "FAILED" / "CONNECTED", bottom line wraps the target detail (host, universe, priority, CID, etc.) so it stays inside the column.

- **Art-Net** — UDP to a node (broadcast or unicast). Pick a Target IP, Port (default 6454), Start Universe. Auto-paginates across universes for matrices > 510 channels.
- **sACN (E1.31)** — multicast DMX over IP, venue standard. Multicast goes to `239.255.<u-hi>.<u-lo>` per universe automatically. Set Source name (shown in receivers' source lists) and Priority (0–200, default 100).
- **TPM2.net** — UDP frame protocol used by many DIY LED controllers (Gledіator-style). One packet per frame.
- **Serial DMX** — ENTTEC USB Pro / DMXKing UltraDMX / Open DMX dongles. Web Serial — **no native module, no driver beyond the standard USB-serial one**. Click **Connect dongle…**, pick the port → 512 channels stream at your render FPS. **Connection feedback — 1.1.8:** the Connected box now names the actual serial port ("FTDI · COM4"), and pulling the dongle out is noticed straight away rather than the card carrying on as if nothing happened — it reports "device unplugged", unticks its own **Send to…** box, and raises an on-screen warning wherever you are in the app. **Output Streaming is deliberately left alone**, because a nudged USB cable should not black out an Art-Net wall running alongside it. Plug it back in and press Connect and it re-arms itself. Same for the Gledіator card. ⚠ **Set *Device type* to match your dongle** (1.1.8): Pro-style dongles have a microcontroller that makes the DMX timing from a framed packet; Open DMX / raw FTDI-RS485 adapters have none, so LMS generates the break itself. The wrong choice gives a BAD signal with no channels rather than an error — see Troubleshooting. You can also route individual patch rows to the dongle — set a row's Protocol to **DMX (USB)** in the Patch Editor (see Output Patch below).
  Dongle not showing up in Connect? Web Serial can only list devices that have a Windows **COM port**. If another DMX app (ENTTEC Pro Manager, FreeStyler, QLC+…) previously opened the dongle in **D2XX / direct mode**, it strips the COM port and the dongle becomes invisible here. Re-enable it: **Device Manager** → **Universal Serial Bus controllers** → **USB Serial Converter** → **Properties** → **Advanced tab** → **tick "Load VCP"** → **OK**, then unplug and replug. The dongle reappears under *Ports (COM & LPT)* and Connect will find it. Also connect it **before** launching LMS (or click 🔄 Rescan) — serial ports are enumerated at startup.
- **Gledіator** — Solderlab's classic serial protocol, speaks to most DIY Arduino / Teensy / ESP LED controllers (WS2812 / SK6812 / APA102). `0x01` sentinel + raw RGB(W) pixel bytes per frame. Selectable baud (1M / 500k / 250k / 115k / 57k — most boards want 1M). ⚠ **The frame carries no size or checksum**, so the matrix dimensions are compiled into the receiving sketch — set **Matrix** to exactly what the board expects or the output comes out scrambled, with nothing LMS can do to detect it. See Troubleshooting.

- **HDMI / Screen Output** — opens a borderless window on a chosen display, rendering the matrix at a configurable pixel scale. For Colorlight / Novastar / Linsn senders that read pixels off a video output. Always-on-top + click-through optional. **Canvas Mode** (see below) for scaler boxes that expect a fixed input resolution.
  No edge line — 1.1.5. Windows 11 rounds the corners of every window and softens the outer edge, which on a monitor you never notice but on an LED wall lights up the outermost row of pixels as a thin line along the top and left. LMS now sizes the output window slightly larger than the display and offsets it, so that soft edge falls outside the screen entirely while your pixel (0,0) still lands on the wall's first pixel. **Side effect worth knowing:** if the wall's output sits next to another monitor you may see a thin black strip at that monitor's edge while the output window is open. It goes when you close it.
- **NDI Output** *(Windows)* — publish the whole matrix as a live **NDI®** source on the network, so **OBS**, **vMix**, another media server or a recorder can pull the show in over the LAN. A **whole-frame mirror like HDMI**: it follows the composited matrix whether or not Output Streaming is on, and it ignores the Patch. Set the **Source name** other NDI apps will see it as (default *Smartshow LMS*) and tick **Send to NDI Output**; the pill flips to *Sending* and it appears in NDI receivers as "<this-PC> (your source name)". Goes black on Stop/Quit so a downstream recorder doesn't hold the last look. **Requires the free NDI Tools / Runtime (https://ndi.video/tools/).**

HDMI Canvas Mode

Many LED-wall scaler boxes (Colorlight X1, Novastar, Linsn etc.) expect a **fixed HDMI input resolution** like 1920×1080 and map a specific region of that frame to the wall — they do their own scaling internally.

If your matrix is e.g. 384×192 but your scaler expects 1080p in, the old "Pixel scale" mode would stretch the matrix to fill the window and the scaler would shrink it back — two scale operations, blurred or distorted result.

Canvas Mode fixes this:

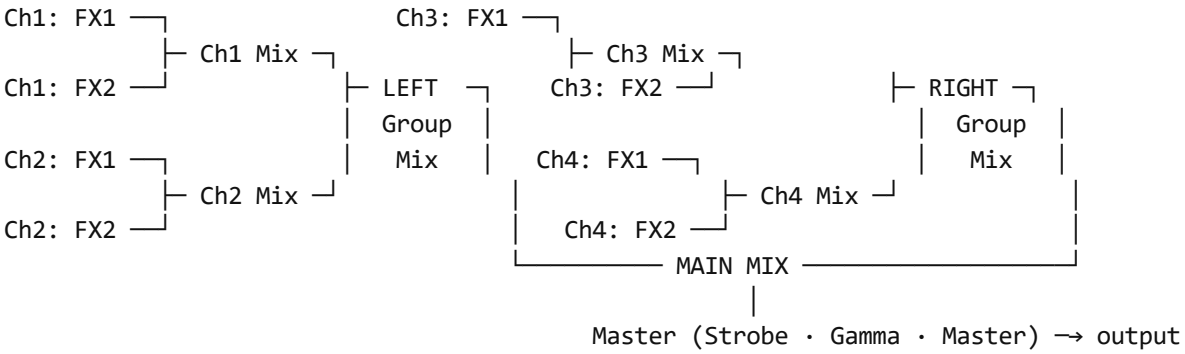
1. Pick **Canvas mode** → **1080p (1920 × 1080)** (or 720p / 4K / custom) in the HDMI card.
2. LMS sends a full **1920×1080** frame to HDMI.
3. Your matrix is drawn at **NATIVE pixel scale** at the chosen **Origin X / Y** (default 0,0 = top-left).
4. The rest of the 1920×1080 frame is the configured Background colour (typically black).
5. The Colorlight reads its mapped region of the 1080p signal and shows it 1:1 on the wall — zero scaling artefacts.

If your scaler is configured to read from a different position in the HDMI frame, set Origin X / Y to match. For Colorlight X1 defaults (mapped to top-left): 0,0.

Scene/Chase Builder

The Scene/Chase mixer surface. Four effect channels, each running two effect generators, mixed down through group mixes to a main mix to a master section. This is where you **design** your scenes and chases.

Signal flow



- **Slot labels: Background (under) ⇄ Foreground (over).** Each channel has two effect slots — the **Background** (bottom layer) on the left, the **Foreground** (top layer) on the right. The Channel Mix Mode keys/blends Foreground onto Background. Photoshop-style mental model: think layers, not "slot 1 / slot 2". ⇄ **swap button** between the two slots flips everything (effect, params, speed, region, invert) between Background and Foreground in a single tap — fastest way to fix "I put the text in the wrong layer".
- **Channel swap ⇅ within a group.** A small ⇅ button sits between each Group Mix and its lower channel — left side swaps Ch1 ⇅ Ch2, right side swaps Ch3 ⇅ Ch4. Moves the whole channel (both slots + region + mix mode + invert) up or down within its group. Useful when you decide late that the order of channels in the group fader should be reversed.
- **Mix-mode hover descriptions:** click any Mix dropdown and hover over the modes to live-preview them; the tooltip on each option spells out what the mode does. Click commits, Esc / outside-click reverts.
- **Effect dropdown** per slot — picks the generator. **Speed** slider next to it. **Edit** button opens the param editor (non-modal — previews stay live behind it). The Edit dialog **auto-widens to two columns** for effects with 7+ params. Inside the Edit dialog every control is reachable by **keyboard (Tab / arrows)** with a clear focus ring, so you can dial a look in without the mouse.
- **Copy / Paste effect** (the two icon buttons on each slot) — **Copy** (overlapping-pages icon) grabs this slot's effect — type, speed, invert and all its parameters — to a clipboard; **Paste** (clipboard icon) drops it into any other channel/slot as a starting point. A quick way to clone a look to another channel without saving a preset. Both flash a green tick to confirm. The Region is *not* copied (it's positional, not part of the effect).
- **Effect Presets** (top of every Edit dialog) — save the current params under a name, recall any saved preset back onto this slot with a single click. **Save As…** creates a new preset; **Save** overwrites the one currently picked in the dropdown; **Delete** removes it. Presets are **per effect** (Shape Gen presets only appear on Shape Gen slots, Plasma presets only on Plasma, etc.), live at the project level, and travel with `.lms` save files so you can share a preset library with another operator. Built-in factory defaults are unaffected — this is a top-up library on top of them.

- **Invert** checkbox per slot — colour-inverts the slot's output before mixing.
- **Mix mode** per stage — see Mix modes below.
- **Fader** blends the two inputs (0 = full first input, 1 = full second). Group + Main mixes have `<` `|` `>` snap buttons. Main has an **Auto Speed** slider that sweeps the fader automatically.
- **Previews** across the top: LEFT group / MAIN (final output) / RIGHT group. Black inner rectangle = matrix area inside the preview canvas.
- **Fixture view (📱 Show on monitors).** For odd-shaped rigs, the MAIN + Show-mode previews can show your *physical* layout instead of the full rectangle: open any patch row's footprint editor (Patch Editor → 📐) and click 📱 **Show on monitors** (lights green). Both big previews then black out every cell that isn't a real patched LED, so a custom map / ring / panel wall previews as its true shape, lit live. Whole-patch, persists across launches, click again to revert. See Output Patch below.
- **Monitor Settings — 1.1.3.** One button in the **MAIN OUTPUT** and **Show-mode** preview headers opens a small panel holding every "how should this preview look" control in one place: **Show** (Full Matrix or Custom Footprint), **Outline** (the patched area), **Pixels** (Square or Round) and **Gap**. They used to sit inline and crowd the header. Everything in this panel is **preview only** and changes nothing on the wire. Click the button again, click away, or press **Esc** to close it.
- **Round pixels — 1.1.3.** Under **Monitor Settings** → **Pixels**. The previews normally draw every pixel as a hard square, edge to edge. Choose **Round** and each pixel becomes a circle with a gap around it, which is a much fairer picture of what a real pixel wall looks like from the front — particularly on low-density rigs where the spacing between LEDs is a big part of the look. It is **preview only** and never touches a single byte of your output. It works on a plain rectangular matrix with no patch at all, so you do not need to build a footprint to get it. Two things worth knowing: it costs nothing per frame (it is a CSS mask, not drawn circles, so it is just as cheap on a 70,000-pixel wall as on a small one), and it **switches itself off when the pixels get smaller than about 5 screen pixels** — below that the gaps eat more than the dots show and the wall just looks dim and gappy. If you have a very large matrix in a small preview, that is why the dots do not appear. Previously this look was only available in Custom Footprint mode; leave the button alone and that older behaviour is exactly what you still get.
- **Pixel gap — 1.1.3.** The **Gap** slider in **Monitor Settings** sets the gap between pixels, from **0%** (touching, edge to edge) up to **50%** of each cell. Drag it until the preview matches the spacing on your actual wall. It works with **square** pixels as well as round ones — which is worth knowing, because a normal SMD panel is square emitters with black between them, so square-with-a-gap is often the more faithful picture. Both previews stay in step, the setting is saved with your show, and like everything else here it is **preview only** and changes nothing on the wire. At 0% with Round off you get the plain edge-to-edge picture LMS has always drawn.
- **"Why is my output black?" badge.** A black preview looks identical whether you've latched BLACKOUT, left the Master at zero, or genuinely broken something — so when the frame is black *for one of those two reasons*, the **MAIN OUTPUT** preview (and the big Show Mode preview) says which: a red **BLACKOUT** badge or an amber **MASTER AT 0%** badge. No badge and still black = it's the look, not a killed output — start checking the mix faders and the effects.
- **Live Edit-dialog repopulation:** while the Edit dialog is open on a slot, changing the effect dropdown reshapes the dialog's controls live — for browsing effects.

Mix modes

How the two inputs (A and B) combine at each stage. The fader controls how the mix blends; each mode interprets the fader differently. The dropdown is grouped into three sections, and you **hover** any mode to LIVE-preview it — click commits, Esc / outside-click reverts to your starting mode.

Direction ⇄ toggle. The Shape / Intensity / Overlay modes each have a direction — *which* input drives the effect. Instead of listing both directions as separate modes, pick the mode once and use the small ⇄ **toggle** that appears next to the dropdown to swap which input is the stencil / dimmer / top layer. Its label follows the stage: ⇄ **Background / Foreground** on a channel mix (slots side-by-side), ⇅ **Upper / Lower** on a Group mix (channels stacked — note the up/down arrow), ⇄ **Left / Right** on the Main mix.

> **The fader jumps when you pick a mode — that's deliberate.** Every mode has a "home" position where it visibly does what its name says, and picking one from a different group sends the fader there: **Progressive / Linear → the middle** (where both inputs show), **Dip to Black → 0** (the start of the transition — the middle would be a black screen), **everything else → full** (Layer, Combine, Stencil and Overlay all do *nothing* at 0 — you'd just see input A and think the mode was broken). Switching *within* a group never moves it, so a level you set on purpose survives.

Layer — put one side on top of the other. **Main mix only** (only the Main mix is physically left-and-right; the other stages use the ⇄ / ⇅ swap buttons and the Overlay mode instead):

Mode	What it does
Left on top / **Right on top**	The named side is the **top layer**: wherever it has content it draws on top **at its own colour**, never tinted by what's underneath; where it's black, the other side shows through. The fader fades the **top layer** in — 0 = the other side only, 1 = fully on top. **This is the mode for laying a border / logo / text over another look.** Don't use a Crossfade for that — Progressive adds both and clips, so your border comes out corrupted by whatever's behind it, and the only fader position where the border is clean is the one where the other side has vanished entirely.

Crossfades — slide A → B:

Mode	What it does
Progressive ⭐	Default crossfade — B fades in 0→0.5, A fades out 0.5→1. **Both inputs at FULL at the midpoint** (additive overlap, saturating). What operators expect when they say "crossfade".
Linear (50/50 at middle)	Standard linear blend. Fader 0 = pure A, 1 = pure B, half each at 0.5. Output dims through the centre.
Dip to Black	Clean dissolve — A fades out to **BLACK** over the first half of the fader, then B fades up from black over the second half. The two looks never bleed or add together. Black at the midpoint. The natural transition at the **Main mix** for cutting between two unrelated scenes.
None — equal display	Both inputs at FULL, fader ignored. Pure additive showing both effects.

Combine — arithmetic blends, the fader scales how much of B mixes in:

Mode	What it does
Add	A + B clipped at 255. Two coloured layers add into a third colour.
Screen	Softer additive — no harsh clipping near peak.
Multiply	A × B per channel. **Tints/darkens** — useful for colour washes.
Max (Lighten)	Per channel, brighter of the two wins.
Min (Darken)	Per channel, darker of the two wins.

Stencil & Overlay — one input shapes/reveals the other (use the ⇄ toggle to swap which is which):

Mode	What it does
Shape (mask)	One input's brightness reveals the other — bright areas show it through, dark areas go black. Text in the stencil + Plasma in the other = plasma-shaped text ("fill the shape"). A separate **Invert** tick flips the polarity: **dark** areas reveal instead of bright, so the shape is **cut OUT** of the effect (a stencil knock-out / silhouette) rather than filled.
Intensity	One input's per-pixel brightness DIMS the other. Plasma in the stencil becomes a dynamic dimmer for the colour in the other input.
Overlay	One input draws ON TOP wherever it has content (non-black); the other fills the gaps. Use for a logo/text overlaid on a plasma backdrop. The ⇄ toggle picks which layer is on top. (At the **Main mix**, prefer **Left / Right on top** — same idea, but it says which side plainly and its fader fades the top layer in.)

The stencil's COLOUR never reaches the output — only its brightness. This catches everyone. On a **Shape (mask)**, the stencil is a *shape*: LMS reads how bright each of its pixels is and uses that as the aperture. So a stencil at **full brightness** (white — or any pure colour like `#ff0000`, since it's the brightest channel that counts) lets the other effect through **exactly**, untouched. A **dim** stencil dims everything coming through it — a mid-grey stencil caps the effect at 50% brightness. And a **black** stencil can't work at all: a black glyph on a black background is an all-zero image, so the shape doesn't exist in the data for any mode to find. If your colour cycle looks washed out or half-bright through a mask, the stencil's colour is too dark — set it to white.

Metaballs make an excellent stencil. They render as *colour* × *field strength* — solid in the blob cores, feathering smoothly to nothing at the edges — so that falloff becomes the mask directly and you get cycle-filled blobs with soft, merging edges. Leave **Autocolor** running if you like: it always produces a full-brightness hue, so the mask is unaffected.

Centre column launcher strip

Three big gradient buttons — **Scenes**, **Chases**, **SHOW** — open their pop-out windows (Scenes / Chases) or jump to the Show Mode page.

Beneath them: a smaller row of two utility buttons:

- 💡 **Streaming** — master output streaming toggle. Lit (green border + yellow glow on the bulb) when streaming is ON; dim when OFF. Mirrors the master toggle in Settings → Output Streaming; the body class also gates the Matrix and Output protocol params in Settings so they can't be edited mid-show.
- ⚙ **Settings** — jumps straight to Settings without going through the File menu.

Scenes + Chases (pop-out windows)

The Scenes / Chases / SHOW buttons in the centre column open the corresponding pop-out window or jump to Show Mode:

- ＋ **Snapshot** — freezes the entire mixer state (all 4 channels, all mixes, master) as a named scene. Click any scene to recall. Double-click to rename. **Update** overwrites with current mixer — the button flashes green with an "Updated ✓" label so you've got visible confirmation the snapshot actually happened. **Del** removes (asks for confirmation first with a styled "Delete scene?" modal). Update / Rename / Delete all live-refresh the full-screen Show Mode grid AND any connected Tablet Remote clients in real time.
- **Chase** = ordered list of scene snapshots with per-step hold + crossfade. The row along the top is the chase itself: pick one from the dropdown, ✎ renames it, ＋ makes a new one, ✕ deletes it.
- ＋ **Add step** (under the list) appends the scene currently selected over in the **Scenes** pop-out. Reorder with the ↑ / ↓ arrows on each step; ✕ removes one. **Loop** = repeat vs one-shot. **Ticked:** the chase runs step 1 → last → step 1 forever, crossfading round the join. **Unticked:** it plays through once and then **parks on the last step, holding that look** — the scene keeps animating (its own effects run on regardless), but the step stops re-triggering and the chase never returns to step 1. The look stays on stage until you Stop it or play something else; a one-shot deliberately does *not* stop itself, because stopping would drop the output back to whatever is up in the Builder mid-show. (Fixed in beta.60b — the last step used to re-trigger every hold period, restarting the scene over and over instead of parking.)
- **Chase edits are live, and save themselves.** There is no Apply or Save button on a chase — there is nothing to save, because every change is already in. Every change — scene, Hold, Fade, Speed, Loop, step order, adding or removing steps, renaming — is written to the model immediately and auto-saved (debounced ~400 ms), so it survives quit and relaunch. The foot of the panel says **Changes save automatically**, and flashes a green ✓ **saved** each time one lands — so the behaviour is stated up front and confirmed as it happens, rather than being an invisible process you have to take on trust. Note this saves into the **app's own storage, not your project file**: to get chase changes into a `.lms` on disk you still need **File** → **Save** (or Settings → **Auto-save on close**). **Edits to a *playing* chase take effect instantly, mid-step** — retiming the current step's Hold to less than the time it has already run advances it there and then, and re-pointing the running step at another scene cuts to it with no fade. Reordering or deleting steps mid-run moves the playhead too, because it tracks a step *number*: delete step 1 while step 2 is live and the chase jumps to what is now step 1. All of this is deliberate — it's what makes a chase programmable while it runs — but it means the safe way to restructure a chase during a show is to **Stop it first**.

- **Blackout at end** (beta.60b, appears only when Loop is unticked) — instead of holding the final look, take the output to black when the chase finishes. The **ms** box is the fade: **0** snaps straight to black, anything else fades the last step out over that time. Use it for a walk-off, a set closer, or any cue that should end dark without you riding the Master down by hand. The fade is scaled by **Speed** along with every Hold and Fade, so running a chase at 200% ends it twice as fast too — the whole shape stays proportional. The chase stays *playing* once black (it doesn't stop itself, for the reason above); Stop, or play another chase, when you're ready to move on.
- **Hold ms** / **Fade ms** are per step, in **milliseconds** (1000 = one second). **Fade runs at the END of the hold, not in addition to it** — a step with Hold 800 / Fade 200 sits still for 600ms and then crossfades into the next step over the last 200ms. Keep Fade shorter than Hold.
- **Speed** is a live rate control for the whole chase — 10% to 400%, where 100% = exactly the times you typed. It scales every step's Hold *and* Fade together, so the shape of the chase is preserved and only its tempo changes. It **doesn't touch your numbers**, so it's safe to ride mid-set; double-click the word **Speed** to snap back to 100%. Right-click it to bind a MIDI controller (it also appears as **Chase speed** in the bindings table).
- While a chase runs, its ▶ **Play** button lights green and the playing step is highlighted in the list, with a **step 2 / 5** counter in the Steps header.
- Deleting a scene that a chase uses will **remove those steps too** — the confirmation names the chases affected before you commit.
- Pop-out windows are draggable by the title bar, close via the red ×, auto-close when you leave the Builder tab.
- Lists scroll after ~12 items.

Show Mode

The live-performance surface. Full-bleed (sidebar auto-hides when you enter), big touch-friendly buttons, all scene + chase triggers in one unified grid grouped by section.

Exiting Show Mode

There's **no Exit button** in Show Mode — exit by:

- Pressing `Esc`, or
- **Clicking the window X (close button)** — on any sub-page (Show Mode / Settings) this jumps back to the Builder rather than quitting.

Only the X on the Builder itself actually quits the app. PIN lockout (Performance Mode) swallows both — unlock first.

Side panel (controls)

- ◀ **back to builder** at the top — small text link (replaces the old button). Esc / window-X also work.
- **Preview** below.
- **Master fader** — overall brightness. Linked bidirectionally with the Builder's Master fader: drag either, both move.
- **Gamma slider** — perceptual-curve adjustment for the whole rig, sitting under Master. Range 0.30 – 3.00. Linked bidirectionally with the Builder's Gamma slider so you can recalibrate live without bouncing back to the Builder.
- **Blackout can take the DMX looks with it (1.2.4, Mike's request).** By default BLACKOUT kills the pixel output only, leaving any DMX fader looks running — right when the DMX is lighting something separate, wrong when it's part of the same look. There is now an option to have blackout kill the DMX looks too, so one button really does mean everything off.
- **BLACKOUT (L) / (F)** + **STROBE (L) / (F)** — Latch tap-to-toggle / Flash press-and-hold pairs. **Strobe Hz** slider below, linked bidirectionally with the Builder's Strobe Hz slider and dynamically capped at the Nyquist limit of your render FPS (max = FPS/2, hard ceiling 30 Hz). At 30 FPS the cap is 15 Hz; at 60 FPS, 30 Hz. Above the cap, strobes alias into beat patterns — neither slider will go past what your output rate can represent cleanly. The Show Mode strobe label is trimmed to just the Hz value to keep the controls column compact; Settings → General's strobe label keeps the max/FPS info for when you're tweaking render rate.
- **FADE IN / OUT** button — momentary fade-to-black then restore. Click from any level → fades DOWN to 0 (remembers where you were). Click again → fades back UP to that remembered value. Mid-fade click **reverses direction immediately** if you change your mind. Speed is set by the **Fade Time** slider below (live ms readout).
- ⇄ **SCENE X FADE** button — when on, scene recalls crossfade the previous graph into the new one over **X-Fade Time** (single-row slider below). Flash mode bypasses it.
- 📱 **Tablet Remote** button — opens the QR popup so you can scan with an iPad / Android tablet. See the Tablet Remote section.

- 🔓 **padlock** in the top-right corner — Performance Mode lock (open padlock = unlocked, tap to lock + set PIN; closed padlock = locked, tap to enter PIN to unlock). Replaces the old big "🔒 Lock" button.

Main Mix (Builder side, not Show Mode): the < / | / > snap buttons now **animate** over the duration set by the slider underneath (range 1 s on the left = fastest → 10 s on the right = slowest). The slider knob slides across visually during the fade and the on-screen value tracks the engine output. Drag the level slider directly to override / cancel an in-flight fade.

Trigger grid

One unified grid of Scenes + Chases, grouped by section. Each section header has a thick pink accent bar and a soft purple glow so you can scan groups at a glance from the operator position. Tap any button to play. Tapping a new scene/chase auto-stops the running chase (one-thing-at-a-time convention).

Reorder sections: each named section header has **▲ / ▼** buttons on the right — tap to move the whole section up or down in the grid. The order persists per project. (To reorder buttons *within* a section, Ctrl-drag a button onto another button of the same kind.)

Corner badges on every button (full words, not initials, since Bob kept forgetting what S / C / F / L stood for):

- **Top-left** — **SCENE** (lavender) or **CHASE** (pink)
- **Top-right** — **FLASH** (pink) or **LATCH** (dim grey)

Latch mode = tap to play, stays playing. **Flash** mode = press-and-hold, releases back to whatever was playing before. Flash also **peeks through BLACKOUT** — press a flash button with blackout latched and the scene shows during the hold, blackout returns on release.

Scene activation = ON-phase first frame. Triggering a scene (or stepping to the next chase step) resets the effect time-clock so any flash/strobe in the scene lands on its **ON** state for the very first rendered frame — and scrolling text starts at its defined entry position rather than mid-flight. Matters most for low-Hz flashed text used with FLASH-mode buttons + peek-through BLACKOUT: previously you could mash the button and get nothing on screen for ~500 ms if the strobe gate landed in the OFF window. Now the first thing the audience sees is always the visible state.

Note on touchscreens: Press-and-hold Flash works perfectly on mouse / keyboard / MIDI controller. On Windows capacitive touchscreens it's limited to ~500 ms (the OS touch driver quantises long touches to "tap" gestures regardless of when the finger actually lifts — a hardware-level limitation we can't override from JavaScript). For touchscreen-driven shows, use **Latch mode** instead and tap a different scene/chase to switch.

Right-click a button (or long-press on touch) for the setup panel

The menu is a **persistent panel** — a title bar (the button's name) with a ✕, and it **stays open** as you change settings so you can set several things on one button in a single go. It refreshes in place after each change; close it with the ✕, **Esc**, or by clicking away. (Actions that open their own dialog — Rename, the colour pickers, the Learn flows, Reset ALL — close it, since the dialog takes over.) Long menus lay out in **two columns** so nothing falls off the screen, with bold headers and clear dividers.

- **Rename…**

- **Text colour / Background colour / Pill colour** — draggable colour picker with a 16-preset palette plus a custom-colour swatch. **Clear** entries appear when an override is set. (Pill = the 6 px left-edge stripe; the active scene / playing chase also gets a bright outline in this colour.)
- **Font size ± / Toggle bold**
- **Mode: Flash / Latch** — swaps between tap-to-latch and press-to-hold.
- **Learn keyboard key… / Learn MIDI input… / Learn Art-Net/DMX in…** — bind a key, MIDI note/pad, or DMX channel to this button (see *External Control IN*). The Clear item shows the current binding next to it.
- **Link a DMX look…** — this button also fires one of your saved DMX fader scenes or chases, so the pixel scene and the conventional rig come up in one press. Linked buttons are marked **DMX LINKED** in gold. A **Flash** button restores the DMX levels on release; a **Latch** one leaves them up. Full detail under *DMX Faders*.
- **Section** — assign to one of 8 sections (or none). "Rename current section…" names the band (e.g. "Verse", "Chorus", "Drop"); named sections get a labelled header with ▲ / ▼ reorder arrows.
- **Hide / Un-hide** + **Show hidden buttons** — hide a button from the grid; the toggle reveals hidden ones (faded, dashed) so you can un-hide them.
- **Set as launch default** ⭐ — recall this scene/chase (or "nothing") automatically on the next launch.
- **Scene preview image** (scene buttons) — a small live preview in the corner of the button. **Show on every scene button** is the global default; then for **just this button** you can pick **Always show**, **Never show**, or **follow the global** — so you can preview a few key buttons without cluttering the rest. **Re-capture this scene's image** grabs a fresh still after you've tweaked the scene. Previews are captured at Snapshot / Update time (**zero per-frame cost**), the key/MIDI/DMX badges always sit *on top* of the preview, and the same global + per-button visibility is mirrored on the Tablet Remote.
- **Reset THIS button / Reset ALL buttons…** — clear overrides for one button, or wipe every button's colours / sections / hidden / flash (asks first).

Keyboard + MIDI control bindings

Trigger Show Mode hands-free from a keyboard or a MIDI controller. **Right-click any button → "Learn keyboard key…"** (or "Learn MIDI input…") then press the key / pad you want — it's bound. Press Esc to cancel a learn. One key does one thing; re-learning moves the binding. Bindings save with the project.

- **What's bindable:** every Scene and Chase trigger button, the action buttons — BLACKOUT (L), BLACKOUT (F), STROBE (L), STROBE (F), FADE IN/OUT, SCENE X-FADE — **and the faders** (Master, Strobe Hz, Gamma, Fade Time, X-Fade Time). Buttons take keyboard keys, MIDI notes or DMX on/off channels; faders take MIDI CCs or DMX fader channels (not keyboard). See *External Control IN* for the full picture + the Control Bindings table.
- **Latch vs Flash:** a key on a **latch** button taps it (toggle). A key on a **flash** button (BLACKOUT (F) / STROBE (F), or any flash-mode scene/chase) presses while you **hold the key** and releases when you let go — same as press-and-hold, but on the keyboard.
- **Badges:** the bound key shows as a small green badge in the top-right of the button so you can see your layout at a glance.

- **Safe by design:** keys only fire in Show Mode (not while typing in a field), and any key with **Ctrl / Alt / ⌘** is reserved for the app — only plain keys bind, so there's no clash with shortcuts. Held keys auto-repeat once (latch) or hold (flash).
- **Kiosk-aware:** bindings respect the same per-feature lockout as MIDI — Scenes/Chases always fire (the trigger grid stays live in kiosk mode); BLACKOUT / STROBE / FADE / X-FADE keys obey their lock-mask tick.

Performance Mode + PIN lockout

Tap the 🔓 **open padlock** in the top-right corner. First-time use prompts for a 4–8 digit PIN (then confirm). Sets the lock state. The padlock turns amber and switches to 🔒 **closed**.

When locked: the sidebar nav disappears, the back-link disappears, and the **native File / Help menus are hidden** (along with their keyboard shortcuts) so there's no escape hatch. The same top-right padlock (now closed + amber) is the only way back into the app — tap → PIN prompt → correct unlocks. Wrong PIN shakes the icon. Clicking the window **X while locked** shows a **"Close LED Matrix Server?"** confirm — Yes shuts the app down (the only way out of a locked kiosk besides the PIN), Cancel stays in the show, so an accidental click can't kill a live show.

Setting / changing the PIN: Settings → Performance Mode PIN. The Settings page is your admin surface — no current-PIN re-entry needed (you've already proven you're admin by reaching Settings).

Lock state + PIN persist across app restarts so a kiosk install boots already locked. **"Lock on next launch (kiosk mode)"** (Settings → Performance Mode PIN card) is **persistent** — once armed with a PIN set, the app re-locks into Show Mode on **every** relaunch, and the native File/Help menus stay hidden, until an operator unlocks with the PIN and unticks the box. Wrong-PIN does nothing destructive.

Per-feature lockout — choose which Show Mode controls to hide

The same Performance Mode PIN card carries a checklist of Show Mode controls. **Ticked = hidden in Show Mode; unticked = stays visible & usable.** Lets engineers tune the kiosk experience to the install — strip everything down to just the trigger grid + BLACKOUT for an end-user-facing club kiosk, or leave Master + Gamma visible if the booth operator still wants live-tweak access while the rest of the chrome is locked. (Lee 2026-06-07.)

- **Master fader** — default ticked. Hides the whole master fieldset (legend + slider).
- **BLACKOUT (L) + (F)** — default **unticked**. Panic button stays available even in lockdown, including via MIDI controller pad.
- **STROBE + Hz slider** — default ticked. Hides both STROBE (L) + (F) buttons AND the Strobe Hz row (label + slider + value).
- **Gamma slider** — default ticked. Hides the Gamma row entirely.
- **FADE IN / OUT** — default ticked. Hides the FADE button + the Fade Time slider row.
- **SCENE X FADE** — default ticked. Hides the X FADE button + the X-Fade Time slider row.
- **Tablet Remote button** — default ticked. Hides the entry point from the locked side panel so end users can't fire up a remote-control connection on their own device.
- **Right-click on Scene + Chase buttons** — default ticked. Suppresses the operator-edit context menu (rename / recolour / re-bind / section). Untick to allow right-click even in lock mode (advanced use).

Always-on lockdown behaviour (NOT per-feature toggleable, kept across all installs): Sidebar navigation hidden, page-X intercepted, Back-to-Builder button hidden, section reorder arrows hidden, right-click

context menus on the Master fader / BLACKOUT (the MIDI-binding menus, accessed by right-click in unlocked mode) suppressed. Scene + Chase buttons themselves are ALWAYS visible and ALWAYS tappable in lock mode (and MIDI scene/chase triggers fire too) — kiosks need the trigger grid live; that's the whole point.

Settings persist automatically. Each tick is debounced-written to localStorage within 400 ms; no manual Save button. A small green ✓ **saved** indicator flashes next to the section header after each change to confirm the write landed. Settings travel with the project file (`.lms`) so engineers can ship pre-configured lockmasks to install machines.

Regions (experimental — Regions Alpha)

Name your regions — 1.1.2. Every row in the **Region Manager** has an editable name field. Type what the region actually lights — *Bar*, *Truss L*, *DJ booth*, *Dance floor* — and that name replaces "Ch2 · Foreground" in the manager list and the info bar. Leave it blank and the slot position shows instead (in italics, so you can tell at a glance which regions you have named and which are still just wherever they happen to sit). Names are saved with the project.

Constrain a generator effect to a rectangular sub-area of the matrix. Combined with the Output Patch, this lets one LMS canvas drive multiple independent zones — different effects running in different "windows" of the same canvas, each routed to its own controller. Bob ran 23 controllers in 4 zones (pixel tubes, table centres, ceiling projectors, lanterns) off one LMS surface at his wedding using this exact pattern.

Setting a Region

On every effect slot row in the Builder, next to the Edit and Reset buttons, there's a **Region** button. Click it to open the config dialog:

- **Enable region** checkbox — when off, the effect renders full-canvas (default behaviour, unchanged from earlier versions). When on, the effect is confined to the rectangle below.
- **X / Y / Width / Height** — rectangle position + size in matrix pixels, clamped to the matrix dimensions.
- **Mini preview canvas** — shows where the region sits on the matrix. **Click + drag** on the canvas to position the rectangle by mouse instead of typing values.
- **Info line** at the bottom — shows the region's size as a percentage of the total matrix area.
- **Reset to full canvas** button — disables the region and resets X/Y to (0,0), W/H to matrix dimensions.

The Region button on the slot row lights up violet when a region is active so you can see at a glance which slots are confined.

How it renders

When a slot has an active region, LMS renders the effect at the **region's** resolution (not the full canvas), then blits the result into the main buffer at the configured (x, y) position. Pixels outside the region stay at zero. This is faster than rendering at full size and clipping — a heavy effect like Plasma or Fire in a small 32×16 region costs a fraction of what it costs across a 200×100 canvas.

Multi-zone workflow with Output Patch

Regions compose naturally with the Output Patch (see Outputs → Open Patch Editor):

1. Configure a big canvas covering all your zones (e.g. 200×100).
2. Use Regions on each effect slot to confine effects to the right sub-area for each physical zone.
3. In the Output Patch, add rows that map each region's universe range to its own protocol + IP + universe. Pixel tubes go to controller A, table centres to controller B, ceiling to C, lanterns to D — all driven by one LMS instance, one Scene snapshot.

One tap on a Show Mode scene button now changes every zone of your install simultaneously.

Channel mixes still work normally

The four-channel mix pipeline is unchanged — channels still blend at their existing levels and modes. Outside-region pixels are zero, so the blend math naturally produces "show whichever channel is non-zero here" at the boundary. You can deliberately overlap regions across channels for transitions or layered looks.

Preview overlay

Region rectangles are outlined on the three Builder preview canvases (LEFT, MAIN, RIGHT) in per-channel colours so you can see exactly where each region sits. The MAIN preview shows every channel's regions; LEFT shows channels 1+2 only; RIGHT shows channels 3+4 only.

Backwards-compatible

Existing scenes / projects without region data render full-canvas, exactly as before. No project migration needed. Scene snapshots now also capture region state alongside the effect graph, so swapping between scenes brings the right region layout for each.

Region Manager (whole-rig view)

The per-slot Region dialog is great for one slot at a time, but designing a multi-zone install you really want to see **every region at once** and align them visually. The 🗺 **Regions** quick button in the Builder centre column opens the **Region Manager** — a single dialog with:

- A big preview canvas showing every enabled region overlaid in its slot colour. The currently-selected region is drawn with a bright white outline so it stands out.
- A sidebar listing all 8 effect slots (Ch1 · BG, Ch1 · FG, Ch2 · BG, …). Click any row to select it; checkboxes on each row toggle enable + lock per slot.
- A toolbar at the top with four batch buttons:
  - ✓ **Enable all** — turn every region on (creates default full-canvas regions for slots that don't have one yet).
  - ✕ **Disable all** — turn every region off (geometry kept, just stops constraining — effects render full-canvas again).
  - 🔒 **Lock all** — pre-show safety: freeze the geometry on every region so accidental drags can't move them.
  - 🔓 **Unlock all** — release the freeze.

Selecting a region lets you **drag it around on the big canvas** (click-and-hold inside it) or **drag an edge or corner to resize**. Other regions stay visible underneath so you can align them. For paint-from-scratch and the X/Y/W/H number inputs use the per-slot Region button — the Manager is the bird's-eye view, not a replacement for the detail dialog.

Output Patch (Matrix → Open Patch Editor)

The Output Patch turns a single LMS canvas into a routing fabric for arbitrary numbers of physical fixtures — Art-Net controllers, sACN universes, DDP framebuffers, TPM2.net devices — each at its own IP, universe, and slice of the canvas. The basic flow is unchanged from earlier betas (one row per universe), but beta.40 layered three big features on top: **per-row footprints**, **variable-length universes**, and **diagonal LED strips**.

Coverage indicator. The summary line above the patch table reports **what percentage of the matrix is actually patched** — computed per-pixel against the real matrix cells, so footprint rectangles, diagonal strips and plain linear rows all count correctly. It turns **amber** when there are *unpatched* pixels or a footprint runs *off the edge of the matrix* (e.g. after shrinking the matrix), and green when the whole matrix is covered. It also flags how many pixels are **overlapped** by more than one row. This is the explicit link between the Matrix setup tab (which sets the canvas size + cabling) and the Patch editor (which carves it up).

Patch columns — what each one means

- **Universe** — the 16-bit universe number this row outputs to. **Art-Net** accepts 0–32767 (15-bit); **sACN** accepts 1–63999. The read-only **Net:Sub:Uni** column to its right shows the same Art-Net value the classic way — *Net (0–127) : Sub-Net (0–15) : Universe (0–15)* — because many receiving nodes display the address that way; it's a cross-check that host and node agree. sACN has no such split (always a flat 16-bit number), so it shows `–`.
- **First pixel** — the **0-based PIXEL index** (NOT a DMX channel) where this row/universe starts, counted along the matrix's cabling walk. Row 1 normally starts at pixel 0, the next at pixel 0 + Pixel count, and so on. Internally the byte offset into the DMX stream is `First pixel × colours-per-pixel` (3 for RGB, 4 for RGBW), but you think in pixels — LMS does the channel maths. (When a Panel Footprint is set on the row, First pixel is ignored and the panel reads its own region instead.)
- **Pixel count** — how many pixels this universe carries. **This drives the universe length put on the wire:** `Pixel count × colours-per-pixel` (3 RGB / 4 RGBW), rounded up to an even number per the Art-Net spec, capped at 512 channels. So a 100-pixel RGB row sends a universe of **300 channels**, not 512 — LMS does **not** pad to 512. Receiving nodes read this per-universe length to know how much pixel data the universe holds.
- **Order** — the colour byte order for this row (RGB, GRB, RGBW, etc.), overriding the matrix default. RGBW orders are 4 colours/pixel, the rest are 3 — which is the "colours-per-pixel" used in the maths above.

About "Pixels / Universe" (toolbar): this field is only a *convenience for generating rows* — + Add Row, Auto Number and the wizards use it as the default Pixel count when they create rows. It does **not** override the length of what's actually output: once a row exists, its on-wire universe length always comes from that row's own **Pixel count × colours** as described above. (Some controllers expect every universe at a fixed 512; if yours does, set Pixel count so 512 ÷ colours pixels fill it — e.g. 170 px RGB = 510 channels, the practical max.)

DMX (USB) rows — output straight to a USB dongle

Set a row's **Protocol** to **DMX (USB)** to send it out an ENTTEC/DMXKing-style USB dongle instead of over the network. First connect the dongle on the **Protocol Settings** → **Serial DMX** card (Connect + tick **Send to Serial DMX**); the row goes live when Streaming is on. If the dongle isn't connected, an amber banner above the table reminds you.

- A USB dongle is a **single 512-channel universe**, so **Universe**, **Target IP** and **Loopback** all read `n/a` — the IP cell shows 🔌 **USB**. There's nothing to address; everything goes to the one connected dongle.
- **First pixel becomes the DMX start channel.** Unlike network rows (where First pixel is ignored once a footprint is set), on a DMX row it stays editable and sets where this fixture's data lands in the 512 channels — pixel 0 = channel 1, pixel 1 = channel *colours+1*, etc. That's how you slot several fixtures into the one universe.
- All DMX rows **compose into one 512-channel frame** and go out together each frame — give each fixture a footprint (which canvas pixels feed it) + an Order (RGB / GRBW / …) + a start channel, and they share the universe cleanly. Anything past channel 512 is clipped.

Per-row diagnostic buttons

Each Output Patch row carries three small diagnostic actions, all bypassing the streaming engine so you can check reachability without going live:

- **Ping** — ICMP ping to the row's IP. Spinner while pinging, turns green with the RTT in ms when reachable, red ✗ on timeout. Auto-resets to "Ping" after 6 s. Hover for the full hostname + reason.
- 🔆 **Test / Identify** — a **latching toggle**: click once to stream **50% white** at this row's universe + IP (button turns a pulsing green ●), click again to turn it off (it then sends a final all-zero frame so the receiver doesn't latch at half white). It **stays on until you turn it off**, so you can walk to the rig and check the panel. Bypasses master streaming / Send-to toggles; honours per-row colour order + footprint pixel count. You can have several rows under test at once to compare panels. Auto-stops if its row is deleted or you turn master streaming on.
- 📐 **Footprint** — opens the Panel Footprint editor for this row (see below).

Plus the existing ↑ ↓ / 📋 / ✕ row controls. A small **Streaming** toggle on the Patch Editor toolbar mirrors the master gate so you can flip everything on/off without leaving the page.

For deeper diagnostics: combine **Ping** (controller reachable?) → 🔆 **Test** (packets going on the wire?) → Wireshark or DMX-Workshop on the destination subnet (packets actually arriving at the controller?). Walks you from front to back without leaving LMS for the simple checks.

🎯 Discover devices — find Art-Net controllers on the network

The 🎯 **Discover devices** button on the Patch Editor toolbar broadcasts an **Art-Net ArtPoll** and lists every Art-Net controller that answers — its **name, IP, MAC and the universes it's bound to** — in a pop-up (scans for ~3 seconds). Click a universe chip (`u3 +`) to add a patch row for that controller IP + universe, or **+ Add all** to patch every universe it reported in one go (duplicates are skipped, pixel count comes from your Pixels/Universe field). Rescan any time. If nothing answers, check the controllers are powered, on the same subnet/NIC as this PC, and have Art-Net enabled — some only reply once their output is active. (sACN has no equivalent active poll. **DDP** has its own **Scan for DDP devices** button on

the Protocol Settings → DDP card, which now runs **two** scans at once: a DDP status query, *and* an **mDNS browse for WLED controllers**. Those are different mechanisms — WLED is a pixel receiver and deliberately never answers a DDP status query, which is why this scan used to come back empty for the most common controller LMS supports. Anything it finds over mDNS is listed with its **name, IP, firmware version, board and LED count**, and the **+ DDP row** button is pre-loaded with that pixel count so you never have to count your own LEDs. A device that advertises but does not answer its web API is still listed — knowing the IP is most of the value — and says so. If nothing appears, check the controllers are powered, on this same subnet, and that UDP 5353 is not blocked by a firewall.)

DDP output specifics. DDP rows compose **per destination IP**: multiple rows pointed at the *same* controller fill one framebuffer (multi-zone layouts), while rows pointed at *different* controllers each get their own frame — so a single 169-px controller receives exactly its 507 bytes, not the whole matrix. The data-type byte is spec-correct (`0x0B` RGB / `0x1B` RGBW per 3waylabs, though most receivers ignore it). **Sync receivers** (DDP card checkbox) is the DDP equivalent of ArtSync / E1.31 Synchronization: for multi-controller rigs it sends each controller's data without an immediate display, then a synchronised PUSH to all so they latch the frame together with no inter-controller tearing. Leave it off for a single controller — each frame already latches on its own last chunk.

🧱 Wall Wizard — tiled-panel walls in one click

Tiled walls built from N identical panels (e.g. Mike's 4×4 wall of 13×13 Smartshow panels = 16 universes) used to mean manually adding 16 rows, footprinting each one, and getting the universe order + per-panel start corner + zig-zag right for every panel. The 🧱 **Wall Wizard…** button on the Patch Editor toolbar does it in one shot.

Click it and you get a small modal with these fields:

- **Panel preset** — pre-canned wiring conventions so most operators never touch the per-panel fields. Options: **Smartshow Pixel Panels** (DI bottom-left, snake rows — the default, matches Smartshow's panels + most consumer-grade WS2812-style boards) / **Top-Left + zigzag rows** / **Top-Left + plain rows** / **Bottom-Left + plain rows** / **Custom**. Picking a preset auto-fills the per-panel Start corner + Pixel wiring fields. A small italic line below the dropdown shows what the preset resolves to (e.g. `→ Start: Bottom-Left · Wiring: Rows zig-zag`) so it's obvious what you're getting without expanding to Custom.
- **Panel width / height** — pixel dimensions of ONE panel (13 × 13 for Smartshow's standard panel).
- **Panels across / down** — grid layout (e.g. 4 × 4 = 16 panels total).
- **Start universe** — first universe number; panels number through `start + N − 1`.
- **Pixels / panel** — read-only, auto-computed (W × H). Panels that don't fit one universe are now **auto-split** — see below.
- **Controller IP** — pre-fills from your global Art-Net host; editable per-row in the patch table after generation.
- **Protocol** — Art-Net / sACN / DDP.

- **Scan order (panel→panel)** — how the wiring runs from one panel to the next on the wall. *Row-major* (left→right, top to bottom — the default, matches most operators wiring the top-left panel as #1 then daisy-chaining right then dropping a row), *Row-major SNAKE* (right→left on alternate rows — matches the common LED-panel daisy-chain where the data cable folds back on each row), or the two column-major variants.
- **Custom only — Per-panel pixel wiring** + **Panel start corner**: hidden when a preset is picked; revealed when you select Custom for weird panels. Same semantics as the per-row Footprint editor.
- **Live preview line** — updates as you tweak inputs: "*Wall: 4×4 of 13×13 panels → 52×52 canvas, 16 universes (1 → 16). Each universe carries 169 pixels.*" When the matrix would resize, the preview line ALSO calls out the change: "*Matrix will resize from 32×16 → 52×52.*" — visible before you click Generate, so an accidental tick of the resize box doesn't surprise you.
- ☑ **Resize matrix to fit** — when checked (default), sets the matrix W/H to `cols × panelW` × `rows × panelH` automatically so you don't have to set the matrix size manually. Untick if you want to keep your existing matrix size — the preview line warns you when the box is set vs the existing matrix dimensions.
- ☑ **Replace existing patch** — when checked (default), wipes the current patch and replaces it with the generated rows. Uncheck to append.

Hit **Generate patch** and the rows appear in the patch table, each with the correct universe number, per-panel footprint (x/y/w/h, start corner, pixel walk), and IP. Edit per-row settings afterward if needed — the wizard is a fast-path for the common case, not a lock.

Big panels are auto-split (Lee 2026-06-12). If a panel holds more pixels than one universe can carry (**170** RGB / 128 RGBW / 85 RGB-16 / 64 RGBW-16, derived from your pixel order + bit depth), the wizard now **splits each panel across the universes it needs** — e.g. a 64×13 = 832-pixel panel becomes 5 rows of ≤170 px each, with sliced footprints (walkOffset/walkLength) on sequential universes. No more "split it by hand" abort. The live preview shows the real total, e.g. "*128 panels × 5 universes each = 640 universes (1 → 640), auto-split at ≤ 170 px/universe.*" (DDP carries whole frames, so it's never split.) If the total would exceed the protocol's universe ceiling (Art-Net 32767 / sACN 63999) the wizard stops and tells you.

⚠ **HDMI is the right tool for big walls.** Once a wall needs many universes (the preview warns past ~64) you're into heavy Art-Net/sACN territory — hundreds of universes is a firehose of UDP and a rack of controllers. For large walls, use **HDMI output** (Settings → Outputs → HDMI) instead. The Wall Wizard will happily generate a 640-universe patch, but the preview nudges you toward HDMI when the scale gets silly.

Mike's exact rig in three clicks: 🧱 Wall Wizard → defaults are already 13×13 × 4×4, Smartshow preset → Generate. 16 universes patched, matrix set to 52×52, ready to stream.

Panel footprints — patch a sub-region of the matrix per row

Each Output Patch row used to read a contiguous slice of the matrix's global pixel stream — fine for one big snake, useless for a wall built from multiple panels with their own internal wiring. The 📐 button on each row now opens the **Panel footprint editor**:

- **Shape** — *Rectangle* (Origin + Width × Height, below) or **Contiguous run** *(1.2.4, Bobby's request)*. See the note under this list.

- **Origin X / Y** — where this panel's top-left sits on the matrix.
- **Width × Height** — panel dimensions.
- **Start corner** — where pixel 0 of the universe lives (TL / TR / BL / BR).
- **Pixel walk** — single dropdown combining scan direction + zig-zag: *Zig-zag horizontal* (rows, alternate L↔R) / *Lines horizontal* (rows, all same direction) / *Zig-zag vertical* (columns, alternate ↑↓) / *Lines vertical* (columns, all same direction).
- **Live snake preview canvas** — shows the actual pixel walk as a bright purple line, green dot = pixel 0, red dot = last pixel.
- **Matrix-map overlays** (Bob's request) — under the matrix map: **Show live output (dimmed)** overlays the actual running output behind the panel rectangles so you can see where content lands relative to your panels, and **Reference image** lets you **Upload…** a venue photo / plan to loosely map your panels against. Both draw dimmed beneath the footprint rectangles; the reference image is saved with the project, and the toggles are remembered.

Contiguous run — filling a universe that doesn't end on a rectangle (1.2.4). A universe boundary almost never lands on a tidy rectangle. On a 32-wide serpentine matrix, 170 pixels (one full RGB universe) is *five complete rows plus ten pixels into the sixth* — an L-shape that Width × Height simply cannot describe. Previously the only way out was a hand-drawn Custom Map.

Set **Shape** → **Contiguous run** and the Width/Height boxes are replaced by **First pixel** and **Pixel count**. The row then walks the *whole* matrix in the chosen **Start corner** + **Pixel wiring** order and takes exactly that slice of it. So a 256-pixel wall becomes two rows — *first pixel 0, count 170* and *first pixel 170, count 86* — and the second row picks up precisely where the first stopped, mid-row or not.

The preview canvas, the row badge and the universe totals all understand the new shape, so what you see matches what goes on the wire. The per-panel universe-split fieldset is hidden for these rows, since the pixel count already says where the split falls.

Arranging panels on the matrix map

The matrix map is interactive — you can lay panels out visually instead of typing X/Y for every one:

- **Drag to move** — grab any panel rectangle and drag it; click a *different* panel to switch to editing it (no need to close and re-open). A faint **pixel grid** is drawn across the matrix so the squares inside a panel count out its pixels.
- **Move: Panel / Row / Column** — a toggle above the map. **Row** drags every panel aligned in that horizontal band together; **Column** does the same vertically — ideal for shuffling a whole row/column of a tiled wall at once. (Mouse shortcut: hold **Shift** = row, **Ctrl** = column, regardless of the toggle.) The dragged group is outlined in bright cyan so it clearly floats above the static panels.
- **Parking margin (off-air)** — the dim, hatched border around the matrix labelled **PARKING**. Drag panels (or a whole row/column) out there to set them aside while you rearrange, then drag them back. Pixels parked off the matrix simply aren't output, so it's a safe staging area — and it gives a full-size wall somewhere to move things to.
- **Snap to panel grid** — tick it and dragging snaps to a panel-size grid so identical panels tile edge-to-edge with no overlap or gaps.

- **Overlap warning** — if a panel covers the same pixels as another patched row, a red banner names the clashing universes (those pixels get sent to both panels). Uncovered matrix cells are dimmed so gaps are obvious; a linearly-patched matrix reads as covered, not a dark void.
- 📱 **Show on monitors (fixture view)** — the footer button (violet; lights **green** when on) sends your patched **shape** to the big previews. With it on, the Builder **MAIN OUTPUT** preview and the **Show-mode** preview black out every cell that isn't a real patched LED — so a custom map / ring / panel wall previews as its true physical shape, lit live, instead of the full rectangular canvas. It's a *whole-patch* view (not just the row you're editing), the choice persists across launches, bridge/null pixels stay dark, and clicking again returns to the normal full-canvas preview.

When a footprint is set, the row's `Start pixel` and `Pixel count` fields go disabled and the panel sends bytes in panel-local order instead of slicing the global canvas. The global Cabling dropdown on the Matrix card no longer applies to that row. Rows WITHOUT a footprint behave exactly as before — fully backward-compatible.

Practical example: a 2×2 wall of 13×13 SmartShow panels (26×26 = 676 pixels). Add 4 rows — universes 0, 1, 2, 3 — each with a 13×13 footprint at the panel's matrix position and the right start corner / zig-zag for that panel's internal wiring. Each panel now sends its own 169-pixel universe regardless of how the surrounding panels are oriented.

Variable-length universes + split-into-N helper

Each row's `Pixel count` can be any value 1–170 (Art-Net per-universe limit) — useful for SmartShow Pro-ONE-style interfaces that accept arbitrary universe lengths (e.g. 165 pixels / 495 channels). When using a footprint, the editor now also exposes:

- **Start at pixel** — how many pixels into the panel walk this universe starts at.
- **Pixels in universe** — how many pixels this universe carries (leave blank = the whole panel).

One physical panel can therefore be chunked across multiple universes — e.g. a 1200-pixel panel split across 8 universes of 165 pixels each — and LMS sends the right slice of the panel walk to each universe.

The **Split into universes…** button automates this: open the footprint editor for the row that defines your panel, click Split, set "Pixels / universe" (default 165) and the starting universe number, and LMS replaces the seed row with N rows in-place — one per universe — each pre-filled with the right walk-offset, walk-length, IP, protocol, and colour order. The badge on each resulting row reads `📐 W×H @ x,y · px 0–164`, `· px 165–329`, etc., so it's obvious at a glance which chunk each universe carries.

Diagonal LED strips

Some installs aren't axis-aligned — LED bars mounted at angles, diagonal pixel strips, anything where a rectangle doesn't describe the physical layout. The footprint editor's **Shape** dropdown switches between:

- **Rectangle / panel** — the default; everything above.
- **Diagonal strip (line A → B)** — define two endpoints (Start X/Y, End X/Y) and an LED count. Each LED is evenly spaced along the line and samples whichever canvas pixel sits under its computed (x, y) position.

A 60-LED bar mounted from (4, 30) to (28, 12) on the canvas? Two endpoints, count = 60, done — no rotation matrix needed. Pixel 0 sits at the Start end. Combine with the walk-offset / walk-length fields

and you can split a long diagonal strip across multiple universes the same way as a rectangular panel.

Per-row Universe Economy

Each protocol row honours the matching protocol's Universe Economy setting from the main Outputs card. When ON, a universe only re-sends if its data has changed since last frame (plus a 1-second keep-alive). Footprint rows are change-detected the same way — same per-universe accounting. Useful on sparse / blacked-out scenes and crowded networks.

Patched row gating

Patched rows self-gate via their per-row **Enabled** checkbox + the master streaming gate. The per-protocol **Send to …** toggles on Settings → Outputs are for whole-frame (non-patched) mode — they do NOT mute patched rows. This is the right mental model: if you've added a row to the patch with a valid IP, you want it to send. To silence a specific row, untick its Enabled checkbox. To silence everything, master streaming OFF.

Untick a row's Enabled checkbox = that panel goes DARK (LMS sends an all-zero frame to that universe, Universe Economy then dedupes so it's a single keep-alive zero-frame per second, panel stays black). Previously a disabled row was skipped entirely — the receiver kept showing whatever it last drew. Re-tick the row and the next live frame lights it back up. This matches the master streaming OFF semantics (3-sweep zero pulses across every universe).

Pixels / Universe — auto-derived from pixel order + bit depth

The **Pixels / universe** input on the Patch toolbar (used by ☑ Add Row and the wizards) now defaults to whatever fills one full universe given your matrix's pixel order + bit depth: **RGB 8-bit = 170**, **RGBW 8-bit = 128**, **RGB 16-bit = 85**, **RGBW 16-bit = 64**. Auto-derived from 512 DMX channels ÷ bytes per pixel. Override if your receiving controller expects a specific value (e.g. SmartShow Pro-ONE wants 165) — once you've typed a non-default value, LMS leaves your choice alone. Bob suggested this 2026-06-07.

🔒 Patch is locked while streaming is ON

When **OUTPUT STREAMING** is ON, the patch table goes read-only — all toolbar buttons (+ Add Row / 🧱 Wall Wizard / ↑↓ Auto Number / 📥 Import patches / 🗑 Clear ALL Patches), every row's Protocol / Universe / IP / Start / Count / Order / Footprint / 🔆 Test / 📋 Copy IP / ↑↓ reorder / ✕ Delete are greyed and unclickable. An amber banner appears above the toolbar explaining why. The intent is operator safety — a stray click during a live show shouldn't reassign a universe, retarget an IP, or delete a row.

Three things stay **live** during streaming: the per-row **Enabled** checkbox (your live-show "blackout this panel" gesture), the per-row **Ping** button (read-only diagnostic, can't change anything), and 🔍 **Discover devices** — an ArtPoll is a read-only broadcast query, so you can hunt for a controller mid-show. Only *adding* a discovered device as a patch row waits for streaming to stop. 📤 Export patches stays live too (it only writes a file); 📥 Import waits, since it rewrites the patch. To edit anything else, switch **OUTPUT STREAMING** off at the top of the page.

🔁 Loopback test column — redirect rows to 127.0.0.1 without touching the IP

Each row has a **Loopback** checkbox column. Tick it and the row's send goes to `127.0.0.1` (loopback) instead of the configured IP — so you can pair the Output Patch with the Output Probe (set to Bind IP = 127.0.0.1) for visual testing without touching your real IP addresses. Bob's original ask: previously the only way to test a 16-row patch with the Probe was to edit 16 IPs to 127.0.0.1, then edit them all back — and if you typo'd a restore, that row silently sends to the wrong target. Loopback mode keeps the IP field untouched.

- **Per-row 🔁 column** — tick the checkbox to redirect just one row. Pink chip lights, "→ 127.0.0.1" badge appears next to the chip, IP gets struck through in the IP column, whole row gets a pink left stripe + faint tint.
- **🔁 Loopback all (toolbar)** — flip every row in one click. Second click restores all rows. Per-row toggles still work independently if you only want, say, panel 5 redirected.
- **Page-level pink banner** at the top of the patch shows the count: "*3 of 16 rows are redirected to 127.0.0.1*". Hard to miss before you walk into a venue.
- **Transient state** — never travels with `.lms` files. Cleared on app boot, project load, and stripped pre-save. So sharing a project file mid-loopback is safe; the recipient gets the real IPs.
- **Locked while OUTPUT STREAMING is ON** — same as the rest of the patch config. To enter / exit loopback, switch streaming off first.
- **Send-side coverage**: Art-Net + DDP per-row IP overrides honour the loopback flag. sACN (unicast/multicast) doesn't accept per-call target IPs today — sACN multicast already routes locally, so loopback is implicit; sACN unicast loopback is a future follow-up if anyone asks.

The send-side IPC handlers also auto-init the protocol instance on first call if you haven't clicked Apply on the global Outputs card. So a brand-new project with patched rows and never-touched Settings still puts bytes on the wire as soon as you add a row with a valid IP and turn streaming ON.

Art-Net spec compliance — per-universe sequence

The ArtDmx Sequence field (byte 12 of the packet) is now incremented **per-universe**, not globally — universe 0 sees `1, 2, 3…`, universe 1 sees `1, 2, 3…` independently. Matches the Art-Net 4 spec's per-universe reordering-detection model so strict receivers (e.g. some ESP32 receivers that drop packets where the sequence falls out of progression) accept every packet. Tolerant receivers keep working too.

Back to the Builder

The ◀ **Back to Builder** button (and `Esc`) returns you to the Scene/Chase Builder, the same as every other page. The Patch Editor is also remembered as your last Settings view — so **File** → **Settings** reopens straight back into the Patch Editor until you switch to another Settings tab.

Output Probe (Help → Output Probe…)

Standalone window that listens on a chosen protocol port and visualises every universe received. Bypasses external tools (sACNView / xLights / WLED / Wireshark) when diagnosing what LMS is actually putting on the wire.

- **Protocol selector** — Art-Net (UDP 6454) / sACN E1.31 (UDP 5568 multicast) / DDP (UDP 4048) / TPM2.net (UDP 65506). Port defaults are set automatically; editable in case your sender uses a non-standard port.
- **Start / Stop / Clear universes** buttons.
- **Status strip** — listening state (green when active, with protocol name), bound address, total packets received, live packet rate (pkt/s), last source IP:port, number of distinct universes seen.
- **Per-universe rows** — each incoming universe gets a card showing its label, byte count, pixel count, source IP, per-row packet rate, plus a live colour-strip canvas rendering each pixel as a 1-pixel-wide band (image-rendering: pixelated for crisp scaling).

Protocol parsers

- **Art-Net** — validates the "Art-Net" magic header + opcode 0x5000 (ArtDmx), extracts universe (15-bit), sequence, length, DMX payload.
- **sACN (E1.31)** — joins 239.255.0.0 multicast on the listening NIC, validates start code 0x00, extracts universe at byte 113 (big-endian), DMX payload from byte 126.
- **DDP** — parses the 10-byte header (flags / push bit, config = 0x01 RGB or 0x03 RGBW, data offset, length).
- **TPM2.net** — validates 0x9C start byte + 0xDA data-frame type, extracts payload between the 6-byte header and the 0x36 end byte.

Diagnostic flow

1. **Patch row Ping** — controller reachable?
2. 🔆 **Test / Identify** — LMS emitting on this row?
3. **Output Probe** — packets actually receiving + decoded correctly at the destination?

All three steps work without leaving LMS. The Output Probe specifically lets you target your own machine: in another window or another LMS instance set a patch row's IP to your machine's LAN address, start streaming, and watch every universe show up in the probe.

🧱 Wall view — virtual tiled-panel preview (no hardware needed)

The default view paints each received universe as a flat colour strip — perfect for protocol-level debugging but hard to map back to "what does this look like on my wall?". The **View: Strips / 🧱 Wall** toggle in the top-right of the controls bar switches to a composite virtual wall mode that paints every received universe at its correct physical panel position. Pair with LMS → Loopback (Bind IP = 127.0.0.1) for an end-to-end test of a multi-panel rig **without owning the hardware**.

Wall view exposes the same shape parameters as the Wall Wizard so the receive side interprets the same data the same way the wizard sent it: **Panel preset** (Smartshow / Top-Left + zigzag / Top-Left + plain / Bottom-Left + plain / Custom), Panel W / H, Cols, Rows, Start universe, Scan order (panel→panel),

Display scale (Fit window or fixed 4× → 32×), Panel gaps toggle (1-pixel dark gutters between panels for visibility), Smooth (turns off pixel-snap rendering for a soft "LED wall from a distance" look), and a ⛶ **Fullscreen preview** button that hides every bit of Probe chrome — Esc returns.

What the Wall view actually does under the hood: it **un-walks** each panel's byte stream back to canvas-local pixel positions using the configured start corner + scan + zig-zag. This is the inverse of how LMS's footprint walker emitted the bytes on the way out — and exactly what the physical LED wiring does automatically. So Wall view shows what the physical panels would display, not the raw byte order on the wire.

Diagnostic bonus: flip the Probe's Panel preset to something different from what the Wall Wizard generated and you'll see the wall render mirrored / row-flipped — exactly the visual symptom of a wrong wiring assumption on a real install. The Wall view doubles as a teaching tool: operators can deliberately mis-pick presets to learn what wrong-wiring looks like, so they recognise it when their physical wall does it.

The wall pixel cache is populated on every incoming frame regardless of which view is active, so switching from Strips to Wall is instant — no need to wait for a fresh round of frames.

Tablet Remote

A WebSocket-powered companion UI for iPads / Android tablets. Operator fires scenes & chases, master, BLACKOUT, flash buttons from anywhere on the same Wi-Fi. **Multi-tablet safe** — every connected device sees the same live state, in sync with the desktop.

How to open it

1. Enter **Show Mode**.
2. Tap the **Tablet Remote** button on the side panel.
3. A draggable floating popup appears with a **QR code**, the primary URL, the connected-tablet count, and a Stop Server button.
4. Scan the QR with the iPad's camera (or open the URL in any browser on the same Wi-Fi).

The popup auto-starts the server on first open and auto-closes if you leave Show Mode. Drag it anywhere by the title bar; it remembers its position for the session.

On the tablet

- **Scene + chase buttons**, grouped by the same sections as the desktop. Tap **fires on press** (instant, like a lighting desk — no waiting for finger-up). Flash buttons are press-and-hold.
- Each button shows its name with a **LATCH / FLASH / CHASE** label, and — for scenes set to show a preview — a small **thumbnail** on the right. The same global + per-button preview visibility you set on the desktop is mirrored here.
- **Master fader** — drag to set overall brightness; it moves the desktop's Master (and the Builder slider) live, and vice-versa. **BLACKOUT (Latching)** button below.
- State stays in sync across every connected tablet + the desktop — fire something anywhere and they all update.
- **Add to Home Screen** (iOS Share menu / Android install prompt) for a full-screen, app-like launcher.

Settings card config

- **Port** — TCP port the HTTP+WS server listens on. Default 8080.
- **Auto-start when entering Show Mode** — ticked by default for fresh projects. Fires the server up automatically the moment you tap Show, so the button just opens the QR without a separate "start" step.

The QR popup itself

- **Server status pill** (top) — green "Server running · port N · M NICs" or red "Server is not running".
- **Primary URL** — prefers an IPv4 address (works on Windows, iPad, Android). The mDNS `lms.local` alias is listed in the "Other URLs on this network" section as a friendlier alternative for devices with Bonjour.
- **N tablets connected** live count — green when at least one tablet is talking.
- **Self-test link** — opens `http://127.0.0.1:<port>/` in your default browser. If that works but your tablet can't connect, the issue is firewall or Wi-Fi.

- **Allow through Windows Firewall** — one-click button that elevates and adds the inbound rule for the port. You'll see a UAC prompt — click Yes. Persistent: only needed once per machine.
- **Stop Server** — shuts the WS server down and closes the popup.

The Show Mode side-panel button shows a green dot when the server is running and a 👥 N badge when tablets are connected.

> **"Localhost works but the tablet can't connect"** is almost always Windows Defender Firewall blocking inbound connections. If there are conflicting *Allow* + *Block* rules on the same EXE, Block wins. Click the popup's Allow-through-Firewall button, or run `Get-NetFirewallRule | Where { $_.DisplayName -match 'electron' -and $_.Action -eq 'Block' } | Remove-NetFirewallRule` in an elevated PowerShell.

DMX Faders

Up to thirty-two raw DMX channels you can drive by hand — show 6, 12, 24 or 32, from **Output → DMX Faders…**. It is for the jobs that do not need a lighting desk: bump a hazer, raise a pair of uplighters, hold a warm wash behind the wall. **Raw channels only** — there are no fixture profiles, no patch and no cue stack, and that is deliberate.

Getting output

1. Pick **Output**. **Serial DMX** (a USB dongle) is the default and the usual choice for the fixtures this page exists to drive. **Art-Net** needs a universe *and* a target IP — ⚠ leave the IP blank and it falls back to the global Art-Net host, which is very often the wrong subnet: the faders move, the page says *Sending*, and nothing happens on stage. If Art-Net looks dead, check the target IP first.
2. Pick how many **Faders** you want — **6, 12, 24 or 32**. Hidden channels are **not transmitted**, so fewer faders means a genuinely smaller DMX footprint, not just a tidier screen.
3. Set the **Start channel**. The faders occupy that channel and the ones after it — at 24 faders, 300 gives you 300–323. The highest start moves with the bank: **481** at 32 faders, **489** at 24, and as high as **507** at 6, because the span always has to fit inside 512.
4. On Serial DMX, press **Connect dongle…** if it appears. A dongle cannot connect itself — the browser layer requires a click.
5. Opening the page starts sending. **Go live** / **Stop sending** toggles it.

The pill tells you the truth. *Sending* means frames are leaving. *No output* means it is live but something is missing — usually no dongle — and says which. *Off* means you stopped it.

⚠ Sharing a universe

If the Art-Net universe you choose already carries patch rows, a warning names the universe, the row count and the exact channels that will clash. It is a **warning, not a block** — putting haze on channel 500 of a pixel universe is a legitimate thing to do if you mean it. On Serial DMX there is only ever one universe, so the pixel output and the faders are merged for you rather than fighting.

Bump buttons — for foggers

A fader is the wrong control for a fog machine: left up, it empties the tank and trips the thermostat. Each channel therefore has a **BUMP** button.

- **Burst = Hold** — the channel is up only while the button is held.
- **Burst = 1–10s** — press and let go; it runs for that long and releases itself.
- **Bump level** sets what a bump goes to. It takes the *higher* of the fader and the bump level, so bumping never dims a channel.

A bump always releases: on the button, on losing the window, on leaving the page, on Blackout, on quit, and on a 30-second ceiling even if you sit on it. While a bump is up the **number and the track show the live level** while the thumb stays where the channel will return to.

Scenes

+ Save current stores every channel in the bank as a scene — all 32, not just the ones on screen, so a scene saved at 6 faders still holds what the hidden channels were set to. Scenes are stored in the project, so they travel in the `.lms` file — and opening a different show gives you that show's scenes.

Right-click a scene for: Recall, Update from the current faders, Rename, Duplicate, colour, reorder, delete — plus its behaviour:

- 🔒 **Latch** — recall it and it stays.
- ⚡ **Flash** — only while held. Give it a **Flash time** (1–10s) and it becomes press-and-let-go instead.

A scene button **rings green while its look is live**. That is worked out by comparing the scene to the faders, so nudging any fader clears it — the ring can never claim a look that is not on stage.

Chases

✎ **Edit chases…** opens the chase editor: pick a chase from the list, then set its steps — each one a saved scene with a **hold** and a **fade**, looping or one-shot. Chase buttons show transport on their face (▶ idle, ■ **2/4** running).

⚠ **Fades suit dimmer and RGB channels.** A crossfade walks a channel through every value in between, which is right for a colour wash and wrong for a gobo, mode or strobe channel on a moving head — those would sweep through every setting on the way. This is not built for movers.

A running chase **owns the channels**. Touch a fader, recall a scene or fire a flash and the chase stops — the same as taking a fader on a desk.

Changing the bank size keeps your levels. All 32 channels are stored in the project whatever you have on screen, so going 32 → 6 → 32 does not wipe the ones you could not see. Only the visible ones are sent, so shrinking the bank also stops the hidden channels dead — a hazer cannot be left running on a fader you have hidden.

Blackout

Blackout takes every channel to zero and stops any chase. Press it again — it now reads **Restore look** — and the previous look comes back, *including restarting the chase that was running*. The fader page and the Show Mode popup share one blackout state, so they can never disagree about whether the rig is dark.

In Show Mode

The side panel has a 🎪 **DMX Looks** button that opens a recall popup: your scene and chase buttons, nothing else — no faders and nothing that edits, because mid-show you press things rather than rename them. It does not block the trigger grid behind it, so a DMX look and a pixel scene can be fired in the same breath. It also carries **Blackout faders** and a shortcut to the fader page.

🎪 Link a DMX look to a Show Mode button

The point of this: **one press brings up the pixel scene and the conventional rig together** — your wall lights up and the uplighters, hazers and pars come with it, instead of you firing two things in two places. (Bob's idea.)

In **Show Mode**, right-click any Scene or Chase button and choose **Link a DMX look…**. Pick one of your saved DMX scenes or chases. That button now fires both, and is marked **DMX LINKED** in gold in its top-

right corner — a button that does two things should say so.

All four combinations work: a pixel **scene** or **chase** can fire a DMX **scene** or **chase**.

- **Latch buttons** leave the DMX look up when you press again, exactly as its own button in the popup would. The pixel chase stops; the DMX look stays until you change or blackout it.
- **Flash buttons** put the DMX levels back where they were the moment you let go — including stopping a DMX chase the flash started. A momentary press must not leave a hazer running.
- **Bindings come for free.** Whatever keyboard key, MIDI note or incoming DMX channel fires that button also fires its linked look, with no second binding to set up.
- **Unlink** from the same menu. The menu shows what a button is linked to, and says **MISSING** if you have since deleted that DMX look — a broken link never stops the pixel scene from firing.

Links are stored per button and travel in the `.lms`, and are deleted along with the scene or chase they belong to. They also fire from the **Tablet Remote** — a linked button behaves the same wherever it is pressed.

⚠ The link runs **one way only**: a Show Mode button fires a DMX look, never the reverse. Two-way links could trigger each other, and a link graph is not something to unpick mid-show.

⚠ Linking is done from the right-click menu, so it follows the same **Performance Mode** rule as the rest of that menu — with *Right-click on Scene + Chase buttons* masked, an end user cannot re-link your buttons. The links themselves keep working.

📱 On the Tablet Remote

The tablet shows a **DMX Looks** section under your scene and chase buttons — your saved DMX scenes and chases, recallable from the floor, plus its own **DMX BLACKOUT**. Whoever can raise haze from the floor can kill it from the floor.

⚠ These are **tap-to-fire only**, even for a look you have set to Flash on the desk. A held flash needs a guaranteed release, and a tablet can lose its network mid-press; a stuck pixel scene is untidy, a stuck DMX channel is a hazer emptying itself.

⚠ The tablet obeys the same **Performance Mode** mask as the desk. Hide *DMX Looks button* in **Settings → General** and it disappears from the tablet too — otherwise masking it on a locked panel would achieve nothing while a tablet sat on the floor with the same controls.

External control

Everything here can be driven from a keyboard, a MIDI controller or an incoming Art-Net/DMX channel, and appears in **Control → Control Bindings** alongside the rest of the show:

- **Scenes and chases** — right-click the button, or use the bindings table. Keyboard, MIDI and DMX.
- **Fader levels** — right-click a fader. MIDI CC or a DMX channel; no keyboard, because a key cannot express a level.
- **Bumps** — right-click a BUMP button. Keyboard works here, since a bump is momentary.
- **Blackout** — right-click the Blackout button.

A bound flash or bump honours press *and* release exactly like the mouse, including the timed variants. Moving a bound fader takes control back from a running chase — but a one-step wobble will not, so a noisy controller cannot stop a chase you never touched.

⚠ Worth knowing

- Fader output is **independent of OUTPUT STREAMING**. You bump a hazer precisely when the pixel show is not running, so the master switch does not gate it.
- Leaving the page does **not** stop the output — a static look survives going back to the Builder. The menu bar shows 🟡 **DMX FADERS LIVE** while it is sending, and that menu can blackout or stop from anywhere.
- Everything — address, labels, bump level, burst, scenes and chases — is saved in the project and travels in the `.lms`.

Performance & large matrices

LMS renders the whole pixel buffer through 8 effect slots, 4 channel blends, 2 group blends, 1 main blend, then master / strobe / gamma — every frame, in JavaScript. The number of bytes touched per frame scales linearly with matrix size.

Live FPS in the title bar

The window title now shows the live measured FPS:

```
Smartshow LED Matrix Server v1.0.0-beta.60d · my-show.lms · 58.7 fps
```

Updates once a second. If the number is well below your target (Settings → Render rate), the loop is starved — see below.

HDMI is the path for big walls

For large matrices (anything bigger than ~64×64), the design target is **HDMI to a Colorlight / Linsn / Novastar scaler**. The Output Patch / Art-Net / sACN / DDP / TPM2 / Serial DMX / Gledіator senders are for small-to-medium installs — at 384×192 = 73K pixels you'd be pushing ~430 Art-Net universes, which neither LMS nor the network can sustain at 30+ fps.

If you're on HDMI only, turn OFF **Output Streaming** in Settings (or via the 💡 button on the Builder). HDMI is unaffected by the master gate; the network protocols stop sending, you reclaim per-frame CPU. Tick "Remember streaming state across launches" so it stays off across restarts.

Strobe at low render FPS

The strobe is **frame-quantised** — pulses are exactly an integer number of rendered frames, which the eye reads as rock-steady. The trade-off is that the achievable rate is constrained by your live FPS: at 30 fps, 12 Hz is the closest clean rate above 10. If FPS dips during a strobe, pulses keep their integer-frame width which keeps them visually steady, but the rate tracks the loop. The internal math now uses your **measured** live FPS (not the configured target), so a 12 Hz request at 10 fps doesn't end up running 6× too slow.

What costs the most

- **Plasma / Fire / Video / large Text** — heaviest effects, scale with pixel count.
- **Output Patch with many universes** — each row is a slice + a network send per frame. Strip down to what you actually use.
- **Audio Input running while you don't need it** — turn off in Settings.
- **Heavy multi-channel mixes** — every empty channel (both fx slots set to None / Blackout) now skips its render + blend automatically.

Text placeholders + multi-line

The **Text** effect and the **Star Wars Crawl** effect both support placeholder substitutions and multi-line input. Type any of these into the Text field:

Token	Renders as
`$DAY`	day of month, 2 digits (e.g. `21`)
`$MONTH`	month number, 2 digits (e.g. `05`)
`$MONTHWORD`	month name (e.g. `May`)
`$YEAR`	full year (e.g. `2026`)
`$TIME`	current time `HH:MM`
`$COUNT`	countdown to the **Countdown target** param (ISO datetime). Format `H:MM:SS` under a day, `Nd HH:MM:SS` over.
`\n`	literal `\n` in the input box → newline at render

Example: `$DAY $MONTHWORD $YEAR\n$TIME` renders as two lines — date on top, time below.

Resolved every frame; the bitmap cache keys on resolved text, so `$TIME` only rebuilds the glyph bitmap when the minute ticks — not every frame.

> Text rendering uses **5× super-sampling**: glyphs are drawn at 5× resolution into a scratch canvas then high-quality downsampled to the matrix size. Edge LEDs light at partial brightness for smooth gradients instead of hard staircase pixels.
>
> That is the right answer on a tight-pitch panel and the wrong one on a coarse matrix, where the soft edge is spread over LEDs several centimetres apart and just reads as dirt. If your text won't *conform to the pixels*, turn on **Pixel-crisp** in the Text effect's **Font** section — every LED then lands fully lit or fully off, the way Jinx does it.

Effects

Each effect has an **Edit** dialog with its own controls. Effects open **pre-widened with their sections laid out side by side across the page** so everything's visible at a glance, and complex effects (Text especially) group their controls under collapsible headings. A ▦ **Spread / ☰ List** toggle in the dialog title bar switches between the multi-column spread and a single stacked column — your choice is remembered. **The ? beside it (1.2.5) opens this help at that effect's own entry**, rather than at the top of a long page — so if you cannot remember what a control does, or what the newline escape is (it is `\n`, not `/n`), it is one click away from the dialog you are already in. *(Mike) (Also fixed in 1.2.5: switching to List used to make the dialog's close ✕ disappear, and switching back did not bring it back.)* Opening a collapsed section automatically scrolls it into view so you never have to chase it with the scrollbar. The dialog is viewport-aware — on small laptop screens the title bar + Close button stay pinned at the top, the body scrolls if there's too many params to fit. Effect dropdown groups effects into categories (Basic colour / Patterns / **Strip / 1D** / Audio Reactive / Generative / Text / Media / Scripting).

Finding an effect (1.2.4). The picker opens with a **filter box** already focused — type *lar* and you're on Larson Scanner. ↑ ↓ move through the matches, **Enter** picks one, **Esc** closes. Category headers **stay put as you scroll** so you always know which section you're in, and **clicking a header collapses that section**; each shows a count of what's inside. Everything starts expanded and your collapse choices are remembered. Searching always looks inside collapsed sections too, so you can never search for something that's there and be told it isn't.

Grouped modal layout (beta.47): every effect's Edit dialog now organises its params into collapsible *sections* with an emoji header (🎛 Field · ⭐ Star look · 🎨 Colour · 🎵 Audio reactive, etc). Sections you rarely touch start collapsed, sections you care about start open. Open/closed state is remembered per (effect, section) in localStorage. Many params now hide entirely when they don't apply — pick "Fixed colour" mode and the Cycle Speed slider vanishes; turn Audio reactive off and the Audio Gain slider disappears. Small italic *help lines* beneath params explain non-obvious controls (what does "Trail decay" actually mean, what's the difference between "Leading edge fade" and "Trail decay"). The same pattern applies to every effect.

Every slider has a typeable value box (1.1.1). The number to the right of any slider is an input, not a readout — click it and type an exact value, or use its up/down arrows (or the ↑↓ keys) to step by that parameter's own increment. Sliders cover a wide range in a couple of hundred pixels — Text **Size** spans 4–256, roughly 1.3 units per pixel of mouse travel — so exact values were previously unreachable. The ↺ button still resets to the default. *(Double-clicking the number no longer resets — it selects the number so you can retype it.)*

R/G/B/W sliders are colour-coded, typeable, and take hex (1.1.1). Each channel's track runs from black at 0 to that channel at full, so the slider shows you what it does as well as which one it is. Every channel has its own 0–255 input with step arrows, and a **hex field** sits next to the Quick-set swatch for anyone working from a brand colour or a lighting plot. It accepts `#rrggbb`, the `#rgb` shorthand, and `#rrggbbww` — the eight-digit form also sets **White**, which the OS colour picker cannot express. Anything unparseable is ignored and the field snaps back, so a mistyped value never disturbs the colour.

Basic colour

- **Simple Color** — R / G / B sliders, **Autocolor** (cycles hue, Autocolor Sync locks multiple slots to the same phase), **Autocolor speed**.
  Advance on beat — 1.1.2. In Autocolor mode, step the hue on each kick instead of creeping round on a clock. Replaces the speed slider while it's on.
  - **Beat sensitivity** — how hard a kick has to hit to count. Around 0.5 it takes the hard downbeats and ignores the soft off-beats, which gives you a half-time colour change. Re-arms at 60% of the threshold, so a sustained loud passage steps once rather than strobing.
  - **Hue step per beat** — how far round the wheel each beat moves. 0.12 ≈ 43°, about eight beats to get back where you started; 0.5 alternates between two opposite colours.
  - **On beat:** **Snap** — hard cut, or **Fade** — slides into the new colour over the **Fade time**. The fade always takes the short way round the colour wheel, so it never smears through the colours in between. Set the fade longer than the gap between beats and it's still moving when the next lands, giving a continuous drift that still breathes with the track.

  Needs audio input. With none running it falls back to the timed cycle rather than sitting on one frozen colour.
- **Flash / Strobe** — colour block that strobes. R/G/B + **Time 1 (on, ms)** / **Time 2 (off, ms)** / **Fade** (edge ramp 0=hard, 1=smooth). Autocolor + Autocolor Sync. **Audio trigger** mode replaces the timer with beat-driven pulses (configurable gain).

- **Border / Frame** — coloured frame around the outer edge of the matrix, now with **8 styles**. **Brightness** (Colour group, 0–2×) dims or boosts the whole frame whatever style it's on — including **Rainbow chase** and **Random colour change**, which have no colour swatch to turn down by hand. Use it rather than the mix fader when the frame is layered over another look: with **Left / Right on top** the fader is what keeps the frame's colour intact, so the frame's own Brightness is the right place to knock it back.
  - **Solid colour** (default) — fixed colour. Pure-black auto-lifts to (10,10,10) for Over (key black) survival.
  - **Chase →** / **Chase ←** — coloured band sweeps around the perimeter clockwise / counter-clockwise. *Chase band width* slider controls how wide the band is.
  - **Rainbow chase** — full hue cycle wrapped around the perimeter, rotating. Band-width slider compresses the rainbow.
  - **Twinkle** — base brightness with random pixels sparkling brighter. Density slider 0.005–1.0 (very rare flickers → constant glow).
  - **Random colour change** — entire border swaps to a fresh saturated random hue every N seconds with a cross-fade between transitions.
  - **Pulse** — brightness sine-fades up and down. Depth slider controls how dim it goes between peaks.
  - **Two-colour alternate** — perimeter split into N segments alternating between two colours, rotating around the loop.
  - **Colour source (1.2.6) — rainbow or random on ANY moving style.** The eight styles above mixed up two separate things: how the frame *moves* (chase, twinkle, pulse) and where its colour *comes from* (a fixed swatch, the rainbow, or a random drift). That made "Rainbow chase" and "Random colour change" whole styles, so a *twinkling rainbow* or a *pulsing random-colour* frame was impossible. Pick **Chase → / Chase ← / Twinkle / Pulse** and a new **Colour source** dropdown appears: **Fixed colour** (default, unchanged), 🌈 **Rainbow**, or 🎲 **Random colour change**. The Style still controls the movement; the source only decides the colour it moves in. Rainbow adds a **Rainbow spread** slider (snapped to whole hue cycles so there is no seam at the top of the frame); Random reuses **Change every**. Default is Fixed, so nothing you have saved looks different. *(Lee)* The same dropdown is in the **Text** effect's Frame section as **Frame colour source**.

  Shared controls: **Thickness** 1–32 px (auto-clamped to half the shorter matrix axis), **Animation speed** 0.05× – 5× (multiplier on top of Channel Speed — independent per-frame tuning), **Colour** (most modes) and **Colour B** (alternate). Mode-conditional sliders only appear for the chosen style. Interior pixels stay black so the effect composites naturally with whatever's in the other slot (pair with Plasma in fx2 + *Over (key black)* for "plasma framed", or *Add* for "plasma + glowing edge"). Same 8 styles also available as the matrix-edge frame inside the Text effect.
- **Plasma** ⚡ *(WebGL)* — **nine** pattern types (Classic / Interfering circles / Spirals / Cell waves / Smooth / **Rotozoom / Frizzles / Distortion waves / Hiphotic** — the last four absorbed from standalone effects in 1.2.4, each with its own controls that appear only when that pattern is picked), seven palettes (Rainbow / Warm / Cool / Fire / Ocean / Purple / B&W). Zoom 0.005–1.0, Rotation speed, Brightness, Contrast.
- **Color Cycle** ⚡ *(WebGL)* — animated colour scroll. Seven styles (Horizontal / Vertical / two Diagonals / Circle / Diamond / Square), seven palettes, Zoom, Speed, Reverse.

- **Gradient Sweep** ⚡ *(WebGL)* — Colour A → Colour B (optionally → Colour C). Four wave shapes (Sine / Triangle / Sawtooth / Square — Square = striping), four direction styles (Linear / Radial / Diamond / Square ring), Angle, Repeat (band count), Brightness.

Text + crawl

- **Text** — scrolling or static text with full placeholder support (see above). **Edit dialog is grouped into collapsible sections — 1.1.6**: Content · Font · Colour & outline · Position & scroll · Motion FX · Flash / strobe · Frame, so the 40+ params don't read as a wall. Section open/closed state remembered per effect. **Controls that do not apply are now hidden — 1.1.6.** Every effect editor has parameters that only matter once something else is switched on: the Wave and Pulse settings need a Motion FX chosen, Strobe rate needs Flash on, Frame style needs a frame thickness. Those were meant to appear only when relevant and, for anything with a *slider* or a *tickbox*, never did — so every editor showed a pile of dead controls. Text alone dropped from 39 visible rows to 21. 50+ font families across Sans / Serif / Mono / Display / Script / Themed / Symbol categories. **Custom font library** — pick ★ **Custom (upload below)** from the Family dropdown to open the file picker (.ttf / .otf / .woff / .woff2). Uploaded fonts go into a project-wide library, appear under a **Custom** optgroup in every Text slot's Family dropdown of that project, and travel with .lms files. ⚙ **Manage custom fonts…** at the bottom of the Custom group opens a small dialog to delete fonts by name. De-duplication: re-uploading the same file returns the existing library entry rather than double-storing. Built-in **Flash / strobe** — Hz + Duty% + Fade. **Per-character outline** (colour + thickness 0–8 px). **Matrix-edge border / frame** now supports the full **8 animated styles** (see Border / Frame effect below) — pick Frame style: Solid · Chase → · Chase ← · Rainbow chase · Twinkle · Random colour change · Pulse · Two-colour alternate. Frame animation speed + chase band width + twinkle density / change rate / pulse depth / alternate segments all gated by the chosen style. Pure-black RGBW auto-lifts to (10,10,10) so a black outline / frame survives the Over (key black) mix mode threshold. **Motion FX (beta.47a)** — independent of Scroll: pick any scroll direction (or Static) AND layer on a motion effect: 〰 **Bob** (whole text bobs along a sine wave), 🌊 **Wave** (ripple travels through the letters), or 💗 **Pulse** (text zooms in/out rhythmically). Every combination works — scrolling text that bobs, static text that pulses, static text rippling in place, etc. Shared params: Wave amplitude · Wave length · Bob/wave rate (Hz). Pulse adds Pulse speed (Hz) + Pulse zoom (0=no zoom, 1=±50% scale). Legacy saves that used the older combined modes ('wave', 'wave-static', 'pulse') auto-migrate to the new Scroll + Motion FX pairs.

  Pixel-crisp (no anti-alias) — 1.1.1. Glyphs are super-sampled and smoothed by default, which looks right on a dense panel and looks like grey dirt on a low-density one. Tick **Pixel-crisp** in **Font** and every LED is fully lit or fully off, Jinx-style. **Crisp threshold** (5–95%, default 50) sets how much of an LED the glyph must cover before it lights — lower gives fatter letters, higher gives thinner. Works with everything else: scroll, Bob, Wave, Pulse and per-character outlines all stay hard-edged, and an outline keeps its own colour instead of bleeding into the fill. Default **off**, so nothing you have saved changes.

  Loop style — 1.1.1. By default a scroll travels the full width of the matrix *plus* the full width of the text before it comes back, so there is always a stretch where the wall is completely blank — and the longer the message, the longer the blank. Two alternatives in **Position & scroll**:

- ♾ **Seamless** — the next copy follows straight on behind the last, so the matrix is never empty. Spacing at the join is the font's own, exactly as if you had typed the message twice in a row; **Gap / pause** adds extra space on top of that, and 0 gives one unbroken ribbon.
  ⚠ **Fixed in 1.1.11: Gap / pause** was being applied once per *batch* of copies rather than once per copy, so most joins had no space at all and the message ran together as one long string. It depended on font size in a way that made no sense — the same text was spaced correctly at one size and butted at the next — because the number of copies LMS prepares at a time varies with how the font's width falls. Every join is now spaced.
- ⇄ **Bounce** — the text reverses at each end, so nothing ever leaves the screen. **Gap / pause** becomes a dwell at each end.
  1.1.11. Text that *overhangs* the matrix travels its overhang, bringing each end into view in turn. Text that *fits* now travels the empty space instead, sliding edge to edge — previously it centred and held perfectly still, which made Bounce look like it did nothing at all. For stationary text use **Scroll = static**.
  ⚠ Also fixed in 1.1.11: the dwell was landing entirely at one end (twice the length you asked for) and not at all at the other, so the text paused going one way and turned instantly the other. Both ends now hold for the same time, and the overall cycle length is unchanged.

Both work on vertical scrolls and combine with Motion FX.

Smooth (sub-pixel) motion — 1.1.1. Scroll position used to be rounded to whole pixels, so unless the speed happened to be an exact number of pixels per frame the text stalled on some frames — at the **default** speed of 24 on a 30fps output that is 0.8 px/frame, i.e. frozen one frame in five. Tick this and the text moves in fractions of a pixel, part-lighting the LEDs at its edges, which the eye reads as continuous motion. It is how scrolling LED signs have always done it. Bob, Wave and Pulse get the same treatment. Not offered alongside **Pixel-crisp**, which exists to keep every LED fully on or off — the two want opposite things. Default off.

Auto colour — 1.1.1. Drives the fill from the hue wheel instead of the Colour picker. 🌈 **Cycle** shifts the whole text through the spectrum together; 🌈 **Spread** lays a rainbow across the text that travels with the letters; 🌈 **Per letter** gives every character its own hue. **Colour speed** is hue rotations per second (0 holds the colours still), **Colour spread** is how many full rainbows fit across the text, and **Colour saturation** takes it from full colour down to greyscale. A per-character **outline keeps its own colour** throughout, so the letters stay legible against a moving fill. Default **off**.

- **Star Wars Crawl** — perspective-receding text crawl. Lines emerge at the horizon (close, big), recede to the vanishing point (small, dim). Supports the same placeholder system + `\n` multi-line. Configurable horizon position, near/far scale, top-fade strength, crawl speed.

Animated patterns

- **Fire** — two render styles selectable via **Style** dropdown: *Smooth* (LMS default — weighted-blur sample from below, polished, less grain) or *Granular* (classic single-cell sample with ±1 random horizontal jitter — pixel-tearing "Doom fire" propagation that reads as more alive on small matrices). Intensity, Flame height (0.2–5×, extends past top for cropped-tall-fire look), Cooling, Wind, Turbulence, Spark rate, Big-spark mix, six palettes (classic / orange / blue / green / purple / rainbow), Brightness, Audio reactive. Heat buffer is now keyed per-size so a full-canvas Fire slot + a region-confined Fire slot can run side-by-side without thrashing each other's state.
- **Kaleidoscope** ⚡ *(WebGL)* — five patterns (Sines / Concentric rings / Checkerboard / Radial spokes / Marble noise), 2–16 mirror arms, Zoom, Pattern speed, Rotation speed, Twist (spiral), Centre X/Y offset, Inner/Outer radius mask, Fit (inscribed vs fill), Saturation, Brightness.
- **Sparkle** — five colour modes (Fixed / Alternate 2 / Random hue per spark / Cycling rainbow / Tinted by audio band), three spark sizes (1 / 3 / 5 px), Density, Decay, Brightness, Audio Trigger.
- **Conway's Life** — 11 cellular-automaton rules. Colour modes: Solid / Age-based / Neighbour count / Birth highlight. Wrap edges toggle, Random sparks, Audio reactive (beats inject cells), Reseed strategy. Dead-cell colour defaults to true black so the background doesn't glow — pick any other colour from the Dead colour swatch if you want a tint.
- **Sine Lines** — 1–3 sine waves, each tied to a frequency band with its own RGBW. **Audio Trigger** drives amplitude from per-band level — even during silence, lines stay visible at ~15% of the static amplitude so you can see at-a-glance that the effect's running.
- **Starfield** — 3D-perspective stars zooming toward the camera. Stars / Zoom / Speed. **Focus X / Focus Y** shifts the vanishing point off-centre (−1 = left/top edge, +1 = right/bottom edge) so the warp banks like a turning starship instead of going straight ahead. Six colour modes (Fixed / Random / Depth gradient / Two-colour / Cycling rainbow / Realistic with rare giants), star size slider, twinkle.
- **Metaballs** — 1–12 bouncing soft blobs merging where they overlap. Optional Collision Detection (elastic pairwise bounces).
- **Fractal Generator** ⚡ *(WebGL)* — three iteration types: *Julia* (z^2 + c with animated `c` — the morphing-fractal workhorse), *Mandelbrot* (classic seahorse-valley view), *Newton* (z^3 - 1, coloured by which root each pixel converges to — the tri-coloured trippy look). Zoom, Centre X/Y, Auto-pan drift, C real / C imag (Julia parameter), Auto-orbit C (animates `c` around a circular orbit so Julia continuously reshapes), Iterations 8–96 (detail vs perf), four palettes (Sweep / Fire / Cool / Mono), Palette speed (colour-cycling), Smooth iteration (sub-pixel banding vs blocky), Audio reactive (None / Bass → zoom / Bass → palette flash / Bass → C orbit). Single-precision shader caps usable zoom at ~10^4× — fine for typical LED resolutions; at 1080p HDMI output deep zooms will band.
- **Expanding Shapes** — shapes spawn at random and grow outward. 10 shape options + Mixed mode, Filled / Outline, 6 origin presets, Rotation, four fade modes, Edge softness, 4 colour modes.
- **Falling Rain** — 4 directions, 1–400 drops, three thicknesses, three trail styles, 5 colour modes (including **Matrix** green), Wind drift, Splash on impact, Audio reactive.

- **Radar / Scan Lines** — 8 directions including ping-pong and Radar (CW / CCW). Width, separate **Leading edge fade** + **Trailing edge fade** sliders (1.0/1.0 = symmetric classic look, drop Leading toward 0 for a clean front edge with a phosphor-style soft trail), **Trail decay** dt-normalised so trails persist for the same wall-clock time at any render rate, motion-blur sub-stepping at low FPS to kill the line-ratchet on large matrices, Offset, and (Radar modes) **Center X / Center Y** sliders to move the sweep pivot off-centre — set both to 0 or 1 to put the radar origin in a corner for a quarter-sweep. Full RGBW + Autocolor.
- **Snow** ❄️ (beta.47) — falling snowflakes with depth-based dimming (slow = far + dim, fast = near + bright), wind drift + per-flake gust jitter for natural fluttering, optional **Accumulation** mode that piles flakes into a drift at the bottom which slowly melts down. Audio reactive turns bass into snowstorm flurries.
- **Matrix Rain** 💚 (beta.47) — the classic green digital-rain code. Per-column streams with a bright head pixel + green fading trail. **Glyph flicker** randomly darkens mid-trail pixels each frame to fake character cycling at LED resolution (where actual katakana wouldn't be readable). White-ish head colour by default for the proper Matrix look.
- **Lightning Storm** ⚡ (beta.47) — jagged bolts via recursive Bresenham with midpoint perturbation. Configurable **Branch depth** (each bolt splits into side forks), Origin (top / bottom / sides / anywhere), **brief whole-matrix atmospheric flash** that decays faster than the bolt linger so you get a proper "flash, then bolt fade" sequence. Audio mode replaces the timer with beat-driven strikes. Pairs naturally with Rain/Snow underneath.

Celebrations & atmospheres (beta.47)

Effects pack aimed at weddings, parties, ceremonies and celebration moments. All share the same grouped-modal layout pattern and four/five palette options each.

- **Falling Petals** 🌸 — particles that *tumble* as they fall. Each petal has its own spin rate; `abs(cos(spinPhase))` collapses petal width as they rotate edge-on → face-on, so they read as rotating instead of just drifting. Wind drift + per-petal sway. Five palettes: Rose · Sakura · Autumn · Lavender · Mono.
- **Floating Hearts** 💖 — heart sprites rise from the bottom with gentle sway, fade in/out at edges instead of popping. Procedural heart mask cached per size. **Beat-driven spawn** mode recycles the topmost heart as a fresh burst on each detected kick. Palettes: Romance · Pink · Pride rainbow · Mono.
- **Confetti Burst** 🎊 — three spawn modes: *Steady stream*, *Periodic burst*, or *Beat-triggered burst*. Physics with gravity + air drag (paper, not rocks) + per-piece spin (streamers shimmer as they flip edge-on). Origin: top / corners / centre (cannon). End-of-life fade-out. Mixed-shape mode blends squares + streamers. Palettes: Party · Gold/silver · Pastel · Pride · Mono. Capped at 400 active particles.
- **Bokeh Drift** ✨ — slow drifting Gaussian-falloff additive light blobs. Each bokeh wanders smoothly toward a random target then picks a fresh target on arrival; per-bokeh twinkle phase gives subtle breathing. Bokeh-major iteration (only touches pixels inside each light's bounding box) keeps it fast even with 50 lights on a big wall. Palettes: Warm · Rose · Icy · Magic · Mono. Romantic backdrop that photographs / videos beautifully.

More patterns (1.2.2–1.2.4)

- **Vortex** 🌀 — arms spiralling out of a movable centre. Arms, Density, Swirl, Centre X/Y; speed goes negative to reverse. Palette or a single Mono colour. Beat pumps the arms outward. *(Called "Octopus / Vortex" before 1.2.4.)*
- **Octopus** ⚡ — a plasma ball: writhing tendrils from a glowing core. Tendrils, Sharpness (tight filaments vs soft glow), Core glow. Classic single-colour look, or a palette with Hue spread along each tendril and Hue drift over time. *(Called "Plasma Ball" before 1.2.4.)*
- **Aurora** 🌌 — polar-light curtains rippling across the top of the matrix. Curtains (how many), Curtain height, Speed. Volume brightens the shimmer.
- **Pacifica** 🌊 — layered ocean swell. Speed and Scale set the water; **Depth** is the contrast between deep water and crests. Classic teal→blue, or any palette; Base hue and Hue spread reshape the classic look without leaving it. Bass swells the swell.
- **Soap** 🫧 — a drifting oil-film. Two separate smears: **Smear (time)** is persistence — how slowly each pixel chases its target — while **Smear (spatial)** bleeds colour sideways into its neighbours, which is the one that actually drags the film around. Spread radius sets how far the bleed reaches.
- **Sun Radiation** 🌞 — shimmering rays from a movable centre. Rays, Shimmer speed, Centre X/Y. Audio can drive Ray reach, Brightness, or both.
- **Drift Rose** 🌹 — a rotating rose curve (r = cos kθ). Petals, Thickness, **Speed / Rotation** and **Size**. Audio can drive Thickness, Brightness or both, from Bass / Mids / Treble / Full mix.
  1.2.5. The speed slider was called just "Speed", which hid the fact that its *sign* is what sets the direction: **negative spins anticlockwise, positive clockwise**, and 0 holds the rose still. Renamed to **Speed / Rotation** and given a help line. New **Size** control (0.2–1.8, default 1) scales the rose about the centre — it scales the petal radius and its stroke together, so a small rose is a true miniature rather than a fat blob; above 1 the petal tips run off the edge of the matrix. Default 1 renders exactly as before, so saved shows are untouched. *(Mike)*
- **Sunrise / Sunset** 🌅 — a sky gradient with the sun rising and setting through it. Loop, Rise once & hold, or Static with a Sun height slider. Sky and Sun colours, plus an optional **Separate glow colour** so the bloom passes through a distinct band (the red/pink you get in a real sunset), with Glow spread and Horizon warmth.
- **Tartan** 🧵 — woven warp-and-weft check in three colours. Weave size and a Drift speed that slides the pattern.

More generative (1.2.2–1.2.4)

- **Star Cloth** ✨ — the classic starcloth drape. Density, Twinkle speed, Flicker, Star size, two star colours with a Colour mix (or a random hue per star), over your own background.
- **Fireworks** 🎆 — proper shells: a rocket climbs with a trail, bursts at altitude, and the particles fall under gravity with drag. Launch rate, Particles per burst, Burst spread, Gravity, Particle life, and an optional brief flash on the burst.
- **Starburst** 💥 — radial ray-bursts spawning across the matrix. Spawn rate, Rays per burst, Speed, Trail. *(Called "Colored Bursts" before 1.2.4 — renamed because Fireworks above is the actual firework.)*
- **Meteor / Comets** ☄️ — meteors crossing the matrix with fading tails. Eight directions including all four diagonals, plus Random angles. Spawn rate, Speed, Tail length; beats launch extra meteors.

- **Bouncing Balls** ⚪ — balls falling and bouncing under gravity. Balls, Gravity, Ball size.
- **Black Hole** 🕳️ — particles spiralling inward and re-seeding at the rim. Particles, Spin, Pull.
- **Sindots** · — dots tracing interfering sine paths, with a persistence trail. Dots, Speed, Wave frequency, Trail.
- **Tetrix** 🧱 — blocks dropping and stacking, clearing when the field fills. **Block size (px)** defaults to *Auto*, which sizes a Tetris cell to suit your matrix — leave it there on a dense wall or the blocks are single-pixel confetti. Block width is measured in cells, and Grid gap adds the classic dark border inside each one.
  Playfield fit — 1.2.5. The playfield is a whole number of cells, so unless the block size divides your matrix height exactly there are leftover pixels. They used to be split top *and* bottom, which left a dark strip **under** the settled stack — the blocks looked like they were floating, and the strip grew with the block size (up to one cell less than the block, so a 9px block on a 32px matrix left 3 dark rows at the bottom). The floor now sits hard on the **bottom edge** and the leftover goes to the top, where the pieces drop in from and nothing ever rests. Pick **Centred** if you preferred the old symmetrical framing. *(Mike)*

Strip / 1D (1.2.4)

Effects whose structure is a **line** rather than a field — the classic LED-strip looks. They are built for a single run of LEDs (set the matrix to something like **150 × 1**), but they are **not** limited to one: every Strip / 1D effect carries a **Strip mapping** control that decides how the line is laid onto the matrix, so they work just as well on a normal wall.

- **Fit width (across)** — the line runs left-to-right and is copied down every row. On a real strip this is the natural choice; on a matrix it gives vertical curtains.
- **Snake (serpentine)** — the line follows serpentine wiring through the whole matrix, so a scanner snakes back and forth across the wall. This is the look most people know from WLED.
- **Rows** / **Columns** — straight linear scan, no zig-zag.
- **Radial (centre out)** — the line is wrapped around the centre, so it blooms outward.

The mapping is deliberately **independent of the Output Patch**. The patch can hold several rows with different wiring orders, so there is no single "chain" for an effect to follow — giving the effect its own traversal keeps the two from fighting, and lets you snake an effect through a wall that isn't wired that way at all.

- **Larson Scanner** 🔴 — the Cylon/Knight Rider eye. 1–4 eyes evenly spaced, eye size, tail length, and Bounce (sweeps back and forth) or wrap-around. Fixed colour, cycling rainbow, or a new hue each sweep. Audio drives the scan speed.
- **Theater Chase** 🎭 — marquee chase. Spacing between lit groups (3 = the classic), group size, speed in steps/sec, reverse. Fixed / rainbow-per-group / cycling.
- **Pride** 🌈 — the FastLED classic. Flowing rainbow whose hue rate, brightness depth and wave speed all drift independently, so it never quite repeats. Hue density and brightness depth exposed.
- **Colorwaves** 🌊 — Pride's softer, palette-driven sibling; the hue creeps rather than marches.
- **Running Lights** 〰️ — sine humps travelling along the strip. Wave count, sharpness (tight pulses vs gentle swell), bidirectional speed.

- **Colour Wipe** 🖌️ — fills the strip a colour at a time. Alternate A/B, a new hue each pass, or random; optional back-and-forth so alternate passes wipe the other way.
- **Sinelon** 🔴 — a single dot swinging on a sine with a fading trail. Hue can follow the dot's position.
- **Juggle** 🤹 — several dots on harmonic rates weaving through each other and periodically re-aligning. Rate spread controls how quickly they diverge.
- **BPM** 🐮 — pulsing colour waves at a set BPM, or driven by the live beat when Audio reactive is on. Stripe density and pulse depth.
- **Twinklefox** ✨ — independent twinkles fading in and out. Density sets what fraction of pixels ever light; rate variance sets how much they differ from each other.
- **Drip** 💧 — droplets falling under gravity with a splash bounce at the end.
- **Popcorn** 🍿 — kernels launched up the strip and falling back. Pop force, gravity, rate; beats pop extra kernels.
- **DJ Light** 🎛️ — bass / mid / treble colours growing out from the centre of the strip. Smoothing softens the jump between frames so it pumps rather than flickers.

> **Strip matrices and the preview.** An N × 1 matrix is a perfectly legal shape and the previews show it at its true aspect — a single line. On the Builder **MAIN OUTPUT** monitor, individual LEDs are drawn as separate dots (with the round/square and gap settings from **Monitor Settings**) only while each pixel is at least 5 screen pixels wide. That panel is roughly 600 px across, so past about **120 pixels wide** the dots merge into a continuous line — the effect is unchanged, it's purely how much room each LED has on screen.

Audio-reactive

- **Audio · Waterfall** 🌊 — a scrolling spectrogram: each new frame writes one row of the spectrum and the picture scrolls Up or Down, so you see the last few seconds of the music. Heat / Spectrum / Mono palettes.
- **Audio · Gravity VU** 📊 — a VU meter whose peak markers fall back under gravity rather than snapping down. Direction, Base colour and Peak colour.
- **Audio · Waverly** 〰️ — noise-driven waves whose scale and motion follow the music. Scale and Speed set the field; gain sets how hard the audio pushes it.
- **Audio · Freqmatrix** 🎛️ — frequency mapped to colour, scrolling across the matrix, so the picture is a moving record of which bands are loud.
- **Audio · Spectrum Bars** — FFT bars in 5 orientations including Center-out V and Center-out H (peak-hold dots paint on BOTH sides of centre at the bar tips). 4 colour modes, peak-hold falloff, gain.
- **Audio · Beat Pulse** — 6 pulse shapes (solid / radial / vertical column / horizontal bar / stripes / pixel storm), 5 colour modes (fixed / cycle / random / audio-band / energy), sensitivity, ambient floor, smooth decay envelope. **Beat hold-off (s)** slider (default 0.15 s) gates re-triggers within the window — kills the double-pulse you get from a single transient ringing through a few FFT frames.
- **Audio · Beat Rings** — expanding rings on beats. 6 colour modes including festival-palette burst, ring vs filled mode, matrix-relative speed/width, centre offset.

- **Audio · VU Meter** — three-zone bar (Base / Mid / Peak colours) with configurable transition positions and smooth-fade vs hard-banding. Direction (Left / Right / Top / Bottom / Centre H / Centre V — centre modes grow OUTWARD from the matrix centre with symmetric peak dots). Mono / Stereo (real ChannelSplitter L/R), Peak Hold, Auto Gain Control. **Bar attack** and **Bar release** sliders for one-pole filtered response (defaults ~10 ms attack / 200 ms release — snappier or smoother as you like).
- **Audio · Oscilloscope** 〰 (beta.47) — time-domain waveform display, completing the audio family. Four display modes: *Line* (classic scope), *Filled to centre*, *Mirrored top + bottom* (DJ-scope), *Lissajous / X-Y* (stereo — plots L vs R as a 2D loop, classic stereo-phase view). Independent **Input gain** (0.1–10) and **Amplitude** (vertical scale, 0.1–2.0). **Auto-level (AGC)** envelope follower with fast attack / slow release keeps peaks near 85% of matrix height regardless of input level. **Time zoom** (0.1–4.0) — 1.0 shows the whole 2048-sample window, higher zooms in on a shorter slice for finer detail. Three triggers: None / Zero-crossing (stable display) / Peak (lock to brightest sample). **Phosphor persistence** — exponential decay buffer between frames with its own dim trail colour for proper CRT-scope afterglow. Vertical offset, line thickness, glow halo.

Inputs

- **Image / GIF** — file picker, GIFs animate properly (decoded frame-by-frame via inline GIF parser — bypasses Chromium's GIF-throttling). Fit modes, Brightness, Offsets, Scroll X/Y per second, Invert. Built-in **Flash / strobe** (Hz / Duty % / Fade) — defaults off; flick on to make a logo / GIF blink without needing a separate Strobe slot.
- **Video** — file picker. Plays MP4 (H.264/H.265), WebM (VP8/VP9), MOV directly. **AVI / MKV / WMV / FLV / DivX / Xvid and other legacy formats are auto-converted on import** — LMS checks whether the file plays and, if not, transcodes it to an MP4 with the built-in converter (you'll see a "converting… %" readout), then uses that. The conversion is cached, so re-picking the same file is instant. Great for bringing old Jinx! video clips straight across. Stored as a `file://` path so big videos don't bloat the project save. Loop / Mute / Speed / Fit / Offsets.
- **Webcam / Capture Card** — picks a camera device. Also the way in for a **USB capture card** (HDMI→USB), so a games console, media player or camera feed maps straight onto the LEDs. **Requires Windows Privacy → Camera → "Let desktop apps access your camera" to be ON**.
- **Screen Capture Input** — pick a screen or application window via the dropdown (thumbnails shown). Crop X/Y/W/H, optional Mirror horizontally.

- **NDI Input** *(Windows)* — receive a live **NDI®** video source (e.g. **OBS Studio** via the DistroAV plugin) straight onto the LEDs. Pick the source from the dropdown (⟳ Refresh to rescan), Fit / Mirror, Brightness + the 🎨 Image adjustments. Defaults to the source's low-bandwidth **proxy** stream (lighter, ample for LEDs) — tick **Full quality** for full resolution. Works same-PC (auto shared-memory fast path) or from another machine on the LAN. **Requires the free NDI Tools / Runtime (https://ndi.video/tools/) to be installed**; if it isn't, the source list says so.
  PTZ camera control. When the source is a PTZ-capable NDI camera (e.g. QSC NC-series, PTZOptics), a **Camera control (PTZ)** panel appears in the effect: **pan / tilt / zoom**, plus **store** and **recall preset** on the camera itself. A per-scene **Recall preset** means a scene doubles as a camera shot — recalling that scene drives the camera to the chosen preset. Sources that don't advertise PTZ simply don't show the panel. *(⚠ Playing a file into LMS over NDI — e.g. from VLC — is NOT a PTZ source and can't test this; you need a real PTZ camera.)*

🎨 **Image — Gamma / Contrast / Saturation** (right next to Brightness) on **Image/GIF**, **Video**, **Screen Capture** and **Webcam / Capture Card**: post-process adjustments so sRGB footage stops looking washed-out on LEDs. **Gamma** above 1 is the fix for dark shades reading too bright — the same trick Jinx! had in the AVI player. All neutral (= off) by default. (Webcam already carries its own inline Contrast, so it exposes Gamma + Saturation.)

Effects marked ⚡ are **WebGL-accelerated** — they compile a fragment shader and run on the GPU. CPU fallback kicks in automatically if WebGL is unavailable. At small matrix sizes the difference is invisible; at 256×128+ they're 10–100× faster.

Audio-reactive effects

Lots of effects can pulse and move to live sound. Turn on an input in **Settings** → **Audio** (a microphone, or system-audio loopback via a virtual cable like VB-Cable / BlackHole), then:

- **Dedicated audio effects** — Spectrum Bars, Beat Pulse, Beat Rings, VU Meter, Sine Lines — are driven straight from the FFT (bass / mid / treble bands + a beat detector).
- **Beat brightness** — Plasma, Color Cycle and several others have a **Beat brightness** slider that pulses the whole effect on the beat. 0 = off.
- **LMS Script** reads audio via `audio_level`, `audio_bass`, `audio_mid`, `audio_treble`, `audio_beat` and `audio_trigger` (0–255).
- **Custom Shaders** receive `u_level`, `u_bass`, `u_mid`, `u_treble`, `u_beat` and `u_onset` (0–1), with an **Audio amount** control to scale reactivity. `u_onset` is a short kick pulse (snaps back fast) — use it for tight strobes; `u_beat` is the softer envelope.

Monitor & shaping (Settings → Audio): a live 0–1 read-out of every audio value sits under the input controls, so you can see at a glance whether a value is stuck high. The **Input level (peak)** meter shows the true incoming level, and controls above/below it shape the signal:

- **Input trim** — level applied *before* analysis. If your source is too hot the bass/beat values sit pinned near the top and effects stop punching on the beat — the meter **flashes red** and an ⚠ **"Input too hot"** warning appears. Pull the trim down (or lower your mixer / booth output) until the meter only peaks near the top on the loudest hits. This is a real fix, not cosmetic: the trim is pre-analysis, so lowering it stops the bands saturating.
- **Attack / Release** — how fast levels rise / fall (lower = smoother motion).
- **Smoothing** — extra jitter filter on the raw bands.
- **Beat sensitivity** — how readily the beat / onset pulses fire (raise for quiet kicks).
- **Noise gate** — silences everything below a threshold; raise it to kill room-noise / mic-hum twitch.

Start / Stop show their state — 1.1.7. The two buttons used to look identical whether capture was running or not. Now only the one that does something is enabled: while it is running, **Start** reads ● **Running** on green with a pulsing dot and **Stop** is the live one; stopped, it is the other way round.

Two buttons sit under the sliders alongside those: 💡 **What do these do?** pops the same explanation up inside the app, so you do not have to come here mid-set, and ↺ **Defaults** puts the trim and all five sliders back to neutral (and switches Auto gain off, so the trim is yours again).

Starting points — 1.1.7. If none of the above means anything to you, ignore all of it and press a button. The **Starting point** row at the top of **Audio envelope** sets the trim and all five sliders in one go, and they are named for the situation you are in rather than for what they do:

- **Natural** — the raw reaction, nothing shaped. If in doubt, start here and change one thing at a time.
- **Punchy** — snaps on the kick and drops away fast. Chases, strobes, anything that should hit ON the beat.

- **On the beat** — the hard-gated one. Everything below the gate reads as **black** and the hit is stretched to full range, so the wall goes *dark* between kicks rather than merely dipping — that is what makes it read as on-the-beat rather than just snappy. Beat sensitivity is raised so quiet kicks still fire (the onset threshold drops from 0.05 to 0.017). **It needs a decent input level to bite:** on a weak or distant source nothing clears the gate and you get a dead wall, so pair it with **Auto gain** or lift the trim.
- **Smooth** — slow rise and fall. Flowing, lava-lamp motion that follows the music without twitching.
- **Quiet room / mic** — a laptop or room mic picking up speakers across a room. Lifts the level and gates out the hum and chatter that would otherwise twitch the effects between tracks.
- **Loud club / line-in** — a hot, constant feed off a mixer or the system output. Trims it back so it stops saturating, and calms the beat trigger down: when everything is loud, everything looks like a kick.

Save your own — 1.1.7. Dial the trim and the five sliders in however you like, then ➕ **Save preset…** and give it a name. Saved presets live in the **— my presets —** dropdown beside that button rather than as chips of their own, so the row stays the same six starting points however many you build up over the years. Pick one and a single extra chip called **Custom** appears on the row as the selected one, with the name shown underneath and an × to delete it (which asks first — there is no undo). Choose any built-in again and the Custom chip goes away. ⚙ **Manage…** opens a small window listing everything you have saved, with its settings spelled out on each row so you can tell two similar ones apart without loading each in turn. Three actions per preset: **Update** overwrites it with whatever the sliders hold right now — the reason the window is worth having, because you tune a setup live at a gig and want the saved one to catch up — **Rename**, and **Delete**. Both destructive ones ask first. Your presets are saved **both** ways — to this installation *and* into the project. The install copy is why a setup you built once is there in every show you open and survives **File › New**: it describes your input chain, this booth, this mixer, this mic, which does not change when the show does. The copy in the `.lms` is why handing someone your show hands them the audio setups that go with it — opening it absorbs anything new into their own library. Deleting one is remembered, so a preset you have removed does not come back the next time you open a project that still carries it. Built-in names are reserved, so you cannot end up with two chips called Punchy doing different things.

A preset is a *starting point*, not a mode — every slider stays draggable, and the chip stops being highlighted the moment you move one, so it can never claim a state you have since left.

Auto gain — 1.1.7. Auto gain (ride the trim for me) under the Input trim watches the peak level and adjusts the trim for you, aiming to sit around **62%** — loud enough to drive the effects, with enough headroom to stay clear of the saturation that pins the bass and beat values. It comes **down fast** on a sudden loud passage and **creeps up slowly** on a quiet one, like any limiter. Crucially it **holds still during silence**: the classic auto-gain failure is winding the gain to maximum in the gap between tracks and then blasting room noise into your effects, and this will not do that. While it is on, the trim slider is disabled and shows you the value it has chosen — picking any Starting point switches it back off, because a preset sets a trim of its own.

> These are **global "front of chain"** controls — the trim, envelope and beat-sensitivity feed the one shared audio analysis, so they affect *every* audio-reactive effect and custom shader at once. (An individual effect's own "Audio amount" slider then scales just that effect on top.)

No audio input selected = silence (all bands read 0), so an audio effect looks static until you pick an input in Settings → Audio. And don't drive *motion* with audio — keep the clock steady and let audio drive brightness / contrast / a strobe, or it lurches on every beat.

LMS Script (the LMS Script effect)

LMS Script lets you write your own live matrix animations in a small, friendly scripting language — instead of picking a built-in effect, you drop the **LMS Script** effect into a channel slot and give it a script. It's the **same language and file format as classic *Jinx!* scripts** (`.jns`), so the huge library of community scripts people have shared over the years runs straight in LMS with no conversion.

Adding a script to a slot

In the Builder, set a channel effect to **LMS Script**, then use the **Script** dropdown on that slot. At the bottom of the list is a **My scripts** section with these actions:

Action	What it does
`+ New script…`	Open the editor on a blank script (a starter template) and write one from scratch.
`+ Load .jns file…`	Pick any `.jns` file from your computer. It's added to your library, appears in the list, and starts playing.
`⬇ Export current script…`	Save the selected script out to a `.jns` file — the counterpart to Load. Works on the **built-in presets** too, so you can pull Swirl or Particles out to a file and tweak or share it.
`✎ Edit current script…`	Open the current slot's script in the live editor.
`⚙ Manage my scripts…`	Rename, edit, **export** or delete the scripts you've saved — each row has its own Export button.

Panel controls a script can opt into. A script that declares `config r1/g1/b1` (and optionally `r2/g2/b2`) gets **Colour A** / **Colour B** pickers on the slot, so you can recolour it without opening the editor. **1.1.7:** one that declares `config size` also gets a **Block size** slider — that is what drives the squares in **Chequerboard** and **Bouncing Squares**. Leave it at **0** and the script keeps whatever size is written in its own source; anything above 0 overrides it. Scripts that do not use these names show neither control and behave exactly as before.

Every script you add or write is **saved inside the project** (the `.lms` file), so it travels with it — share a project and your scripts go along. The slot's global **Speed**, **Auto-colour speed** and **Brightness** controls sit on top of whatever the script draws, and a script slot mixes, layers, regions and outputs exactly like any other effect.

The live editor

The editor gives you a **live preview** of the script running at your real matrix size, and **checks the script as you type** — the status pane on the right shows either **✓ No errors** or the exact `Line N: …` of any problem.

- **Line numbers** run down the left of the code, and **the line of any error is highlighted red** so you can jump straight to it.

- 📖 **Script writing guide** (blue button under the code) opens the full **LMS Script writing guide** — the complete language reference with worked examples — *without* leaving the editor.
- ⬇ **Export .jns** (teal button, bottom-left) saves the script to a `.jns` file you can back up, send to someone, or post on the forum. It exports **what's in the editor right now**, not the last saved version — so you can try something, export it, and close without committing it to the project. The editor stays open.
- ⛶ **Expand** (title bar) expands the editor to near-fullscreen for long scripts and wide lines; click again to restore. You can also drag the dialog by its title bar to reposition it.
- **Tab** inserts a tab; edits mutate the running effect immediately so you see changes live in the preview and on the wall.

Full language reference: the complete guide — structure (`:init` / `:render`), variables & arrays, loops & conditions, every drawing command (`pset`, `line`, `rect`, `circle`, `text`, `fade`, `hsv2rgb`…), built-in values and worked examples — is the 📖 **Guide** button in the script editor (also opens here: LMS Script writing guide ↗).

If an imported script "does nothing": it's usually the script, not LMS — some `.jns` files are near-invisible test patterns or need a bigger matrix. Open it with **Edit script** to see the live preview and any errors, tweak it, and save your own version. A built-in preset always opens as an editable copy, so you can't break the originals.

Custom Shader (the Custom Shader effect)

Import or write your own GPU shader and run it as a normal effect. It lives under **Other** (next to LMS Script) and inherits scenes, chases, regions, foreground/background layering and Art-Net output like any effect.

- **Three dialects** accepted, adapted automatically: plain GLSL (write `gl_FragColor` in `main()`), **Shadertoy** (`mainImage()` with `iTime` / `iResolution`), and **ISF** (a JSON header that **auto-generates the controls**). Book of Shaders / glslCanvas shaders import too.
- **Load .frag** or paste into the editor; **Export .frag** to save one out. Everything is stored in the project.
- **Uniforms** supplied: `u_time`, `u_res`, the full audio block (incl. `u_onset`, a short kick pulse), plus user sliders/colours — and for ISF, the shader's own named inputs.
- An **Audio amount** control plus post-process **Brightness / Gamma / Contrast / Saturation** (under 🎨 Output) apply to any shader — **Gamma** above 1 is the fix for dark shades looking too bright on LEDs. All neutral by default.
- **ISF multi-pass** works — `PASSES` with a `PERSISTENT` target give feedback / trails (branch on `PASSINDEX`). **ISF filters** work too: an ISF whose image input is named `inputImage` processes the **layer below** — put it in a channel's **fx2** over an effect in **fx1**.

Where to find or create shaders — for **Shadertoy**, pick single-*Image*-tab shaders without `iChannel` / Buffer tabs (that multi-pass style isn't supported; ISF `PASSES` multi-pass is):

- ISF library ↗ (https://interactiveshaderformat.com/) and The Book of Shaders ↗ (https://thebookofshaders.com/) — most likely to import cleanly.
- Shadertoy ↗ (https://www.shadertoy.com/) — huge library; use single-*Image*-tab, no-`iChannel` shaders. GLSL Sandbox ↗ (https://glslsandbox.com/) too.
- KodeLife ↗ (https://hexler.net/kodelife) — write your own, live.

> **Full guide:** the 📖 **Shader guide** button in the shader editor — dialects, the uniform contract, ISF format, where to get shaders and examples (also: Custom Shader writing guide ↗).

Pixel mapping — odd shapes (footprints)

The Output Patch can map the picture onto shapes that aren't a plain rectangle — rings, arcs, polygons, rotated panels, or a hand-drawn layout. In a patch row's footprint editor, pick a **Shape**:

- **Ring / arc** — centre, radius, LED count, start/end angle (0–360 = full circle).
- **Polygon** — triangle, hexagon, any number of sides; LEDs spaced round the outline.
- **Custom / imported map** — a list of pixel positions in wire order (JSON `[[x,y]]` or CSV `x,y` lines; **Import** / **Export** as CSV). It has a full mini-editor with four modes and its own **Undo / Redo** (Ctrl+Z/Y) and **Clear all** — see below.
- **Rotate°** spins any footprint (e.g. 45 for a diamond).
- **Colour order** — the byte order this fixture's pixels expect (GRB for most strip, **GRBW** for RGBW tubes). It's the same per-row order as the patch table's Order column — set it in either place. Handy for mixing fixture types on one rig.

Prebuilt fixtures — one-click presets that drop a correctly-sized footprint so you don't build it by hand:

- ⬢ **HexVIBE…** — **Flow** lays a 93-pixel hexagon (map one patch row per FLOW panel — Start Pixel 0 = left, 93 = right, nudge Centre X apart). **Flash** is a provisional placeholder pending confirmed specs. Needs a 2-D matrix (e.g. 64×64) so the hexagon has room.
- 📏 **SmartShow Pixel Tube…** — PIXEL-100 / 200 / 300 drops a straight **vertical** line of that many pixels (1 pixel per matrix cell), and **auto-sets the colour order to GRBW** for you (these are RGBW tubes). If the matrix is shorter than the pixel count it clamps and warns you to make it taller for full detail.

The custom-map editor

With **Custom / imported map** selected you edit pixels straight on the mini-map. One mode is active at a time (toggles under the map); leave them off for the default **Edit**:

- **Edit** — **drag** a pixel to move it (grabs the nearest), **double-click** empty to add, **right-click** to delete.
- ✣ **Move whole map** — drag anywhere to shift every pixel together.
- 🔗 **Re-wire** — set the chain order: **drag along the cable** (or click pixel by pixel) and each pixel becomes the next in sequence — drawn as a bright numbered cyan chain, un-wired pixels dimmed. Right-click steps back; ↺ **Start over** resets.
- 🖍 **Trace** — build a map from scratch by clicking along your chain in order. Turn on **Show ref** and load a photo of your rig first to trace straight over it.

Bridge (null) pixels: Ctrl-click a pixel to toggle it — it stays in the chain (keeps addressing) but is never lit, for LEDs that just pass through somewhere you don't want light. Shown as an orange slashed ring (⊘) with a running tally; Ctrl-click again to turn it back on. In CSV a bridge line ends with `,skip`.

The map shows the **wire order** so you can tell which LED is which: pixel 0 is green, the last pixel red, with a path along the sequence and a hover readout (`pixel N · DMX ch a–b`). Tick **Show output** to light every LED with the running effect and see the shape as it'll actually look.

Remember: effects always paint a *rectangle*; a footprint just samples a region of it — so make the canvas comfortably bigger than the shape. **Full guide + a worked ring example:** the 📖 **Guide** button in the footprint editor (also: Pixel mapping guide ↗).

Colour pickers

Every colour picker in the app has the same shape:

- **16-preset palette** at the top — 8×2 grid of show-lighting colours. **Single click = applied + dialog closes**.
- **Custom swatch** below — click to open the OS colour picker.
- **Clear / default** button — removes the override.
- Dialog is draggable by the title bar. Esc = cancel.

Bit depth

LMS is **8-bit per channel end-to-end** (R/G/B/W 0–255). This matches DMX / Art-Net / sACN native channel size and the vast majority of LED chips (WS2812B / WS2811 / SK6812 are all 8-bit native). 16-bit transport only adds value on APA102/SK9822, WS2814, or 16-bit DMX fixtures — for everything else, the extra precision is quantised away at the strip.

External Control IN

LMS accepts incoming control from lighting desks, DAWs, DJ decks, tablets, and hardware controllers, on the Settings page → **External Control IN** card. Several can run at once. **Any incoming DMX channel, MIDI note/CC, or keyboard key can be LEARNED onto LMS** — the Master and every Show-page fader, plus every scene / chase / action button — and you can review or edit every assignment from one **Control Bindings** table (see below).

Art-Net DMX IN

Listens for Art-Net OpDmx packets on UDP (default port 6454). Patch LMS like any other fixture from your lighting board, then Learn its channels onto faders / buttons. Triggers fire on any non-zero value; faders map 0..255 → 0..100%. Auto-detects universe + channel during Learn — just push the fader you want. Multi-universe supported.

DMX IN (USB — ENTTEC Pro)

Reads DMX **straight off an ENTTEC USB Pro (Mk1 / Mk2)** plugged into the PC — no Art-Net node required. Click **Connect dongle**, pick the Pro, and incoming DMX feeds the same learn/bind system as Art-Net; the status pill reads "Receiving · N frames" when it's live. (The cheaper *Open DMX USB* is output-only and won't work — input needs a real Pro.)

sACN E1.31 IN

Multicast DMX over IP — venue standard. Joins multicast group `239.255.<u-hi>.<u-lo>` per universe. Configure one or more universes (comma-separated) on the Settings card. Same binding model as Art-Net.

OSC IN

Open Sound Control over UDP (default port 8000). Used by TouchOSC (iPad / Android), Resolume, QLab, ProPresenter, Bitwig, and any modern show-control software. LMS's OSC parser supports OSC 1.0 bundles + messages with `i` / `f` / `s` / `b` / `T` / `F` typetags.

Built-in addresses (no learning needed if you use these):

- `/lmc/master` — first float arg 0–1 sets master fader
- `/lmc/crossfade` — first float arg 0–1 sets crossfade A↔B
- `/lmc/scene/<sceneId>` — fires scene with first truthy arg (or no args)

Any other address can be **learned** onto any LMS binding via right-click → Learn OSC on a Show Mode button (when wired into the right-click menu).

MIDI Clock IN

Receives 24 ppq MIDI clock from a DAW / DJ deck / drum machine and locks chase tempo to it. Plug a USB MIDI device in, tick the **Sync chases to MIDI clock** checkbox, hit play on the master. LMS's running chase advances exactly one step per beat — no drift, no manual nudging.

- Live BPM readout (smoothed from the last 24 ticks).

- Responds to 0xFA (start) / 0xFB (continue) / 0xFC (stop) transport messages.
- Works alongside the existing MIDI input learn flow — same device, different message stream.

MIDI Clock OUT

Broadcasts LMS's tempo as 24 ppq clock to a chosen MIDI output device. Makes LMS the master clock that DAWs / drum machines follow. Send a 0xFA start + clock ticks at configurable BPM (20–300). 0xFC stop on disconnect.

MIDI Time Code IN

Receives MTC quarter-frame messages (0xF1) and decodes the full `HH:MM:SS:FF` timestamp. For film / show / theatre sync workflows where a timecode master drives the cue list. Live readout on the Settings card.

> **MIDI feedback OUT — controller buttons light up when scenes fire.** Per-scene/chase bindings let LMS send a note-on to a hardware controller (APC mini / Launchpad / X-Touch) so the matching pad lights up when the scene is active. Velocity sets colour on RGB grids. Bindings travel with the project (`project.midiFeedback`).

Learning controls onto faders & buttons — keyboard, MIDI & Art-Net/DMX

Right-click any Show Mode **button** (scene / chase / BLACKOUT / STROBE / FADE / X-FADE) or any Show-page **fader** (Master, Strobe Hz, Gamma, Fade Time, X-Fade Time) and you'll find **Learn keyboard key…**, **Learn MIDI input…** and **Learn Art-Net/DMX in…**. Pick one, then press the key / hit the pad / move the desk fader — that input is now bound. Re-learn or **Clear** any time; the Clear item shows the current assignment (e.g. `Clear MIDI input (Note ch1 #36)`).

- **Faders take continuous controls; buttons take on/off.** Bind a fader to a MIDI **CC** or a DMX **fader** channel — it follows the value smoothly. Bind a button to a MIDI **note** or a DMX **on/off** channel — it fires the instant the value leaves zero. (A scene bound to a CC won't trigger; a fader bound to a note won't move — match the control type.)
- **"Already assigned" prompt.** If you learn an input that's already used elsewhere, LMS asks before overwriting — **Cancel** keeps things as they were, **Overwrite** moves the input to the new control. Each control keeps one input of each type.
- **BLACKOUT** stays live even in Performance Mode lock (panic button) unless you tick BLACKOUT in the lockmask. Scene/chase triggers always fire in lock; the Master fader respects the lockmask "Master fader" flag.
- Bindings travel with the project — keyboard + MIDI on the Builder model, Art-Net/DMX + OSC in the project file. Load the project on another machine and they're wired the same way.

Control Bindings table — see & edit everything in one place

Reachable from the menu bar — 1.1.6. Control → Control Bindings…, or `Ctrl`+`B`, from anywhere in the app. It used to be a button at the very bottom of the External Control tab, which is the one place you cannot get to during a show. The button is still there, now at the top of the card.

Settings → External Control → **All control bindings…** opens a single table of every assignment, grouped **Faders / Actions / Scenes / Chases**, with a column each for ⌨ **Keyboard**, 🎹 **MIDI** and 🎛 **Art-Net/DMX**. Each cell shows the current binding (or **+ Learn**), with inline **Learn / Re-learn** and **Clear** — so you can wire up or audit your whole control surface in one window instead of hunting through right-click menus. **"Show only assigned"** collapses it to what's actually bound. The window is draggable and closes with the ✕ or Esc.

> **MIDI feedback OUT — controller pads light up when scenes fire.** Per-scene/chase bindings can send a note-on back to a hardware controller (APC mini / Launchpad / X-Touch) so the matching pad lights when the scene is active; velocity sets colour on RGB grids.

Keyboard shortcuts

Key	Action
`Esc`	Return to the Builder from any sub-page (Show Mode / Settings) — unless Performance Mode is locked, or a dialog is open
`Window X` on a sub-page	Return to the Builder (does not quit)
`Window X` on the Builder	Quit the app
`F1`	Open this help
`F11`	Toggle full-screen (also under File menu)
`Ctrl+N`	New project
`Ctrl+O`	Open project…
`Ctrl+S`	Save project…
`Ctrl+,`	Open Settings
`Enter` in a prompt	Accept / OK
`Esc` in a prompt / colour picker	Cancel
`Right-click` a Show Mode button	Open the per-button context menu
`Double-click` a scene name in the Scenes pop-out	Rename

Project files — .lms and legacy .lmc

Projects save as `.lms` from beta.60b onwards. **Open** still accepts the legacy `.lmc` extension used by every earlier build, plus a raw `.json` payload — nothing you have saved is stranded, and there is no conversion step to run.

The two extensions are the **same format**; only the name changed. A `.lmc` opens, edits and plays identically, and an operator on an older build can still open a file you re-save as `.lms` if you rename it back.

Opening an old `.lmc` and hitting **Save** suggests the same name with `.lms`, leaving the original file untouched — so upgrading a show is deliberate, never a surprise. To overwrite the `.lmc` in place instead, pick **LMS Project (legacy)** in the save dialog's file-type dropdown. One exception: **Auto-save on close** (Settings) writes back to whatever file you actually opened, so a legacy project stays `.lmc` until you Save it somewhere new yourself.

Updates & partnership

LMS is built by Lee James Apps (https://leejamesapps.co.uk) **in partnership with Smartshow Lighting (https://smartshow.lighting/)** — the Smartshow logo + leejamesapps.co.uk link both appear in the brand strip under Master in the Builder, and again in the About dialog.

In-app auto-update: the installed (NSIS) build checks for a new version on every launch (and via `Help → Check for updates…`). When one's available you get a centred popup with the release notes and a **Download & install** button — it downloads inside the app with a progress bar, then **Restart & install now** installs it and relaunches. No website visit, no manual download. (The *Portable* build can't self-install — it shows a banner with a download link instead.)

Anonymous telemetry: a one-line launch ping (version, OS, locale, install-id) is sent to the LMS backend so we can see install counts + version distribution. No personal data, no fingerprinting; the install-id is a random UUID stored in the app's userData folder.

Troubleshooting

Audio-reactive effects aren't reacting.
You haven't started audio capture. Settings tab → Audio Input → pick a Source → **Start**. Live level meter should move.

System music isn't reaching audio.
Microphone input picks up the room only. Install a loopback driver (VB-Audio Cable on Win, BlackHole on Mac), route your music player's output through it, then pick that as the source in LMS.

Webcam shows black.
Almost always Windows Privacy → Camera → "Let desktop apps access your camera" being OFF. Toggle it on, restart the effect.

Video file won't play.
AVI / MKV / WMV / FLV / DivX and similar are now **auto-converted to MP4 on import** (a "converting… %" readout shows during conversion). If conversion fails, the file may be corrupt or use an exotic codec — re-export it from your editor as MP4 (H.264) and re-import. Converted files are cached under the app's data folder; that cache self-prunes when it gets large.

Live FPS is well below target.
Either the matrix is too big for the current effect stack, or one of the effects is expensive (Conway's Life and audio bars with lots of bands can be heavy). WebGL-marked effects ⚡ are the cheapest.

Gliediator board isn't lit.
Most DIY firmwares want 1,000,000 baud — try that first. Some older boards need 500k or 115k. Reconnect after changing baud.

A DMX tester reports BAD signal / no channels on a USB DMX dongle.
Check *Device type* on the Serial DMX card — 1.1.8. There are two completely different families of USB DMX dongle and they need different data on the wire:

- **ENTTEC USB Pro / DMXKing** contain a microcontroller. LMS hands them a framed packet and the dongle generates the DMX break and timing itself.
- **Open DMX / raw FTDI-RS485 adapters** — the cheap Waveshare and eBay boards — have **no microcontroller at all**. They are plain USB-to-RS485 converters, so whatever bytes LMS writes land on the wire verbatim and **LMS has to generate the break itself**.

⚠ Set to the wrong one, nothing errors: the Pro framing bytes go onto the DMX line as if they were channel data, with no break anywhere. A tester shows a **BAD** signal, timings that match nothing, and no channels received — which is exactly what one looked like before this setting existed. The same dongle in another program works because that program has a separate Open DMX path.
⚠ The Open DMX break is timed in software, so it is far longer than the ~100µs a microcontroller produces. Measured on a tester's machine it ranges from **4.5 ms to 18 ms**, and it varies because generating it costs two USB round trips whose replies have to be scheduled back onto a renderer that is busy drawing your matrix — **your frame rate lands inside the break**. That is still within spec, as DMX512-A permits a transmitter break up to a full second, but a meter will report it as unusually long while showing OK and 512 channels.
Refresh rate — 1.1.10. A frame is that break plus **22.6 ms** of data, so with a break of 4.5–18 ms the

real ceiling is roughly **25–37 Hz**, not the 44 Hz DMX allows. **Raising your render rate will not raise the DMX rate** — the break has already spent the headroom.
LMS paces Open DMX from the moment a frame *finishes* leaving the port, never from when it started. That distinction matters: the serial write reports success when the bytes are **accepted**, not when they have been **transmitted**, and they keep draining for another 22.6 ms afterwards. Pacing on a fixed interval let the next break be asserted mid-frame, which **truncated the frame** and showed up on a meter as an intermittently bad signal at high render rates. Frames that arrive too soon are **skipped rather than queued** — queuing would trade a dropped frame for growing latency, and on a lighting output late is worse than never. Skipped frames are deliberately **not** counted as drops. The Connected box shows the measured break, the rate achieved and the ceiling this machine can actually reach.
ENTTEC Pro dongles are unaffected — their microcontroller does its own timing and genuinely reaches 44 Hz.

Open DMX output is unstable, or a tester reports a very long break.
1.1.11. The break LMS makes on a raw adapter is generated in software, and how long it lasts is mostly out of our hands. A tester measured **7–9 ms** from another program on the same dongle and machine, so a few milliseconds is simply what USB costs — but LMS was adding **10–17 ms** on top, and the excess grew as the render rate rose.
⚠ The cause is that waiting for the "break on" command to be acknowledged **hands the app its own event loop back**, so a matrix render or a garbage collection can be scheduled *inside* the break.
The answer is the native FTDI driver. Set **Device type** on the Serial DMX card to **Open DMX — native FTDI driver**. It makes the break in the driver rather than in the browser, which the browser simply cannot do properly, and it is the recommended setting on Windows.
An experimental **Break timing** control existed in earlier versions for hand-tuning the break on the raw Web Serial path. The native driver made it unnecessary and it is no longer shown. If you set it in an older version, your setting is still honoured — nothing has changed on your rig.
ENTTEC Pro dongles are unaffected either way: they time their own break in hardware.
⚠ The break figure on the Connected box reads **break cmd** because that is what it is — how long our commands took, not the break on the wire. The two can differ by an order of magnitude. Only a DMX tester measures the real thing.

Reading DMX *in* from a cheap USB-RS485 adapter.
Experimental — 1.1.9. The **DMX IN (USB)** card has a *Device type* setting. An **ENTTEC Pro** is the reliable route and the one we recommend: its microcontroller finds the DMX frame boundaries in hardware and hands the PC clean packets.
⚠ A raw FTDI / RS485 adapter has no microcontroller, and DMX carries **no length and no checksum** — the only thing marking the end of a frame is the break, which a serial port reports as an error rather than as data. In the experimental mode LMS reconstructs the frames itself, using the break where the driver reports one and otherwise the idle gap between frames. **It may drop or mis-frame packets.**
1.1.10. Where the driver reports breaks, LMS now uses **only** that and ignores the idle-gap method entirely. Running both was worse than running either: they split each frame between them and neither received a whole one. The gap method is a fallback for drivers that do not report breaks, and it only works **below about 25 Hz** — that is arithmetic rather than a limitation we can tune away. At 40 Hz the silence between frames is about 2.4 ms, while the serial driver hands bytes over in lumps that leave gaps of around 11 ms *inside* a frame, so the boundary being hunted for is smaller than the noise. **If the fallback is all your adapter offers, wind the sending desk down to 20 Hz or slower.** That is usually

fine for *triggering* things — a fader update arriving late is invisible, unlike a dropped output frame — but use a Pro anywhere it has to be dependable.
The card shows a diagnostic line while the experimental mode is connected: which framing method is working, byte rate, measured gap and frame length. **If you report a problem with this mode, please include that line** — it is what tells us which part is failing.

Glediator output is scrambled, skewed or rolling.
The matrix size in LMS almost certainly does not match the size the board's sketch was built for. The Glediator protocol carries **no width, height, length or checksum** — it is a `0x01` sentinel followed by raw pixel bytes, and the receiver simply reads *width* × *height* × *3* of them and lights its strip. Those dimensions are compiled into the sketch, so if it was flashed for 16×16 and you set LMS to 20×20 you get a diagonal, rolling mess rather than an error. **Nothing in LMS can warn you about this**, because there is nothing in the protocol to check against — it is the price of a format simple enough for an ATmega to parse. Set **Matrix** to exactly what the sketch expects, or reflash the board to match.
Two related things: the **Pixel wiring** and **Start corner** on the Matrix tab must also match how the strip is physically snaked (a serpentine wall fed as rows comes out with every other line reversed), and a small board can simply run out of memory — an Arduino Uno/Nano has 2KB of SRAM against 3 bytes per pixel, so it tops out around 300–400 pixels however you configure it. For anything bigger, people use a Teensy with an OctoWS2811 adapter.

Locked yourself out and forgot the PIN.
Quit LMS. The PIN persists in localStorage. Set `LMC_DEVTOOLS=1` env var, restart, Console: `localStorage.removeItem('lmc.jinxmix.v1')`. That clears scenes/chases/show config too — back up your `.lms` project file first.

LED Matrix Server — by Lee James Apps · in partnership with Smartshow Lighting