SlyLED User Manual

3D Volumetric Lighting System — v1.0.0

Complete guide to designing, programming, and running LED + DMX lighting shows with the SlyLED orchestrator, performers, and DMX bridge.

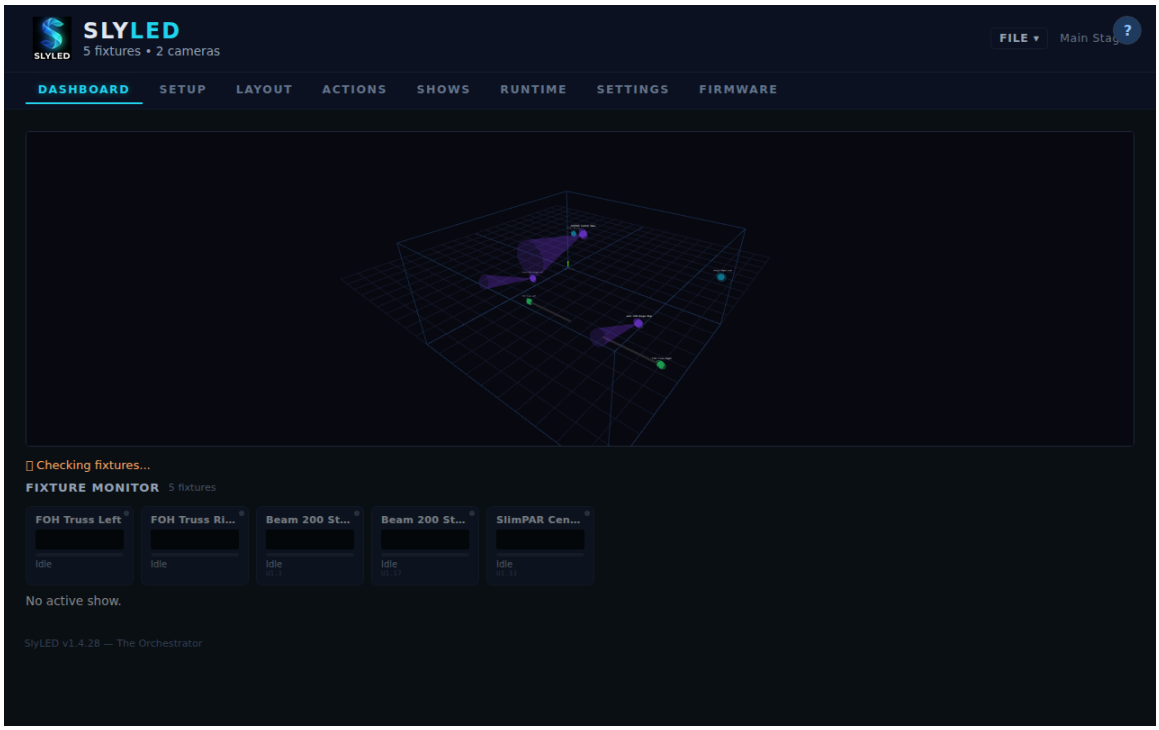
electricrv.ca/slyled

Table of Contents

1. Getting Started with 3D Stage Design

Overview

SlyLED is a three-tier lighting system: the Orchestrator (Windows/Mac desktop app) designs and controls shows, Performers (ESP32/D1 Mini boards with LED strings) execute lighting effects, and DMX Bridges (Giga R1 boards) drive professional DMX fixtures over Art-Net.

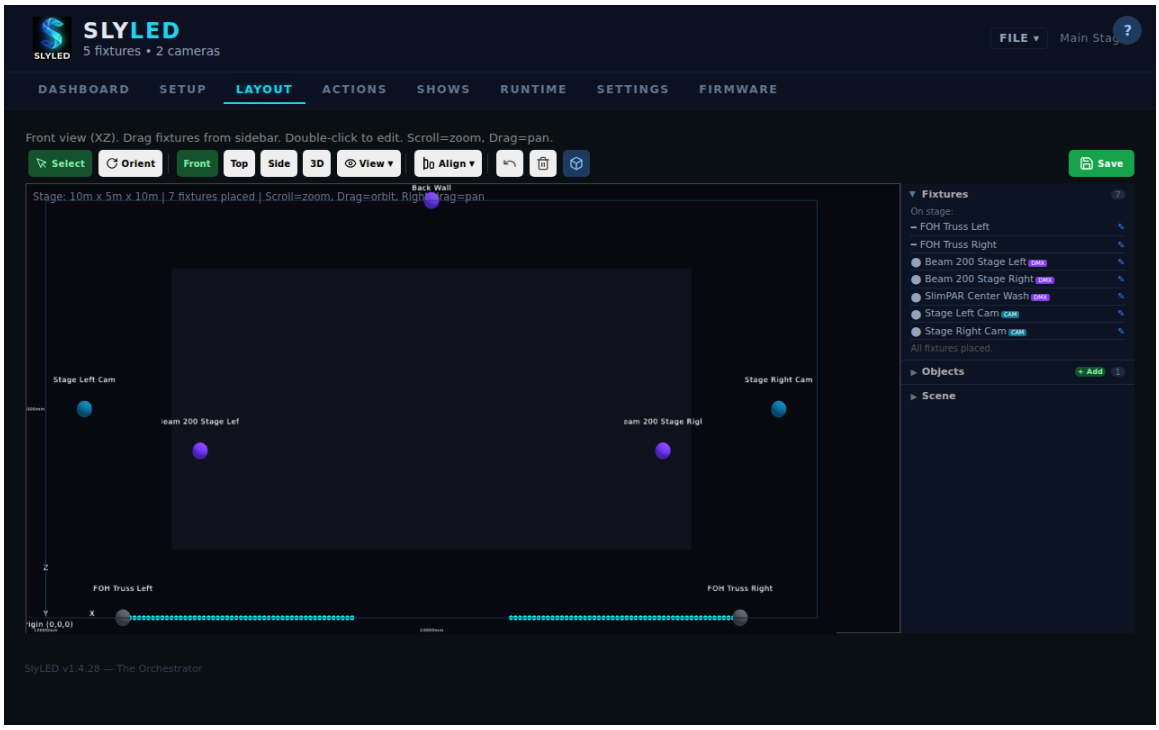


Dashboard — live performer status overview

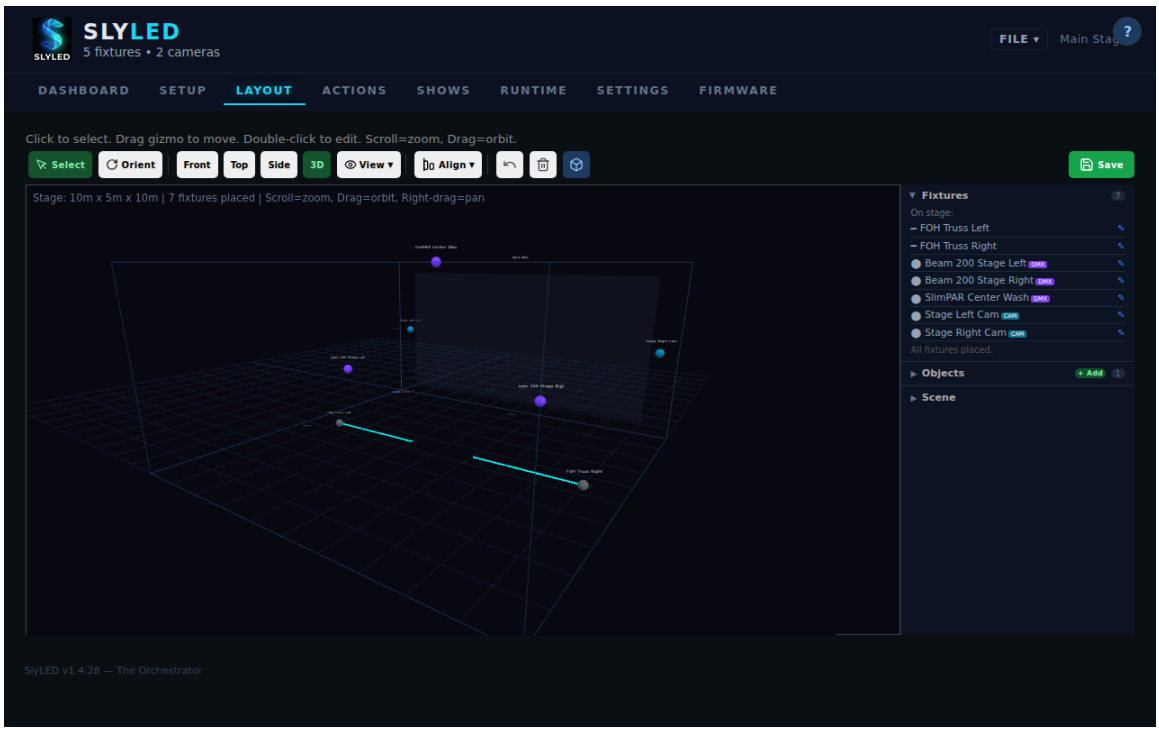
Switching Between 2D and 3D

The Layout tab offers two views:
- 2D Canvas: Flat top-down layout for simple setups
- 3D Viewport: Interactive Three.js scene for complex multi-level installations

Both views share the same position data. Switching is instant and non-destructive.



Layout tab — 2D Canvas with placed fixtures



Layout tab — 3D Viewport with stage wireframe, fixtures, and objects

Navigating the 3D Viewport

Action	Control

Orbit (rotate)	Left-click + drag
Zoom	Scroll wheel
Pan (shift view)	Right-click + drag
Select fixture	Left-click on node
Move fixture	Drag the colored arrows
Edit fixture	Double-click on node
Place from sidebar	Drag unplaced fixture into viewport

Coordinate System

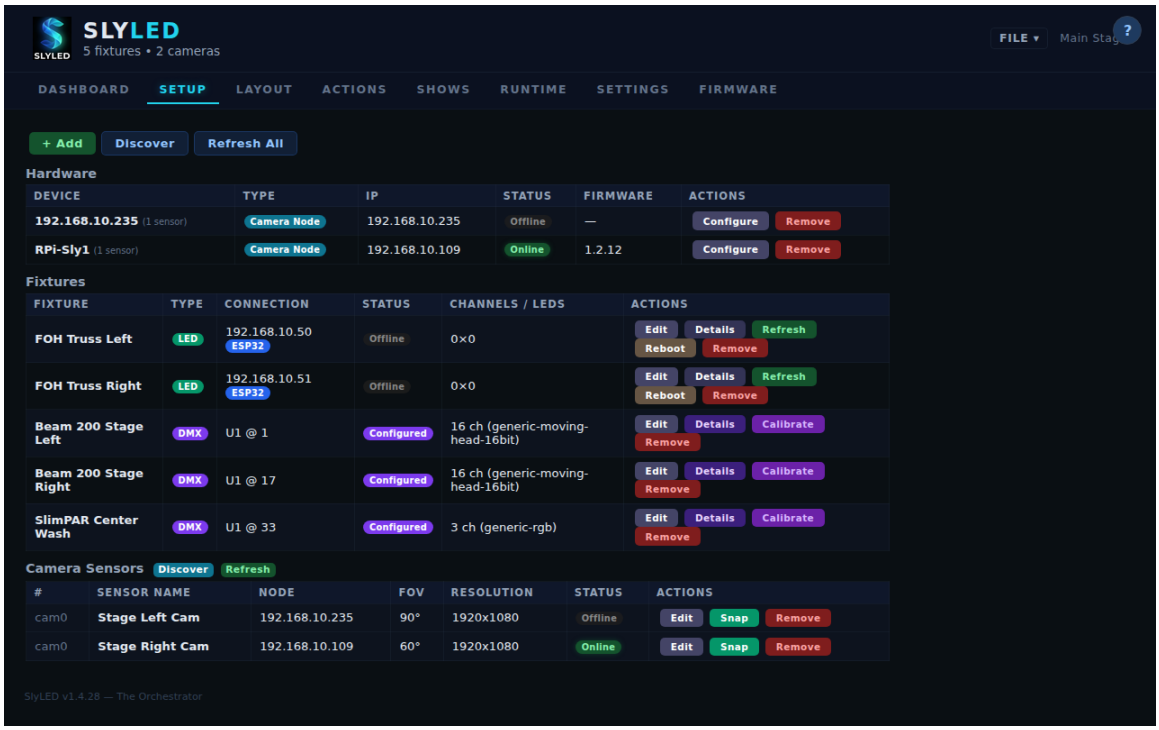
- X-axis (red): Width — left to right
- Y-axis (green): Height — ground to ceiling
- Z-axis (blue): Depth — front to back
- Origin: Bottom-left-front corner of the stage
- Units: Internally millimeters; displayed in Settings-chosen unit

2. Fixture Setup

What Are Fixtures?

A fixture is the primary entity on the 3D stage. It wraps physical hardware and adds stage-level attributes like position, rotation, and aim point.

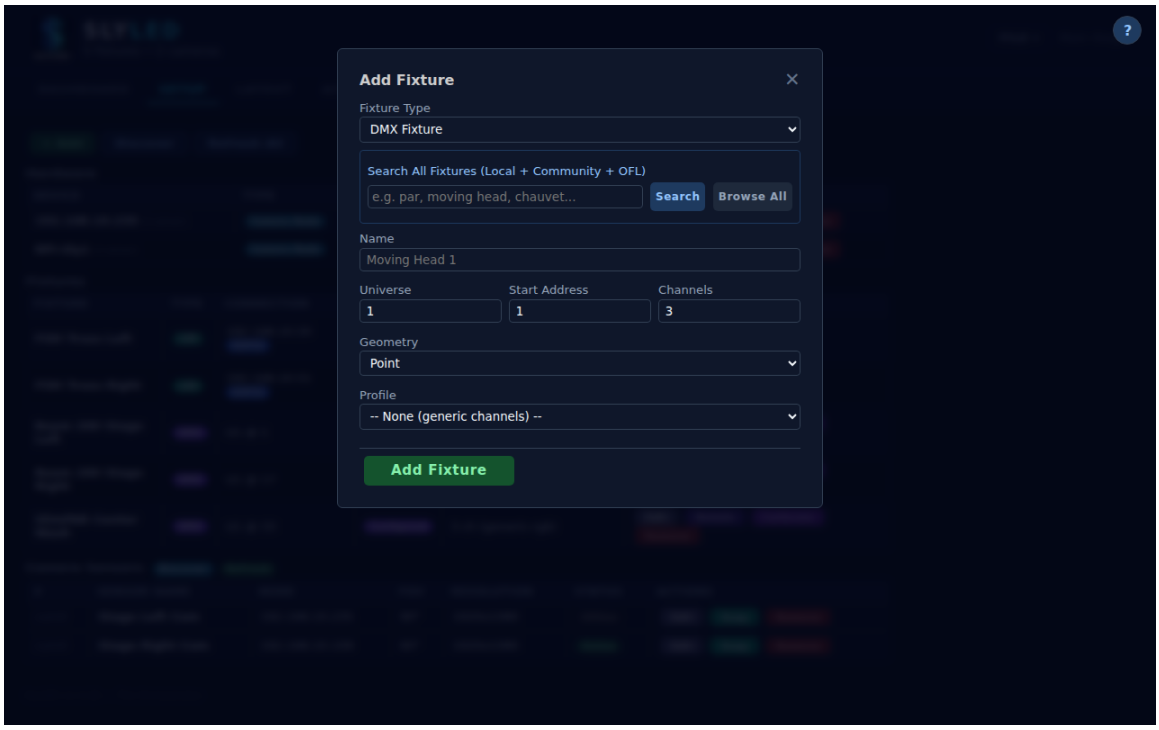
- LED Performers: Auto-created when hardware is registered via Setup tab
- DMX Fixtures: Created manually with universe, address, and profile assignment
- Fixtures can override child attributes (e.g., rotate a horizontal string to vertical)
- The baking engine uses the fixture's position and rotation, not the child's raw config



Setup tab — LED and DMX fixtures with type badges and status

Adding an LED Performer

Click Add Fixture, select "SlyLED Performer (LED)", and enter the device IP address. The system probes the device via UDP PING and HTTP, registers it as a child, and auto-creates a linked fixture.



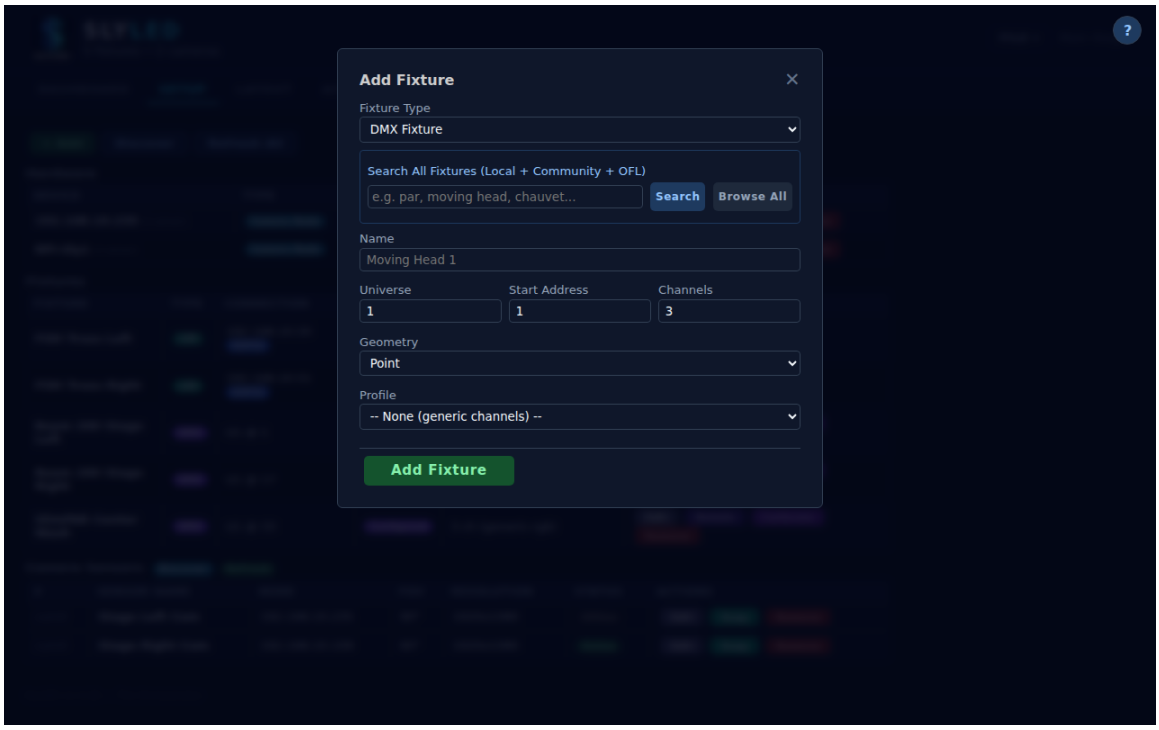
Add Fixture — LED performer flow with IP address entry

Adding a DMX Fixture

Click Add Fixture, select "DMX Fixture", and configure:

- Name: Descriptive label
- Universe: DMX universe number (1+)
- Start Address: DMX start channel (1-512)
- Channel Count: Number of channels the fixture uses
- Profile: Select from the built-in library or import from Open Fixture Library

DMX fixtures appear in the setup table with purple "DMX" badges.

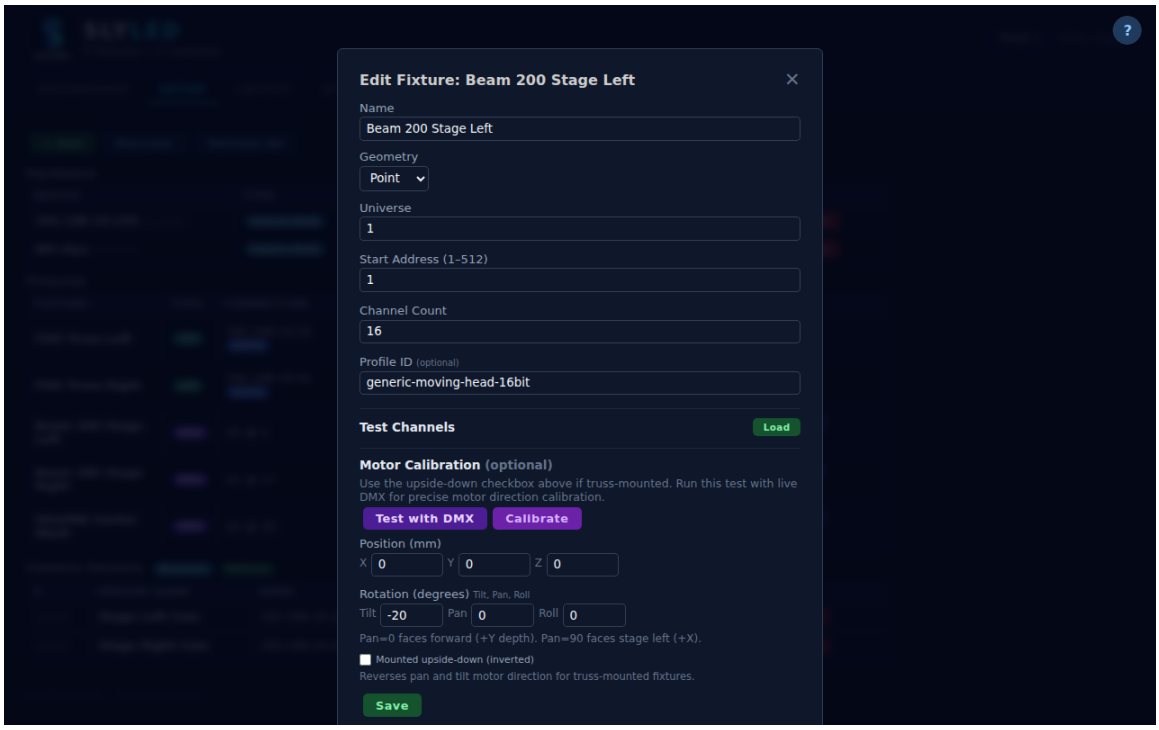


Add Fixture — DMX fixture with profile dropdown

Editing DMX Fixtures

Click Edit on a DMX fixture to modify its properties. The edit modal shows:

- Universe, Start Address, Channel Count
- Profile ID (for channel name/type mapping)
- Aim Point (X, Y, Z in mm) — where the beam points for moving heads
- Test Channels — interactive sliders to control each DMX channel live
- Rotation Override (degrees)



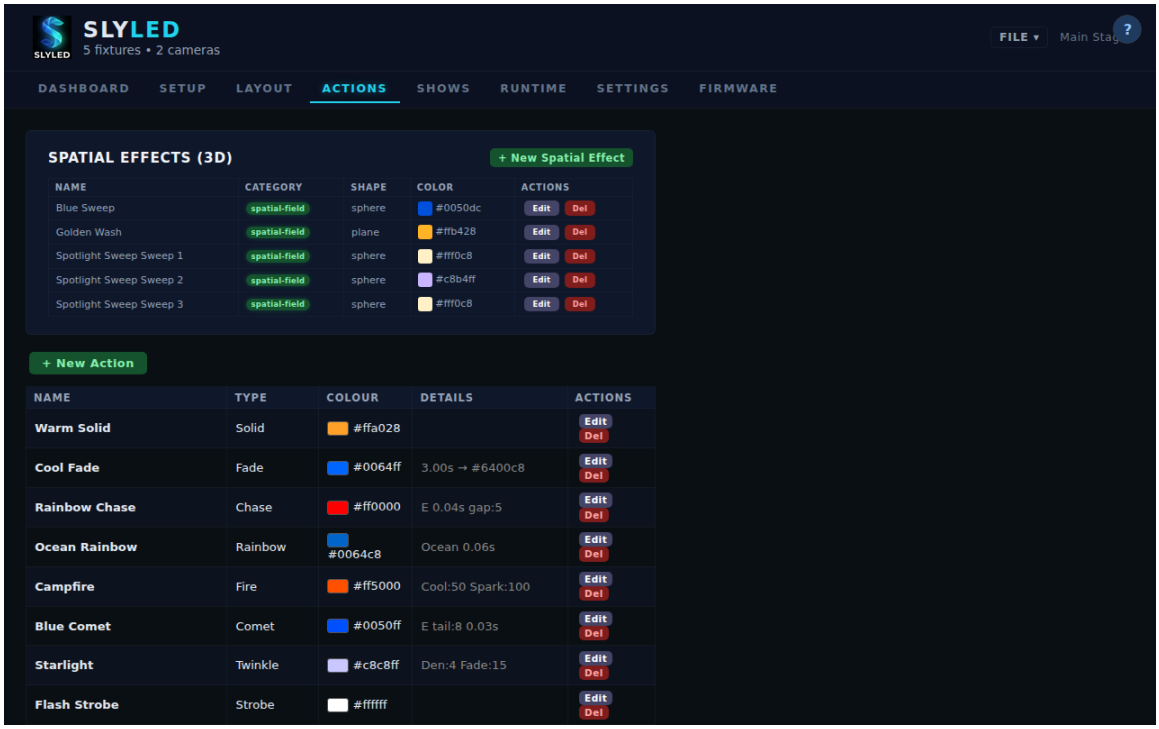
Edit DMX fixture — aim point, profile, and test channels

Fixture Types

Type	Description	Use Case
Linear	LED strip/string with pixels along a path	LED performers
Point	Single light source with area of effect	DMX pars, spots, moving heads
Object	3D mesh as a projection target	Walls, screens
Group	Named collection of fixtures	Zones, grouped control

3. Creating Spatial Effects

Spatial effects operate in 3D space. A sphere of colored light sweeping across the stage illuminates different fixtures at different times based on their physical position. For DMX moving heads, the effect center becomes the aim target — heads track the effect as it moves.



Actions tab — spatial effects and classic action library

Spatial Fields

Field	Description
Shape	Sphere, Plane, or Box
Color	RGB color applied to pixels inside the field
Size	Radius (sphere), thickness (plane), or W/H/D (box)
Motion Start	Starting position [x, y, z] in mm
Motion End	Ending position [x, y, z] in mm
Duration	Travel time from start to end
Easing	Linear, ease-in, ease-out, ease-in-out
Blend	Replace, Add, Multiply, or Screen

Moving Heads + Spatial Effects

When a spatial effect is applied to a DMX moving head fixture:

- The effect's center position becomes the aim target
- Pan/tilt angles are computed from fixture position to aim point

- Color is applied from the effect's RGB values
- Moving effects produce time-sliced segments that track the motion
- 3D viewport shows beam cones pointing at the effect center

DMX Action Types

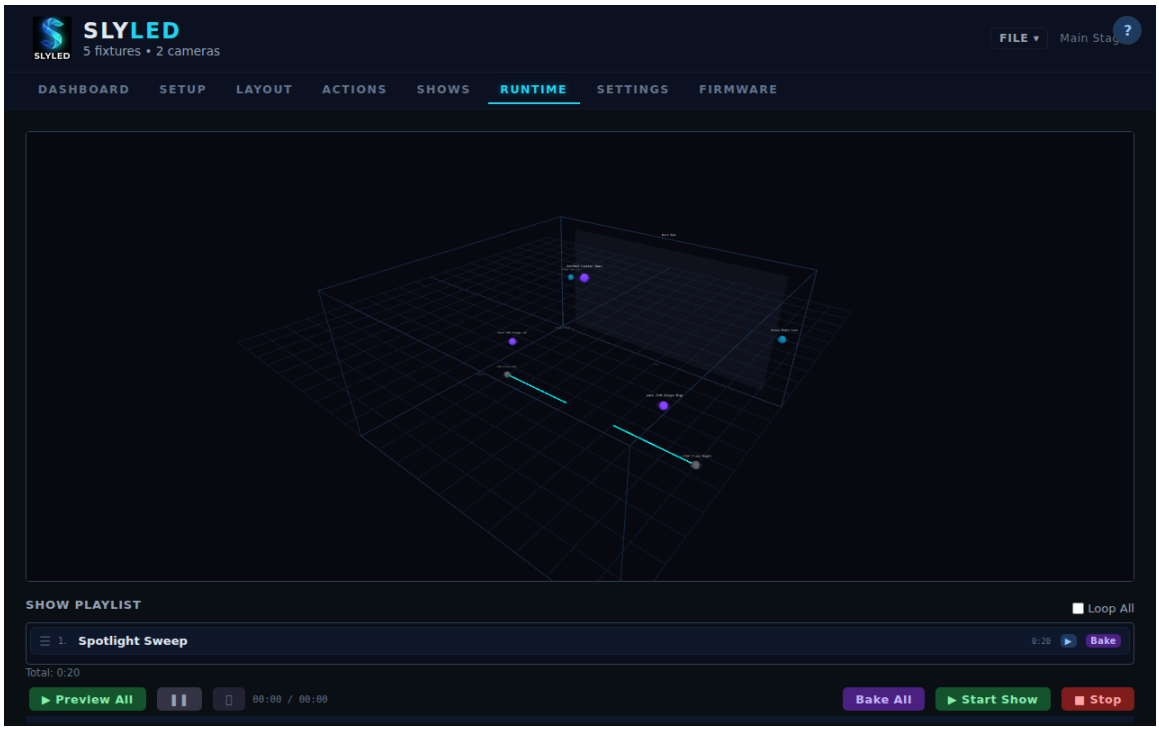
In addition to classic LED actions (Solid, Chase, Rainbow, etc.), four DMX-specific action types are available for direct control of DMX features:

Action Type	Description
DMX Scene	Set exact values for dimmer, pan, tilt, strobe, gobo, color wheel, prism
Pan/Tilt Move	Animate pan/tilt from start to end position over time
Gobo Select	Select a gobo wheel position with optional color
Color Wheel	Select a color wheel position

Classic LED actions (Solid, Breathe, Chase, etc.) are automatically converted to DMX Scene segments when assigned to DMX fixtures. The dimmer is auto-set to 255 when color is active, and pan/tilt default to center (0.5).

4. Building a Timeline

Timelines are multi-track, overlapping effect sequences with precise timing. Use the Runtime tab to create and edit timelines.



Runtime tab — timeline editor

Creating a Timeline

1. Click "+ New Timeline" and enter name and duration
2. Select the timeline from the dropdown
3. Add tracks (one per fixture or "All Performers")
4. Add clips referencing spatial effects or classic actions
5. Adjust clip timing by editing start time and duration

5. Baking & Playback

What Is Baking?

Baking compiles a timeline into minimal action instructions for each performer. The smart bake engine analyzes each clip's spatial geometry directly and computes per-string sweep patterns, directions, and speeds.

For DMX fixtures, baking also computes:
- Pan/tilt angles from spatial effect motion paths
- Color values from the effect's intersection with the fixture position
- Dimmer values (auto 255 when color is active)
- Time-sliced segments for smooth moving head tracking

Playback

After baking and syncing:
- LED performers receive action steps via UDP and execute locally
- DMX fixtures are driven by a 40Hz playback loop sending Art-Net/sACN packets
- All channels (RGB, pan, tilt, dimmer, strobe, gobo) are sent per profile
- 16-bit channels (pan/tilt) are split into coarse + fine bytes automatically

6. Show Preview Emulator

The emulator shows a real-time preview of your show on the Runtime tab.

- LED fixtures: Colored dots along string directions, animated per-pixel
- DMX fixtures: Beam triangles from fixture position toward aim point
- Beam color and alpha reflect the fixture's current DMX state
- Time counter shows elapsed position

7. Preset Shows

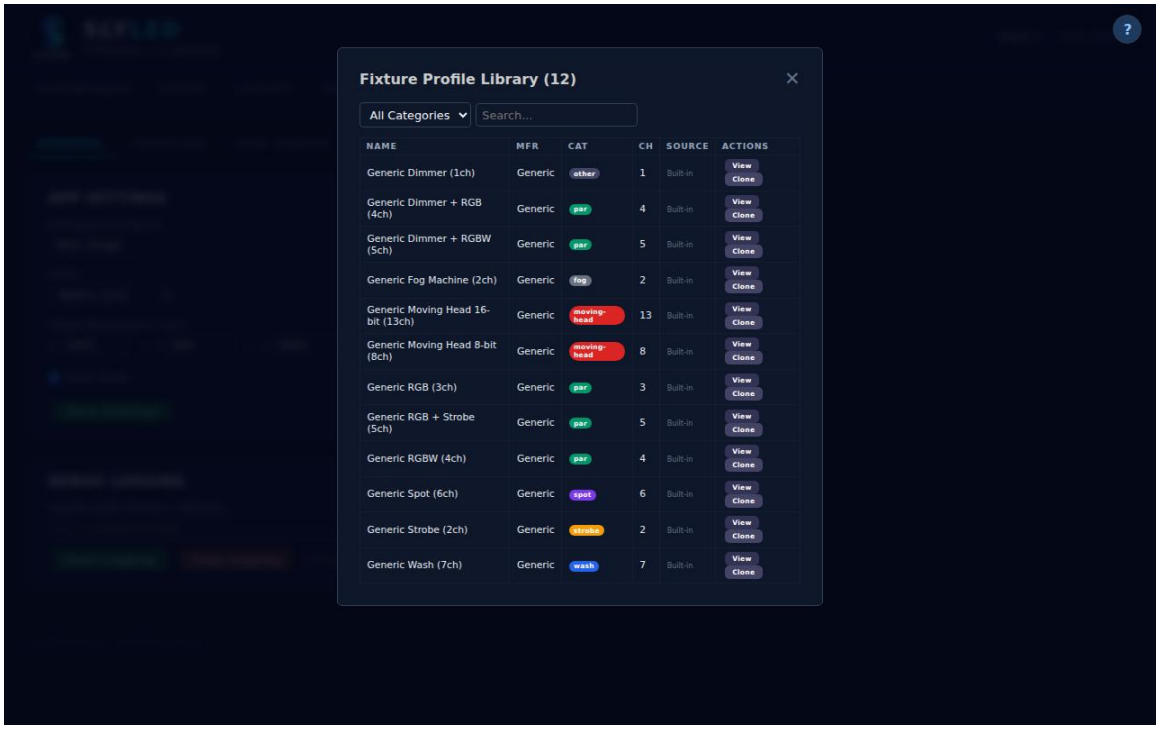
14 themed shows are available from the Runtime tab. Shows are dynamically generated based on your actual fixtures — every fixture gets coverage with no dark periods. LED fixtures get pattern effects, DMX pars get color washes, and moving heads get pan/tilt sweeps that track spatial effects across the stage.

Each load produces a unique variation of the theme with randomized timing and positions.

Preset	Type	Description
Rainbow Up	Spatial plane	Rainbow from floor to ceiling
Rainbow Across	Spatial sphere	Rainbow sweeping left to right
Slow Fire	Classic action	Warm fire effect on all fixtures
Disco	Classic action	Pastel twinkle sparkles
Ocean Wave	Spatial (2 effects)	Blue wave sweep with teal wash
Sunset Glow	Mixed	Warm breathe with golden plane sweep
Police Lights	Mixed	Red strobe with blue box flash sweep
Starfield	Classic action	White sparkles on dark background
Aurora Borealis	Spatial (2 effects)	Green curtain with purple shimmer
Spotlight Sweep	Spatial (moving heads)	Warm orb sweeps stage — heads track it
Concert Wash	Mixed (moving heads)	Magenta flood + amber tracking spot
Figure Eight	Spatial (moving heads)	Crossing orbs — heads trace X paths
Thunderstorm	Mixed (moving heads)	Lightning strikes — heads chase bolts
Dance Floor	Mixed (moving heads)	Fast orbiting spots — rapid tracking

8. DMX Fixture Profiles

Profiles define the channel layout and capabilities of DMX fixtures. Each channel maps a DMX offset to a function (pan, tilt, red, green, blue, etc.) with capability ranges that describe what each DMX value range does.



Fixture Profile Library — 12 built-in profiles

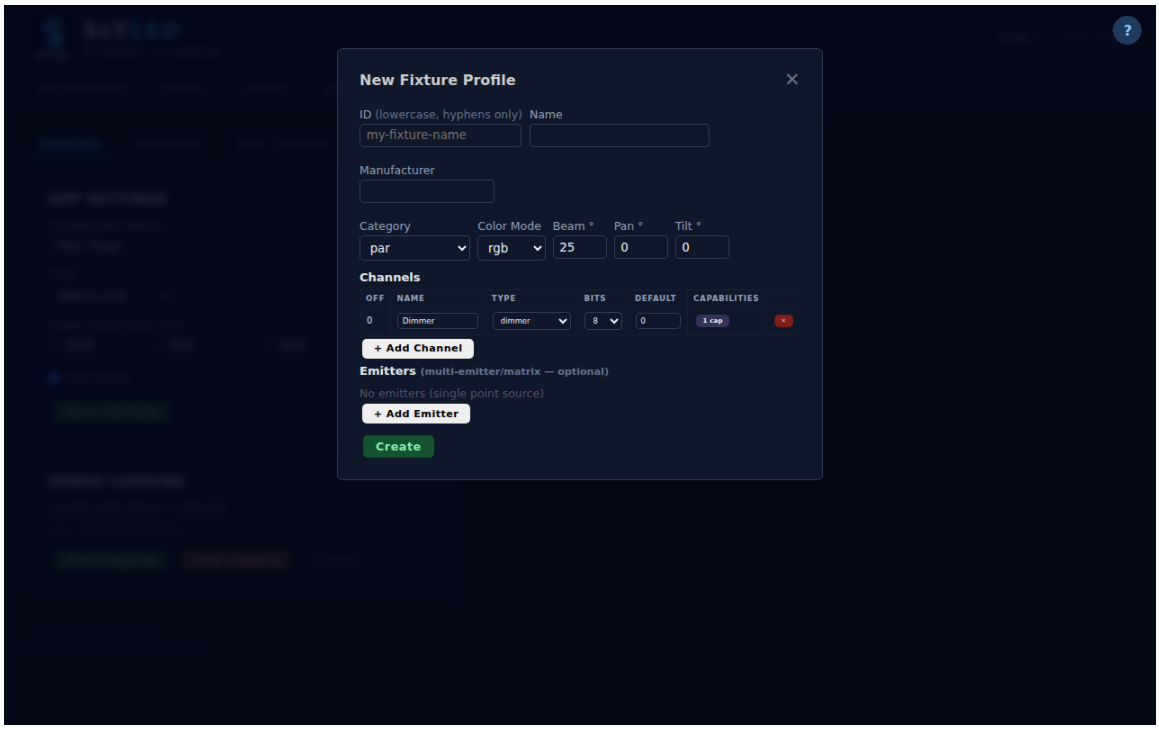
Viewing a Profile

Click View on any profile to see the full channel table with capabilities. Moving head profiles show pan/tilt ranges, beam width, and color mode.

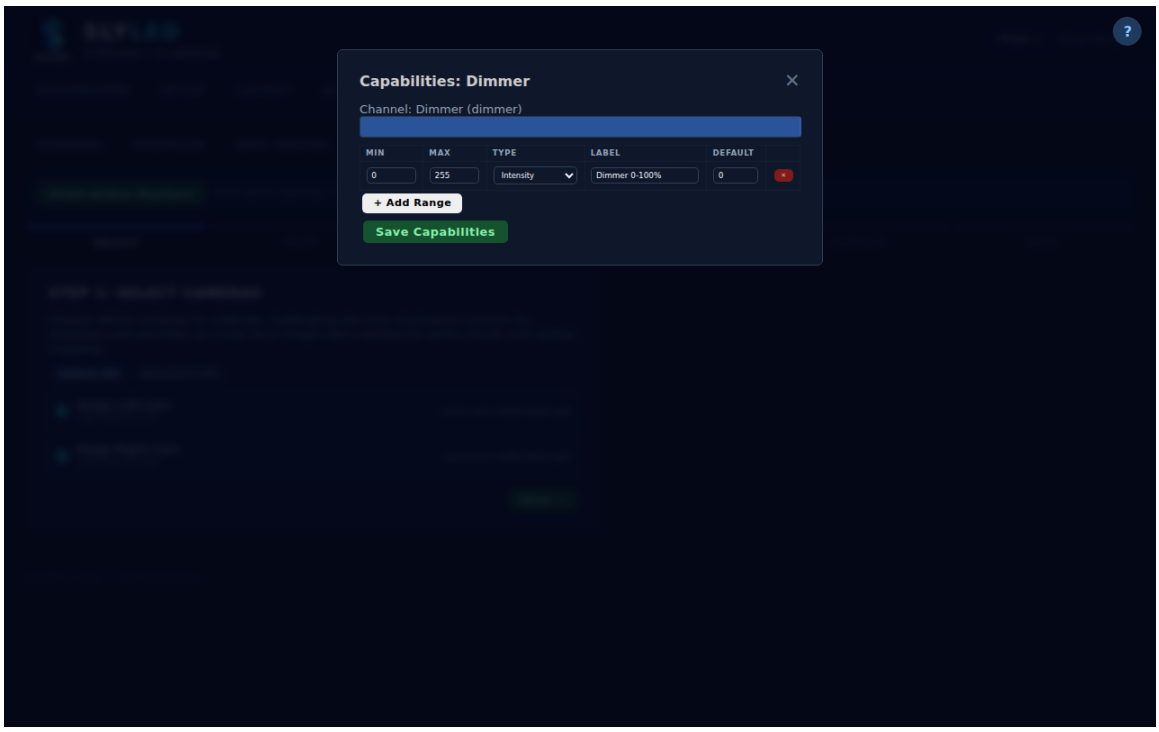
Moving Head 16-bit profile with 13 channels and capability ranges

Creating Custom Profiles

Click "New Profile" to open the editor. Define channels with type, name, bits (8/16), and capability ranges. Click the capabilities button on each channel to define what DMX value ranges do (e.g., gobo positions, strobe speeds, color wheel slots).



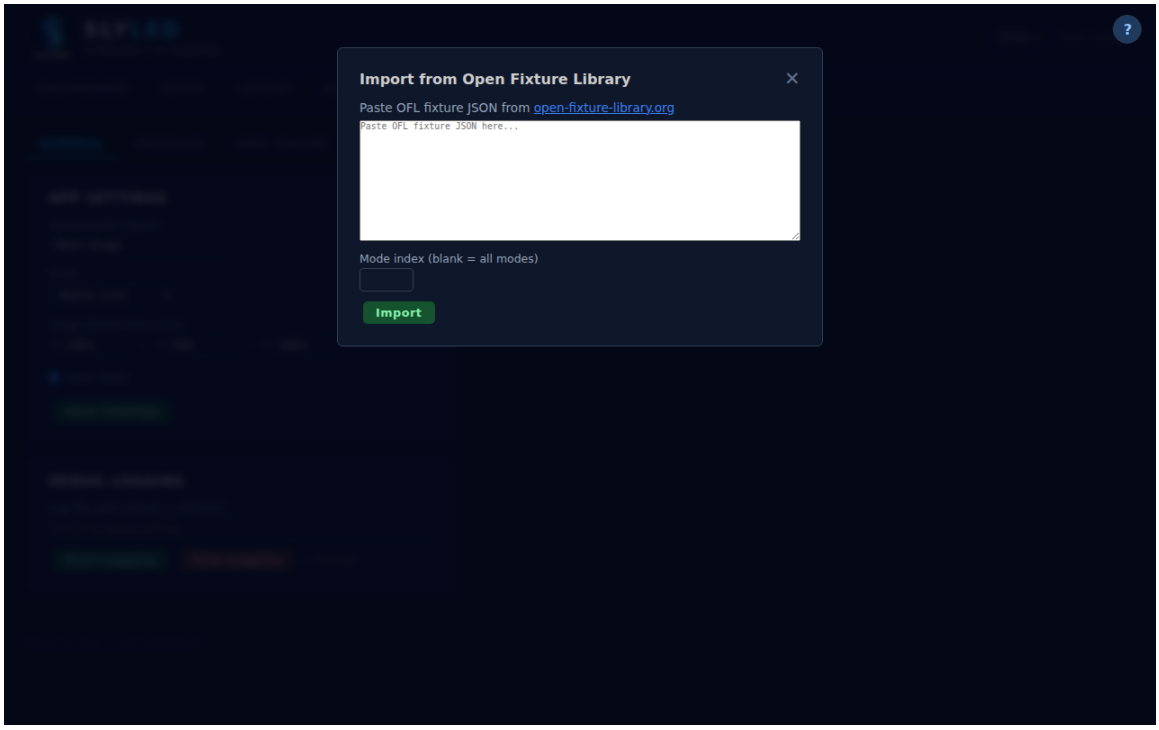
Profile editor — channel table with type dropdowns



Capability range editor for a channel

Importing from Open Fixture Library

Click "Import OFL" to paste fixture JSON from open-fixture-library.org. The importer converts OFL's capability model to SlyLED format, handling multi-mode fixtures, 16-bit channels, and color detection automatically.

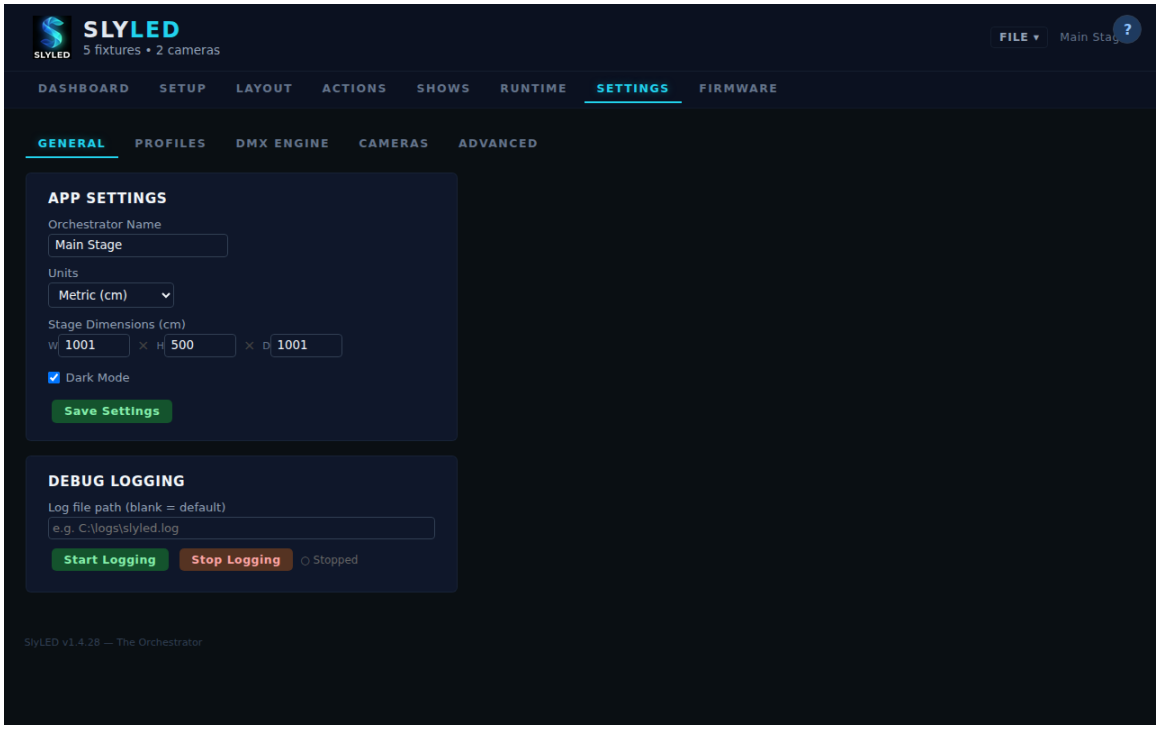


Open Fixture Library import modal

Import/Export Bundles

Export your custom profiles as a JSON bundle file for backup or sharing. Import bundles to add profiles from other installations.

9. Settings & Configuration

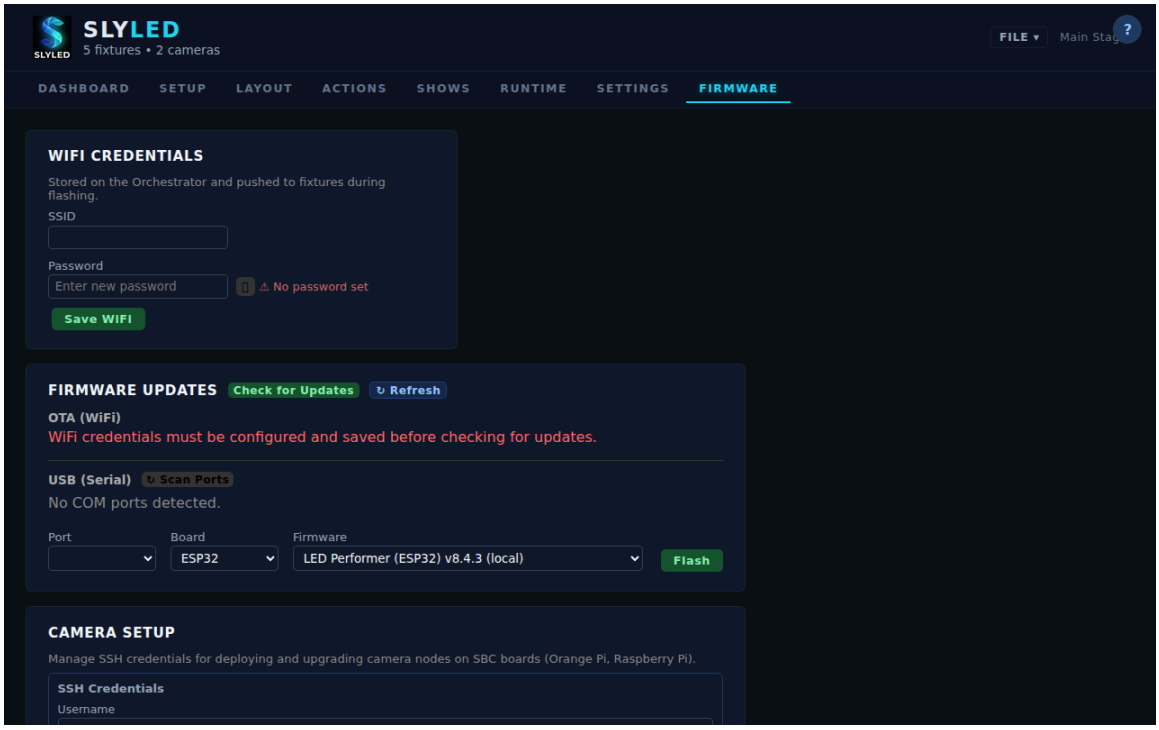


Settings tab — app configuration and DMX output

The Settings tab contains:

- Orchestrator name and units
- Stage dimensions
- Dark mode toggle
- Logging control
- Configuration export/import
- Show export/import
- DMX output settings (Art-Net/sACN, frame rate, universe routing)
- Fixture Profile Library
- Factory reset

Firmware



Firmware tab — board detection and flash controls

The Firmware tab provides:

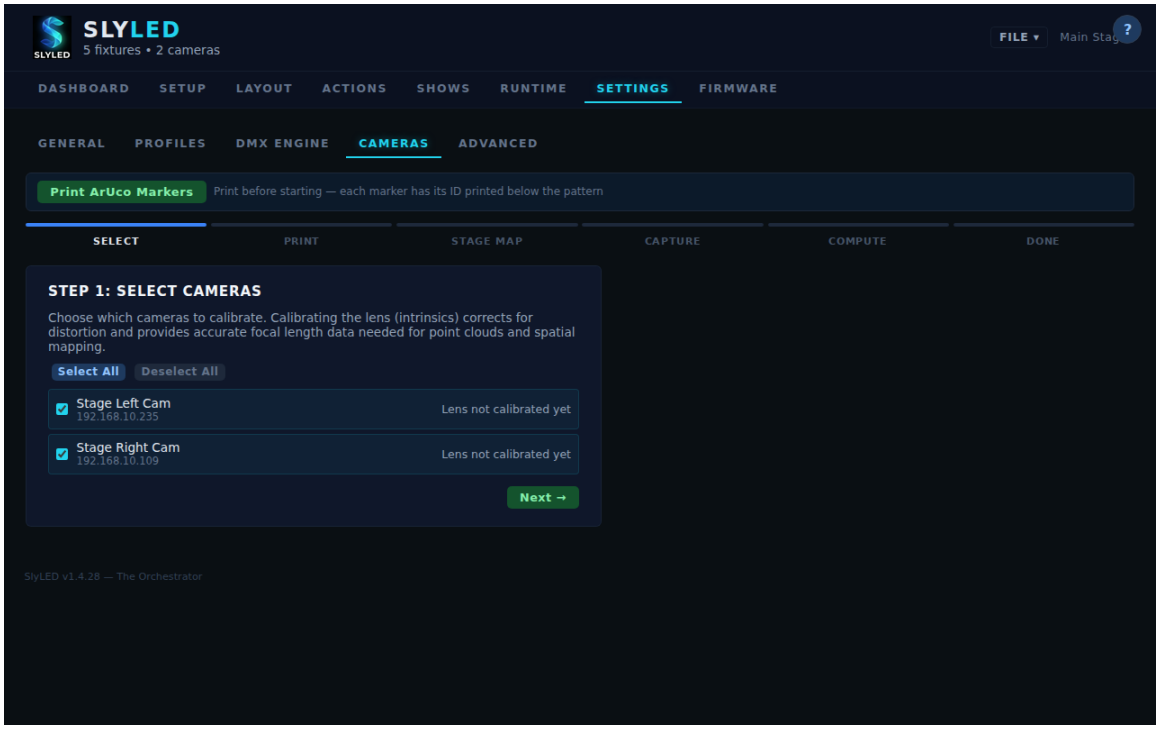
- COM port detection with board type identification
- Serial version query
- WiFi credential management
- One-click firmware flashing for ESP32, D1 Mini, and Giga R1 boards

10. Camera Setup & Calibration

Adding Camera Nodes

Camera nodes are Orange Pi or Raspberry Pi SBCs running the SlyLED camera firmware. They capture snapshots for calibration, beam detection, depth estimation, and tracking.

To add a camera:
1. Connect the camera node to the same network as the orchestrator
2. Go to Setup tab and click "Discover" — camera nodes respond to UDP broadcast
3. Or manually add via Settings > Cameras with the camera's IP address
4. Each USB camera sensor appears as a separate fixture that can be placed in the Layout



Settings > Cameras — camera list with calibration status

ArUco Marker Calibration

ArUco markers are used for camera intrinsic calibration (lens parameters).

Step-by-step:
1. Print ArUco markers: Settings > Cameras > "Print ArUco Markers" (6 markers, 150mm each)
2. Place printed markers on flat surfaces visible to the camera at various angles
3. Click "Capture" 5+ times from different positions — each capture detects markers automatically
4. Click "Compute" — the system runs cv2.calibrateCamera() to determine fx, fy, cx, cy

5. RMS error < 2.0 indicates good calibration

All ArUco detection runs on the orchestrator PC (not the camera hardware), so any camera that can serve JPEG snapshots works — even low-end Raspberry Pi boards.

Stage Map — Camera Pose

The Stage Map determines each camera's 3D position and orientation in stage coordinates using solvePnP with ArUco markers at known stage positions.

Step-by-step:
1. Place 3+ ArUco markers on the stage floor at known positions (measure with tape)
2. Enter each marker's stage coordinates (X, Y, Z in mm) in the wizard
3. Click "Compute Stage Map" — the system detects markers and solves for camera pose
4. Result: camera position in stage mm, floor-plane homography, reprojection error

The homography maps any pixel to stage floor coordinates (X, Y at Z=0), enabling beam detection in real-world millimeters rather than pixel space.

Mover Calibration Wizard

Moving head fixtures need calibration to map DMX pan/tilt values to real-world positions.

The wizard runs automatically:
1. Discovery: Spiral search to find the beam in the camera's field of view
2. Adaptive settle: Waits 0.8-2.5s per move, verifies beam has stopped (double-capture)
3. BFS mapping: Explores visible region, stops at boundaries when beam leaves camera view
4. Grid build: Creates a bilinear interpolation grid for fast pan/tilt lookup
5. Boundary verification: Flash on/off at edges to confirm true visibility limits

After calibration, the system can aim any fixture at any stage coordinate using grid_inverse().

11. 3D Environment Scanning

Point Cloud Capture

The environment scan creates a 3D point cloud of your venue using depth estimation.

Step-by-step:
1. Position and calibrate 2+ cameras (Stage Map required)
2. Go to Settings > Space > "Start Scan"
3. Each camera captures a snapshot, the CV Engine runs Depth-Anything-V2 locally
4. Point clouds from each camera are transformed to stage coordinates and merged
5. Floor is auto-detected and normalized to Z=0

Result: 5,000-20,000 colored 3D points representing your venue geometry.

Surface Detection

After scanning, click "Analyze Surfaces" to run RANSAC-based detection:

- Floor: Horizontal plane detection (normal near vertical)
- Walls: Vertical planes along stage boundaries
- Obstacles: Clustered points that don't belong to floor or walls (pillars, furniture)

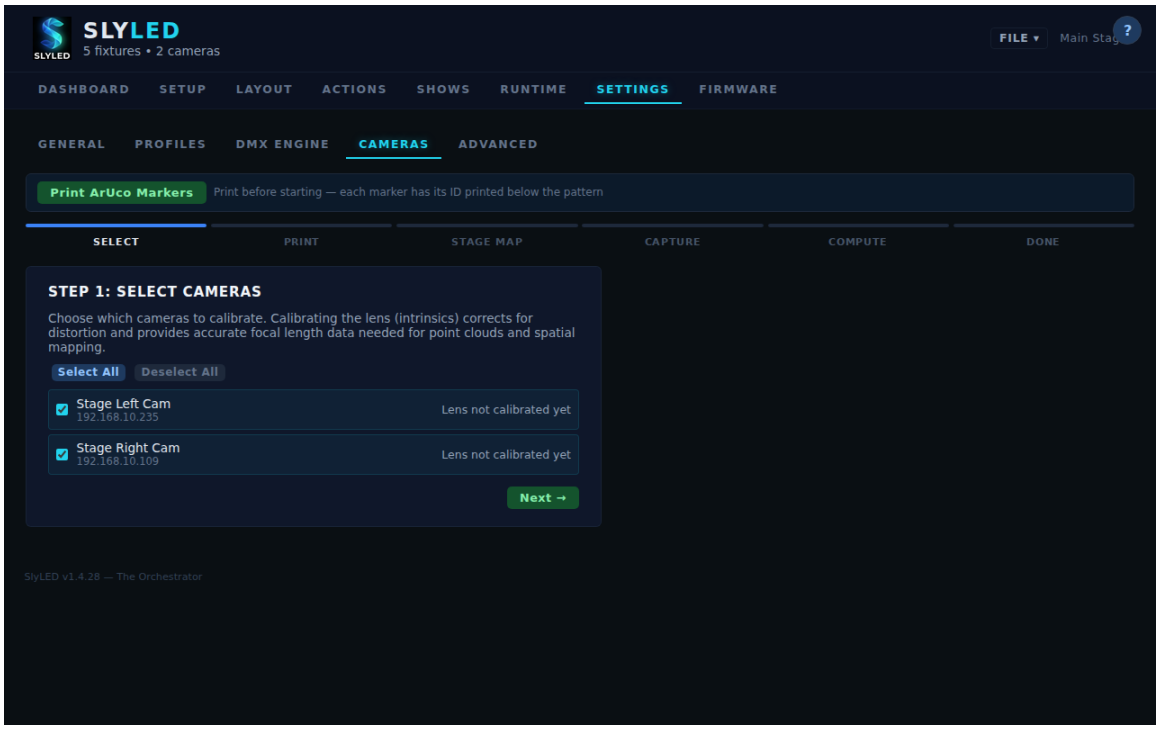
Detected surfaces are used by the calibration system to predict where beams land and by the structured-light refinement to verify 3D accuracy.

CV Engine

All computer vision processing runs on the orchestrator PC, not the camera hardware. The CV Engine wraps three model pipelines:

- Beam Detection: Fast color-filtered bright spot detection (<100ms)
- Depth Estimation: Depth-Anything-V2 monocular depth (1-2s on x86)
- Object Detection: YOLOv8n for person/object detection (100-200ms)

Camera nodes only provide JPEG snapshots. This means any camera works — even a Raspberry Pi 3 that can't run ONNX models locally.



CV Engine status — model loading state in Settings

12. Stereo 3D & Light Mapping

Stereo Triangulation

With 2+ calibrated cameras, the stereo engine triangulates 3D points from pixel observations. Each camera converts a pixel to a 3D ray; the intersection of rays from different cameras gives the 3D position in stage mm.

Use cases:
- Verify fixture positions by observing them from multiple cameras
- Track moving objects (people) in 3D
- Cross-validate depth estimation against triangulated ground truth

Accuracy depends on camera baseline (distance between cameras) and calibration quality. Typical error: 20-100mm at 3-5m distance with 60° FOV cameras.

Per-Fixture Light Maps

A light map records where each moving head's beam lands at every pan/tilt position in real stage coordinates.

Step-by-step:
1. Calibrate the fixture (mover calibration wizard)
2. Click "Build Light Map" on the fixture
3. The system sweeps the beam across the visible area in a grid pattern
4. At each position: detect beam pixel, convert to stage mm via homography
5. Result: (pan, tilt) → (stageX, stageY, stageZ) lookup table

Inverse lookup: given a target position on stage, find the exact pan/tilt to aim there. Uses inverse-distance weighted interpolation of the 4 nearest samples.

Beam-as-Structured-Light

During calibration sweeps, each beam position that hits a surface provides a definite 3D contact point. These points refine the environment model:

- Floor height correction: Beam contacts adjust the detected floor Z value
- Wall boundary extension: Contacts outside known wall boundaries expand the model
- Surface verification: Predicted ray-surface intersections are compared with observed positions

This is more accurate than depth estimation alone because the beam position is precise (bright centroid) and the ray direction is known (from pan/tilt geometry).

13. Project Files

Export

File > Export saves your entire project as a .slyshow file containing:

- All fixtures, children, layout positions, and stage dimensions
- Actions, spatial effects, timelines, and show playlist
- DMX settings, fixture profiles, and calibration data
- Camera calibrations (intrinsics, stage maps, mover grids)
- Point cloud (gzip-compressed for portability)
- Light maps (per-fixture pan/tilt-to-stage lookup tables)

The project file uses schema version 2. Files from older versions (v1) import successfully — new spatial fields default to empty.

Import

File > Import loads a .slyshow file, replacing all current state:

1. All playback is stopped
2. Fixtures, actions, timelines, and objects are restored
3. Calibration data (homographies, grids, light maps) is restored
4. Point cloud is decompressed and loaded into the 3D viewport
5. Camera SSH passwords must be re-entered (not portable between machines)

This enables a workflow where a designer builds the show on their laptop, exports the project, and a runtime operator loads it on the show PC — all spatial data transfers automatically.

Spatial Data in Projects

Point clouds can be 10,000-20,000 points (1-3 MB uncompressed). The export pipeline gzip-compresses the point array to ~200-500 KB. On import, the compressed data is automatically decompressed and loaded.

Light maps (typically 200-300 samples per fixture) are stored alongside mover calibration data.

14. System Limits

Resource	Limit	Notes
Children (performers)	8 max	Protocol constant
Strings per child	8 max	ESP32 supports up to 8 GPIO pins
LEDs per string	65535 max	uint16_t addressing (protocol v4)
Steps per runner	16 max	LoadStepPayload array limit
Bake segments per fixture	64 max	Supports PT move time slices
Bake frame rate	40 Hz	Art-Net output rate
DMX universes	Unlimited	One Art-Net packet per universe
DMX channels per universe	512	DMX-512 standard
Pan/tilt resolution	8 or 16 bit	Per profile channel definition
Preview resolution	1 fps	1 color per string per second
NTP sync offset	5 seconds	GO command sent with future epoch

15. Troubleshooting

3D Viewport Not Rendering

Browser doesn't support WebGL. Use Chrome, Firefox, or Edge.

Performers Not Syncing

Children offline or on different network. Check Setup tab status.

Bake Error: "No fixtures"

Add and position fixtures in Layout before baking.

DMX Not Outputting

Check Art-Net engine is started (Settings → DMX Output → Start). Verify universe routing matches your bridge address.

Moving Heads Not Tracking

Verify fixture has a profile with panRange/tiltRange > 0. Check rotation is set (Edit fixture → Pan/Tilt fields).

Preview Shows No DMX Beams

Bake the timeline first. DMX preview requires baked data.

Factory Reset

Settings → Factory Reset clears ALL data including fixtures, effects, timelines, and profiles.

16. Examples

These worked examples walk through common workflows from start to finish. Each example includes the exact steps, configuration values, and screenshots showing what you should see at each stage.

Example A: Camera Tracking — Moving Heads Follow a Person

Make DMX moving heads automatically follow people detected by a camera.

Prerequisites

- At least one camera node online (Firmware tab -> deploy + verify)
- At least one DMX moving head fixture placed on the Layout tab
- Moving head profile configured with pan/tilt range
- Art-Net/sACN engine running (Settings -> DMX -> Start)
- Mover calibration completed (see Example C) for accurate aiming

Steps

1. Verify hardware - Open the Setup tab. Confirm movers show green status and camera nodes show online. If cameras are offline, check WiFi and deploy firmware from the Firmware tab.

2. Start camera tracking - Go to the Layout tab. Click a camera fixture to select it. Click the Track toggle button. The status bar shows "Tracking active." The camera node begins running YOLO person detection at 2 fps and creates temporal stage objects for each detected person.

3. Create a Track action - Go to the Actions tab. Click + New Action:
   - Name: Person Follow
   - Type: Track (last option in the dropdown)
   - Color: Pick the beam color (e.g. red for a spotlight)
   - Leave Target Objects empty (follows ALL detected people)
   - Cycle Time: 2000 ms
   - Check Fixed assignment for strict 1:1 mapping

4. Create a timeline - Go to the Shows tab. Click + New Timeline, name it "Person Tracking", set duration to 600s, enable Loop.

5. Start playback - Click Bake, then Start. The 40 Hz DMX playback loop begins.

6. Test - Walk in front of the camera. Within 2 seconds a pink person marker appears in the 3D viewport. The moving heads aim at you.

Assignment Behavior

People in View	With 2 Moving Heads

1 person	Both heads aim at the same person
2 people	One head per person (1:1)
3+ people (cycling)	Heads cycle through people every 2s
3+ people (fixed)	First 2 tracked, 3rd ignored

Example B: Mover Tracking with Spatial Effects

Make moving heads follow a virtual object sweeping across the stage. No camera or physical hardware required — this runs entirely in the emulator.

Part 1 — Stage and Fixture Setup

1. Set stage dimensions - Open the Settings tab. Under Stage, enter:
   - Width: 6000 mm, Height: 3000 mm, Depth: 4000 mm
   - Click Save

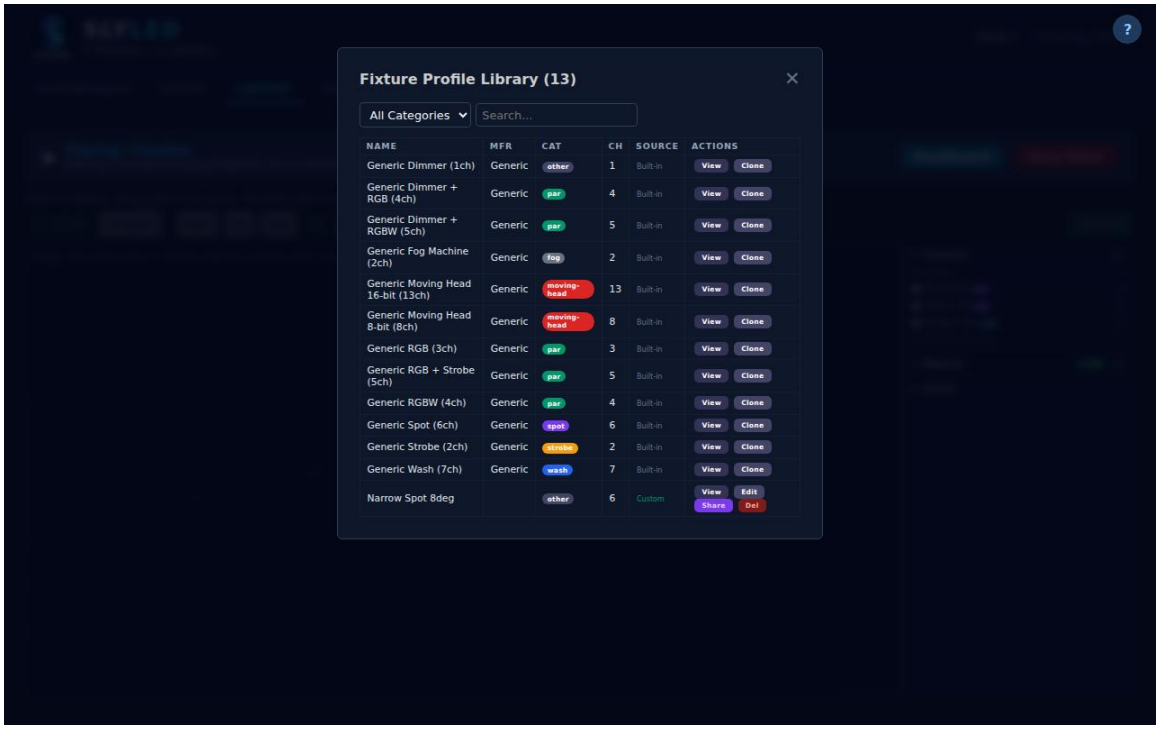
2. Create a DMX profile - Go to Settings -> Profiles -> New Profile:
   - Name: Narrow Spot
   - Beam Width: 8 degrees
   - Pan Range: 540, Tilt Range: 270
   - Channels: Pan (16-bit), Tilt (16-bit), Dimmer, Red, Green, Blue
   - Click Save Profile

Profile editor — Narrow Spot 8-degree beam

3. Add two movers - Setup tab -> + Add Fixture (twice):
   - Mover SL: Universe 1, Address 1, profile Narrow Spot
   - Mover SR: Universe 1, Address 14, profile Narrow Spot

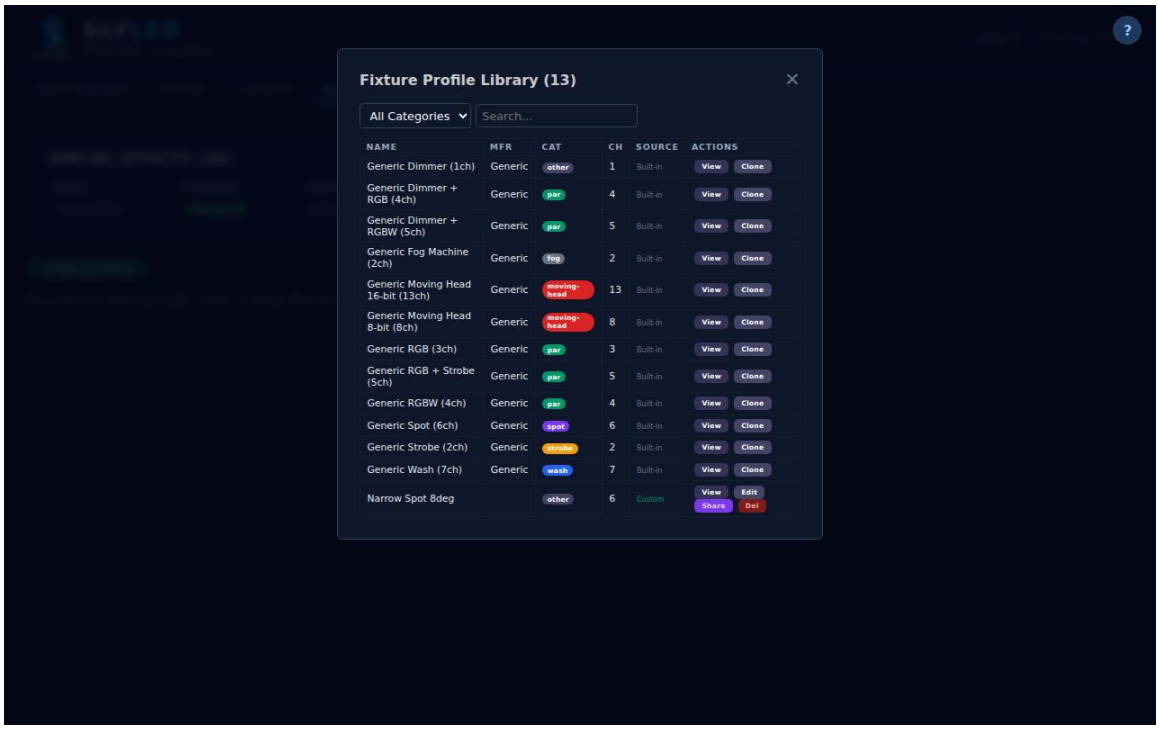
Part 2 — 3D Layout and Spatial Effect

4. Position movers on the truss - Layout tab -> drag each mover:
   - Mover SL: X=1500, Y=0, Z=2800 (stage left on truss)
   - Mover SR: X=4500, Y=0, Z=2800 (stage right on truss)
   - Switch to 3D view to confirm both are elevated and aimed downward.



Layout tab — 3D view with two movers on truss

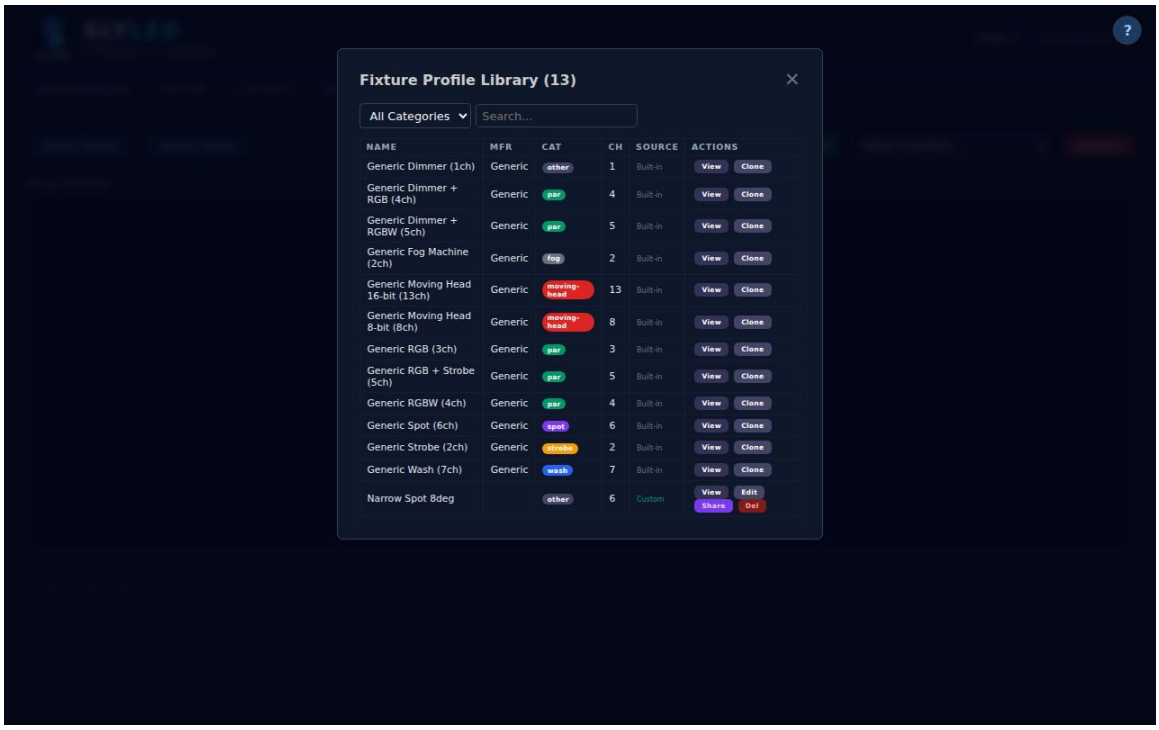
5. Create a spatial effect - Actions tab -> + New Action:
   - Name: Sweep Green
   - Type: Spatial Effect, Shape: Sphere, Radius: 800 mm
   - Color: Green (0, 255, 0)
   - Motion Start: X=1000, Y=2000, Z=0
   - Motion End: X=5000, Y=2000, Z=0
   - Duration: 8 seconds, Easing: Linear
   - This creates a green sphere that sweeps left to right in 8 seconds.



Actions tab — Sweep Green spatial effect

Part 3 — Timeline, Bake, and Playback

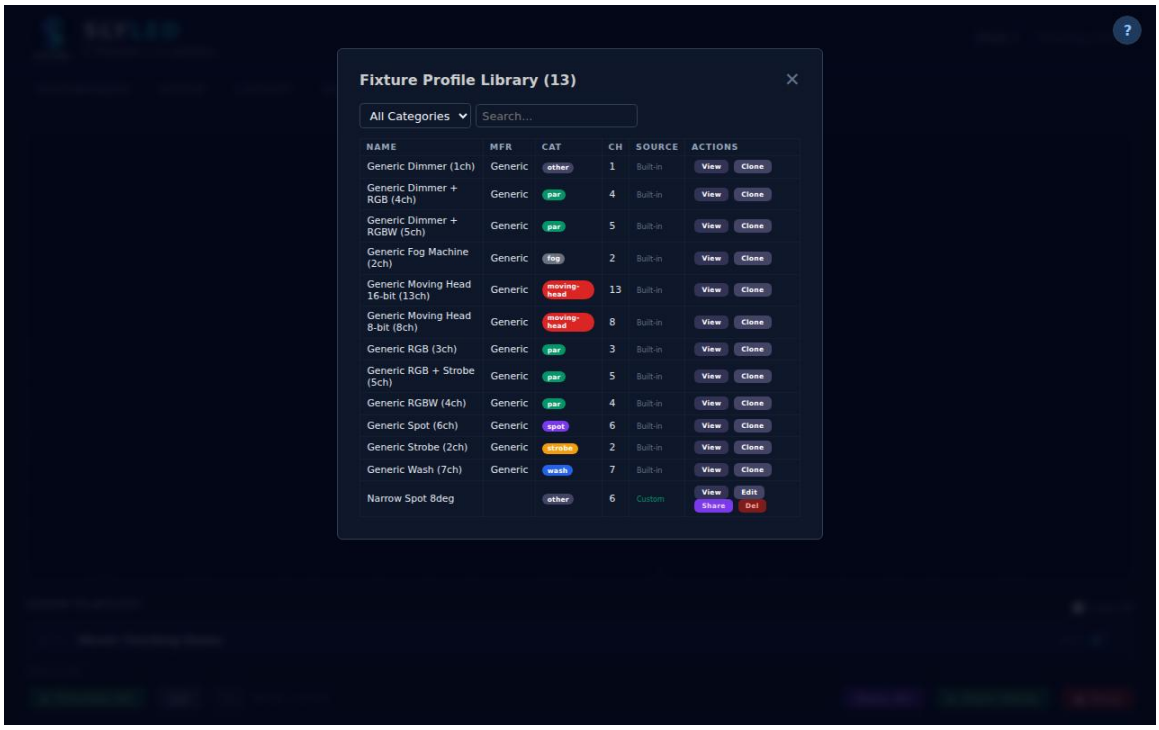
6. Create a timeline - Shows tab -> + New Timeline:
   - Name: Mover Tracking Demo, Duration: 20s, Loop: On
   - Add track targeting All Performers
   - Add clip referencing Sweep Green, 0s start, 8s duration



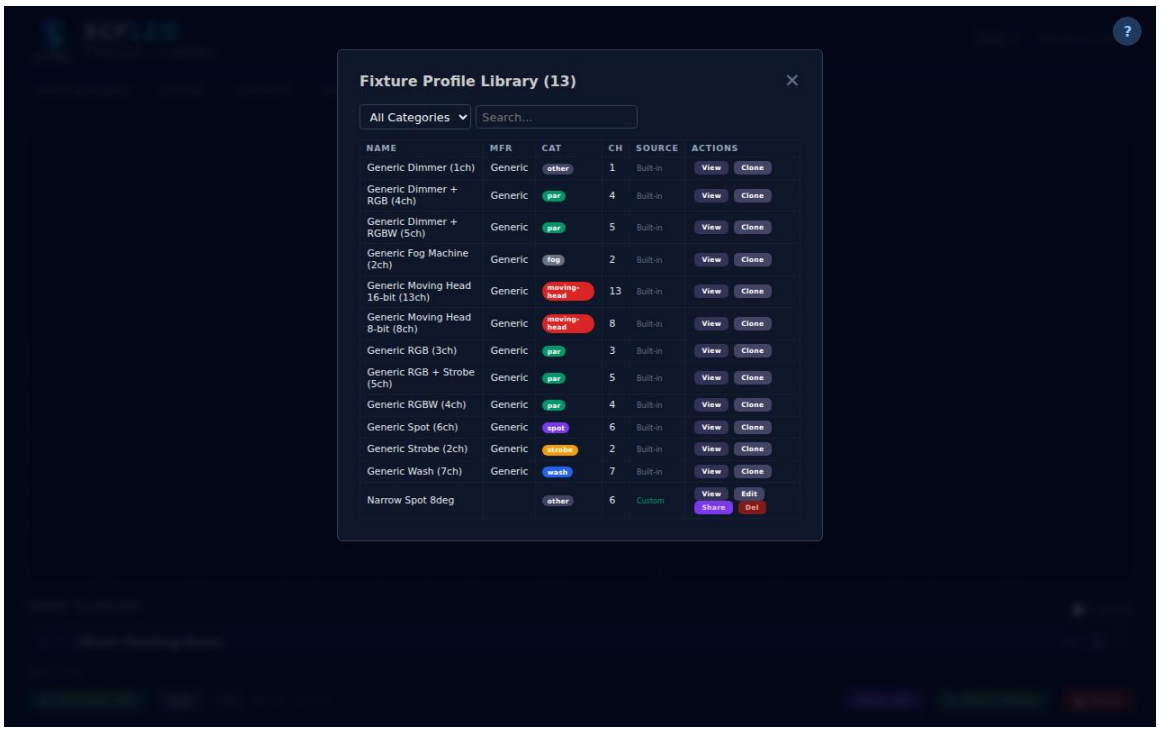
Shows tab — timeline with spatial effect clip

7. Bake the timeline - Click Bake. The engine computes per-fixture pan/tilt angles for each 25ms frame.

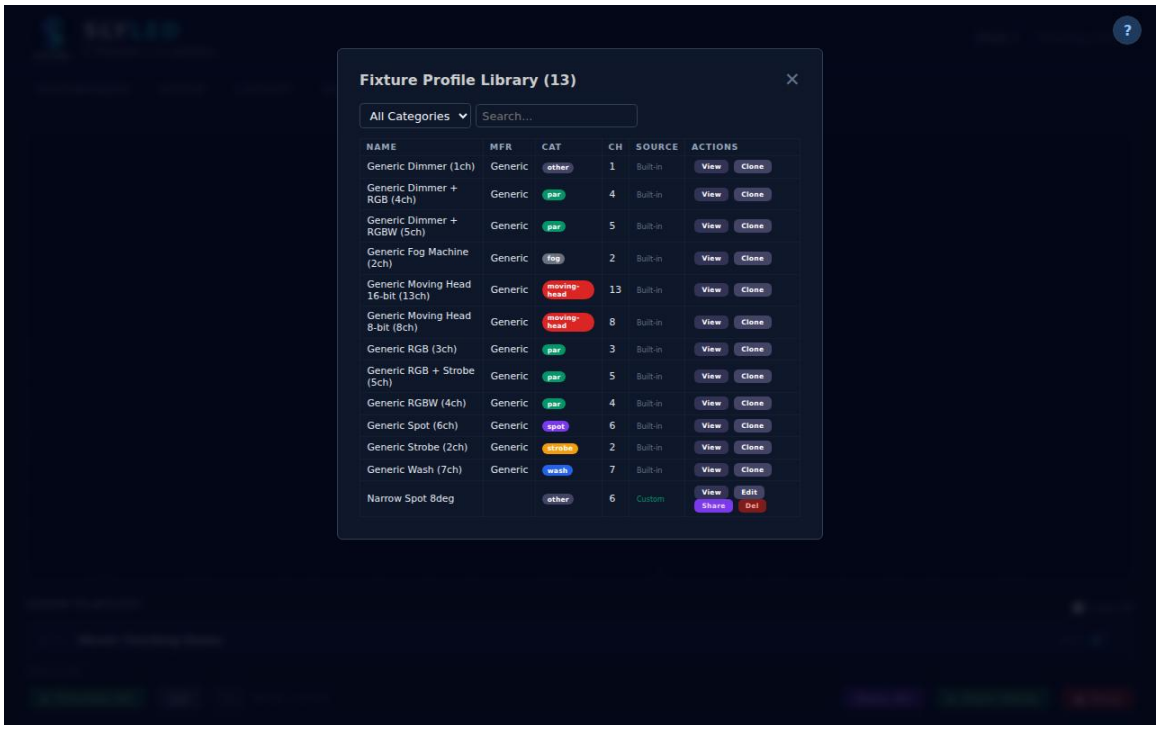
8. Start playback - Switch to Runtime tab. Click Start. Both beam cones animate in real-time, tracking the green sphere across the stage.



Runtime — beam cones at start position (T=0s)



Runtime — beams tracking mid-sweep (T=5s)



Runtime — beams at end position (T=10s)

What to look for:
- Both beam cones should be green (matching the effect color)
- The cones should move smoothly from left to right
- Beam opacity > 0 during the sweep, indicating active output
- If beam cones don't appear, ensure fixtures are positioned and the timeline is baked

Example C: Mover Calibration

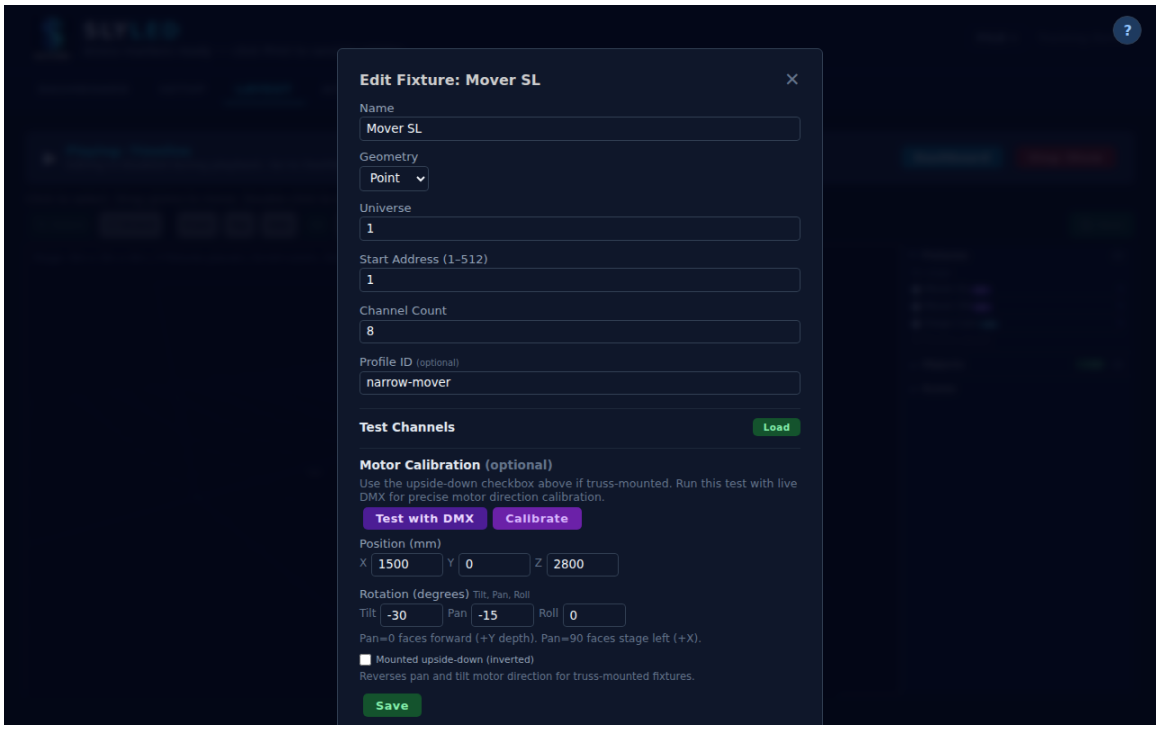
Calibrate a moving head so the system knows exactly where its beam lands for any pan/tilt position. This two-part process discovers the visible range then builds a light map for inverse lookup.

Prerequisites

- Camera node positioned on Layout with camera calibration complete (Example D)
- Moving head fixture placed on the Layout tab
- Art-Net engine running
- Dim ambient lighting (beam must be visible to camera)
- Beam aimed at the floor within the camera's field of view

Part 1 — Pan/Tilt Discovery and Grid Calibration

1. Open calibration - Layout tab -> double-click a moving head -> click Calibrate.



Calibration panel — fixture name and options

2. Choose beam color - Select a color that contrasts with your floor:
   - Green on dark floors, Magenta on light floors

3. Run discovery - Click Start Calibration. The system sweeps a coarse 8x5 grid across the full pan/tilt range. The camera watches for the beam on the floor.
   - Phase 1: Coarse grid scan (~40 positions)
   - Phase 2: Fine spiral refinement from detected position

Discovery in progress — coarse grid scan

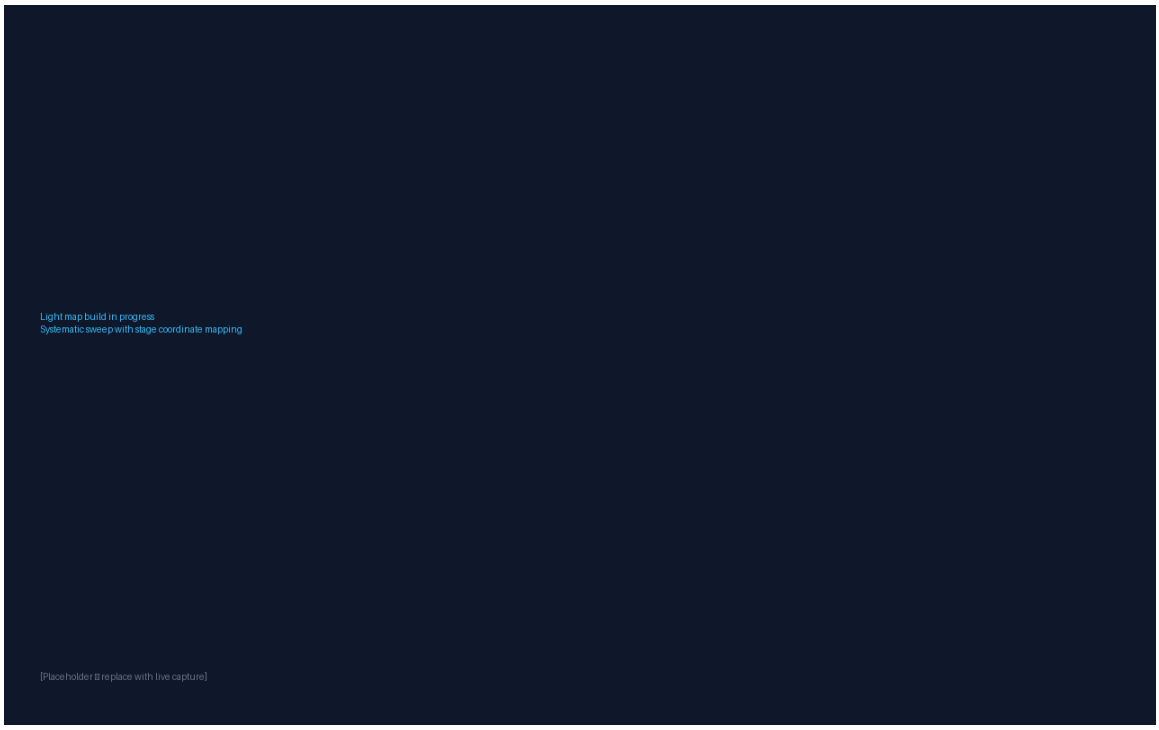
4. BFS mapping - The system explores the visible region in 4 directions from the discovered position. Collects up to 60 samples using adaptive settle times (0.8-2.5s per move). Mapping stops at boundaries.

5. Grid build and review - The calibration summary shows:
   - Sample count (aim for 30+)
   - Pan/tilt range (normalized 0.0-1.0)
   - Grid density

Grid calibration complete — sample count and range

Part 2 — Light Map (Stage to Pan/Tilt Lookup)

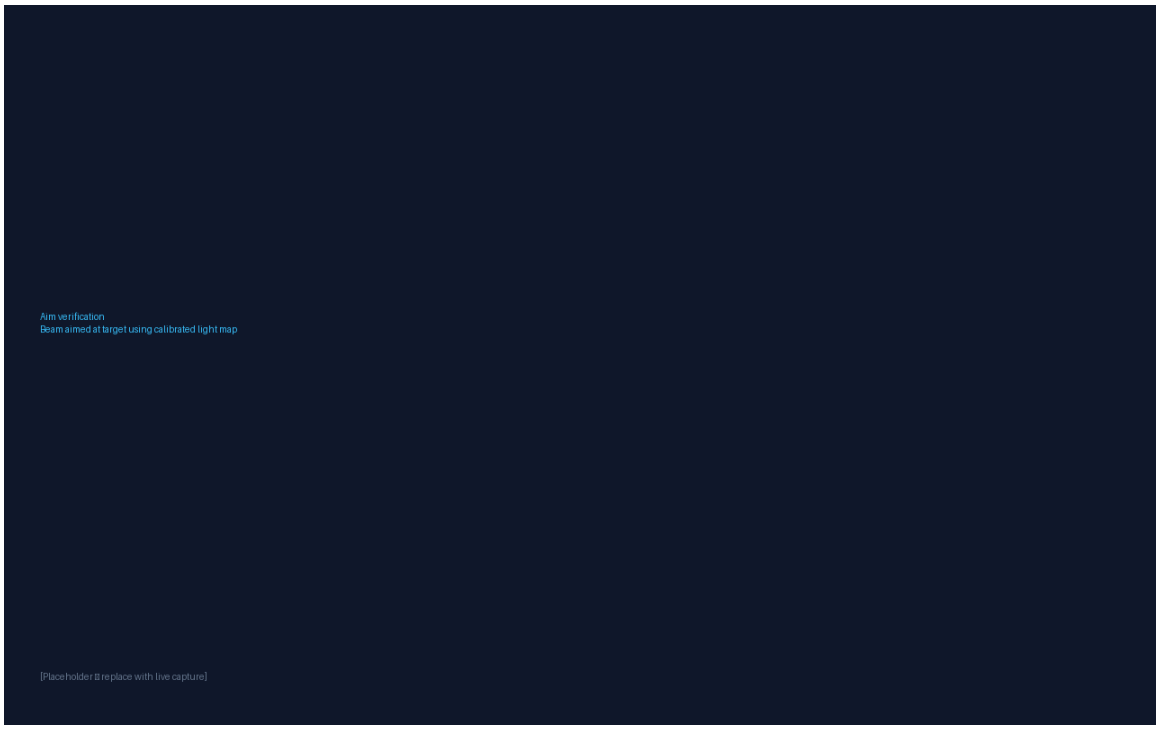
6. Build light map - Click Build Light Map. The system sweeps a 20x15 grid:
   - At each position: move, detect beam, convert pixel to stage mm via homography
   - Builds a (pan, tilt) -> (stageX, stageY, stageZ) lookup table
   - Typical time: 5-10 minutes for 300 samples



Light map build — systematic sweep in progress

7. Verify inverse lookup - Click Aim and enter a target stage position:
   - The system uses inverse-distance interpolation to find the pan/tilt
   - Verify the beam lands within 100-200mm of the target
   - Test 3-4 positions across the stage

8. Save - Calibration data is auto-saved with the fixture and included in project exports.



Aim verification — beam at target position

Manual Calibration (No Camera)

If automated calibration isn't available:

1. Double-click mover -> Manual Calibrate
2. Define 4-6 physical markers at known stage positions
3. Jog beam to each marker using pan/tilt sliders, click Record
4. Click Compute — the system fits a 3D affine transform
5. The transform extrapolates beyond calibrated points

Re-calibrate when: fixture moved, venue changed, pan/tilt range updated, or accuracy degrades.

Example D: Camera Calibration with ArUco Markers

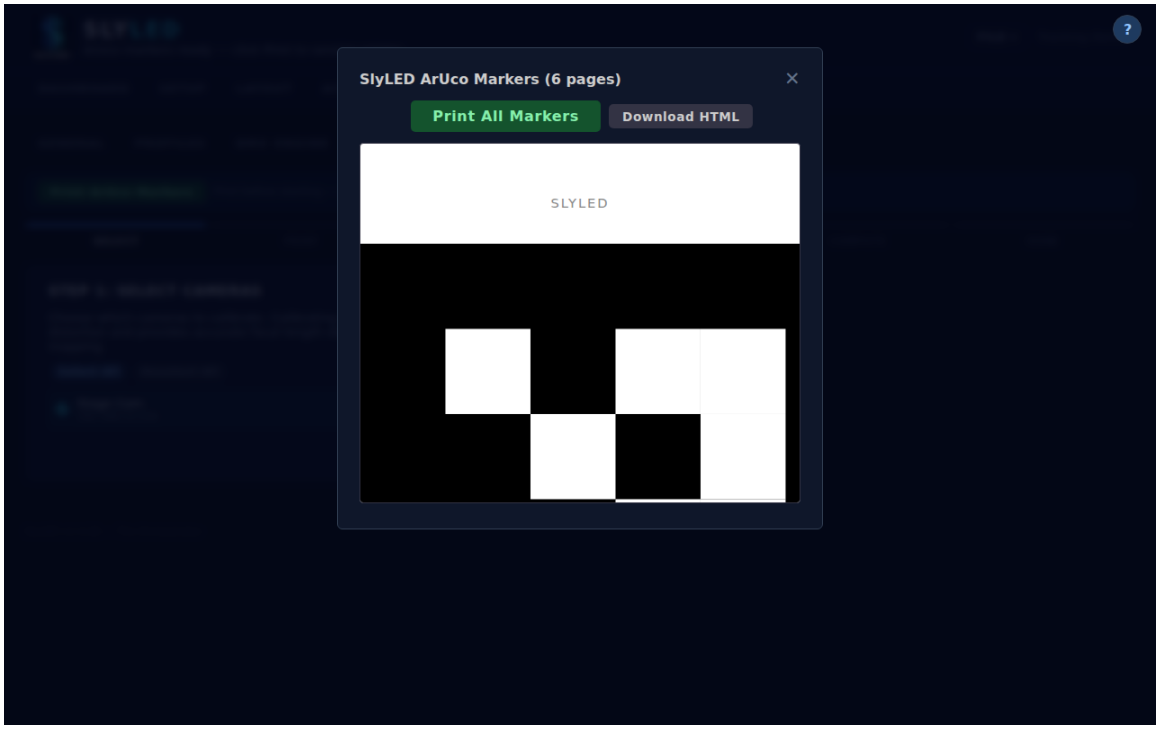
Calibrate a camera so pixel coordinates map to real stage positions. This is a prerequisite for beam detection, person tracking, and mover calibration.

Prerequisites

- Camera node online and reachable on the network
- Camera fixture registered (Setup -> Discover, or Settings -> Cameras)
- Camera placed on the Layout tab at its physical position
- Printer for ArUco marker sheet (A4/Letter)
- Tape measure for recording marker positions on stage

Part 1 — Prepare and Place Markers

1. Print ArUco markers - Settings -> Cameras -> Print ArUco Markers. A modal shows 6 ArUco 4x4 markers (IDs 0-5), each 150mm.
   - Print at 100% scale (no fit-to-page)
   - Card stock is more durable than regular paper

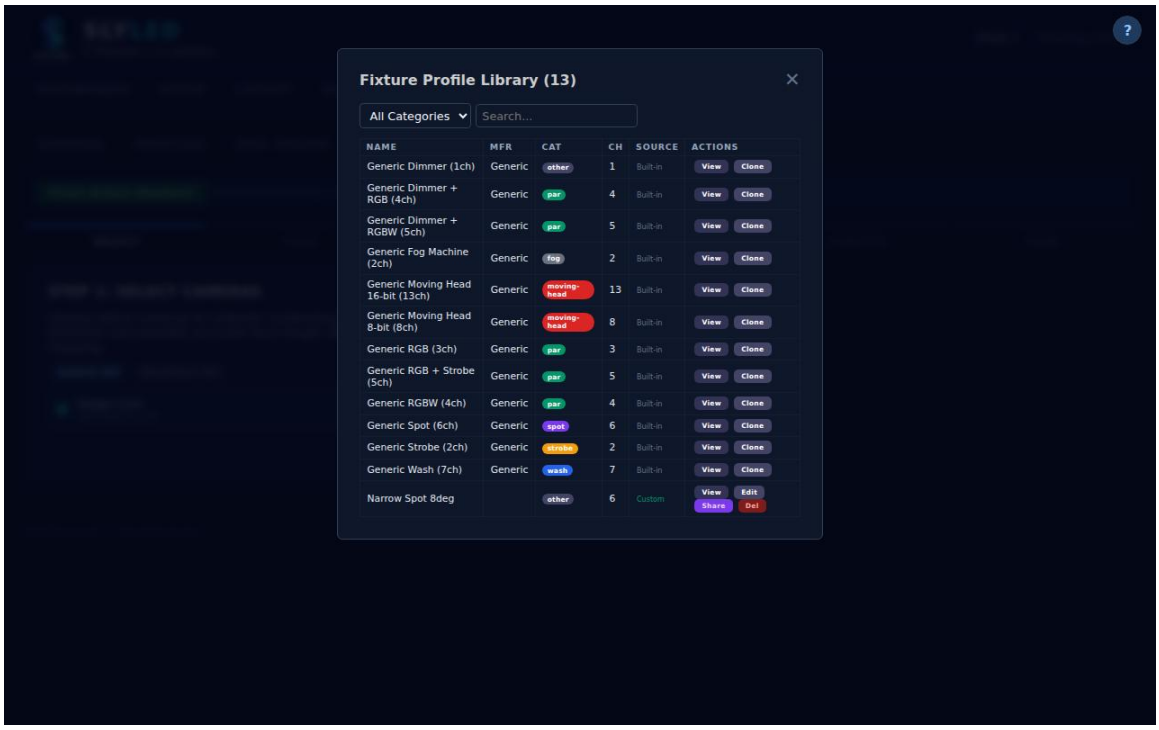


ArUco marker print dialog — 6 markers

2. Place markers on the stage floor:
   - Minimum 3 markers, recommended 4-6
   - Spread across the camera's field of view
   - Place flat on the floor (tilted markers reduce accuracy)
   - Measure each marker's X, Y position from the stage origin (mm)

Part 2 — Register and Position Camera

3. Register camera - Setup tab -> Discover, or Settings -> Cameras -> add IP.
   Each USB camera sensor appears as a separate fixture.



Camera configuration — list with IP and status

4. Position in 3D - Layout tab -> drag camera to its physical position.
   Set rotation to match aim direction. The camera frustum should cover the markers on stage.

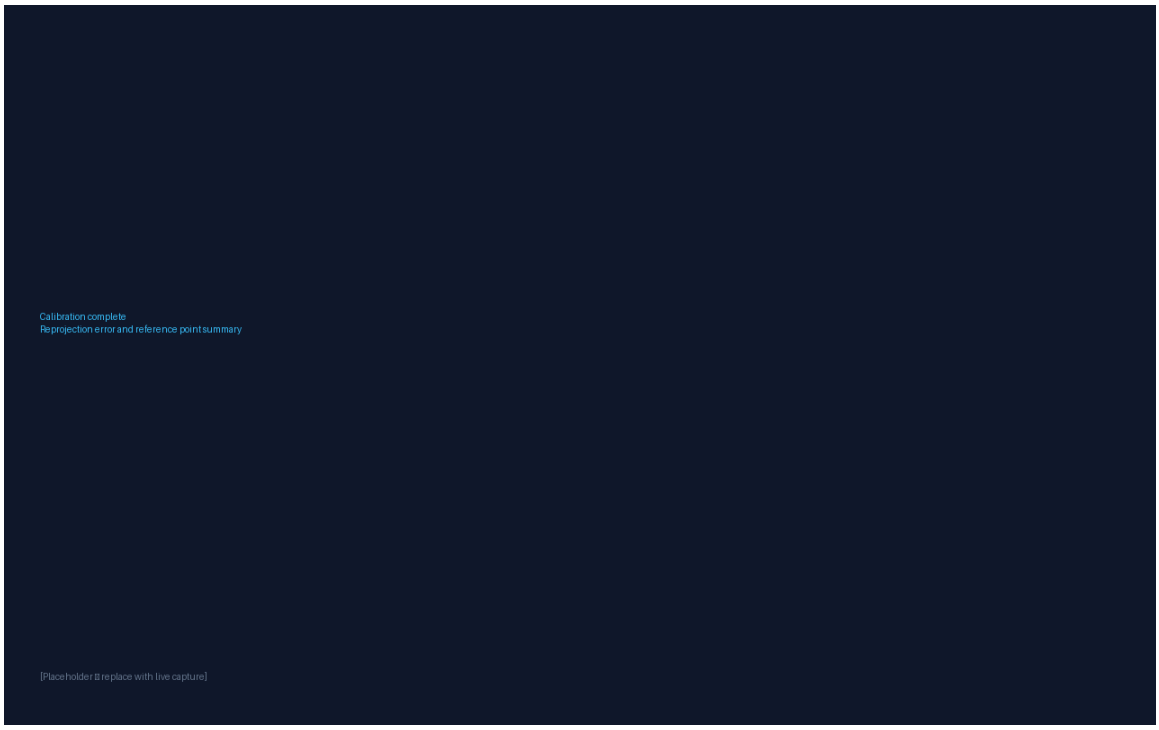
Part 3 — Calibrate and Verify

5. Run calibration - Layout tab -> click camera -> Calibrate:
   - Camera captures a frame, ArUco markers are auto-detected
   - For each marker: enter the real-world X, Y coordinates, click Record
   - Click Compute — builds a homography matrix (pixels -> stage coords)

ArUco markers detected — green overlays with IDs

6. Review results:
   - Reprojection error: < 10mm excellent, 10-20mm good, > 50mm re-try
   - Reference points: should match markers recorded
   - Coverage area: larger is better

7. Save - Click Save. The camera badge shows a green checkmark.
   All tracking and detection features now use stage coordinates.



Calibration complete — error and coverage summary

Tips

- Use large markers (> 10 cm) for reliable detection at distance
- Place markers at the extremes of the camera's view, not just center
- Avoid direct glare on glossy printed markers
- Re-calibrate when: camera moved, lens changed, or stage reconfigured

17. API Quick Reference

All endpoints are served from the Orchestrator at http://localhost:8080.

Fixtures

Method	Endpoint	Description
GET/POST	/api/fixtures	List / create
GET/PUT/DELETE	/api/fixtures/:id	CRUD
PUT	/api/fixtures/:id/aim	Set DMX aim point
POST	/api/fixtures/:id/resolve	Compute pixel positions

DMX Profiles

Method	Endpoint	Description
GET/POST	/api/dmx-profiles	List / create
GET/PUT/DELETE	/api/dmx-profiles/:id	CRUD
GET	/api/dmx-profiles/export	Export bundle
POST	/api/dmx-profiles/import	Import bundle
POST	/api/dmx-profiles/ofl/import-json	Import OFL fixture

Timelines

Method	Endpoint	Description
GET/POST	/api/timelines	List / create
POST	/api/timelines/:id/bake	Start baking
POST	/api/timelines/:id/start	Start playback
POST	/api/timelines/:id/stop	Stop playback

DMX Output

Method	Endpoint	Description
GET	/api/dmx/status	Engine status
POST	/api/dmx/start	Start engine
POST	/api/dmx/stop	Stop engine
GET	/api/dmx/fixture/:id/channels	Fixture channels with values

Calibration

Method	Endpoint	Description
POST	/api/cameras/:id/aruco/capture	Capture ArUco frame
POST	/api/cameras/:id/aruco/compute	Compute intrinsics

POST	/api/cameras/:id/stage-map	Compute camera pose
POST	/api/calibration/mover/:id/start	Start mover calibration
GET	/api/calibration/mover/:id/status	Poll calibration progress
POST	/api/calibration/stereo/calibrate	Build stereo engine
POST	/api/calibration/stereo/triangulate	Triangulate 3D point

CV Engine

Method	Endpoint	Description
GET	/api/cv/status	Model loading status
POST	/api/cameras/:id/detect	Object detection (local)
POST	/api/cameras/:id/depth	Depth estimation (local)
POST	/api/cameras/:id/beam-detect	Beam detection (local)

Space

Method	Endpoint	Description
POST	/api/space/scan	Start point cloud scan
GET	/api/space	Get point cloud
POST	/api/space/analyze	Detect surfaces
GET	/api/project/export	Export project (with spatial data)
POST	/api/project/import	Import project