Throwcast

Projector placement & simulation — user manual

Covers	The 3D editor, optics, blend arrays, simulation, the device library, cables, trays, racks and reports
Screenshots	Taken from the “Demo1” and “Demo2” projects described in §1.5
Units	Metres, degrees, lux — Y-up, right-handed

Contents

CHAPTER 1

Getting started

Throwcast is a projector placement and simulation tool. You build the venue, hang projectors in it, and it tells you what the light actually does — image size, coverage, pixel density, illuminance, and where two machines overlap.

1.1 What Throwcast is for

The working loop is short: put a screen in the room, put a projector in front of it, and read the numbers. Everything else in this manual — blend arrays, the heatmap, cables, trays, racks and reports — hangs off that loop.

Three properties are worth knowing before you start:

- **The scene is real.** Distances are metres, angles are degrees, and illuminance is lux. Nothing is nominal or “to scale”.
- **Nothing is baked.** Metrics and the heatmap recompute as you drag. There is no render step to wait for.
- **Derived beats stored.** Wherever the software can work something out from geometry — which trays connect, which projectors sit next to each other in a blend, which devices are in a diagram section — it does, rather than storing a link that could go stale.

1.2 Signing in

Throwcast opens on a sign-in card. Enter an email address and a password.

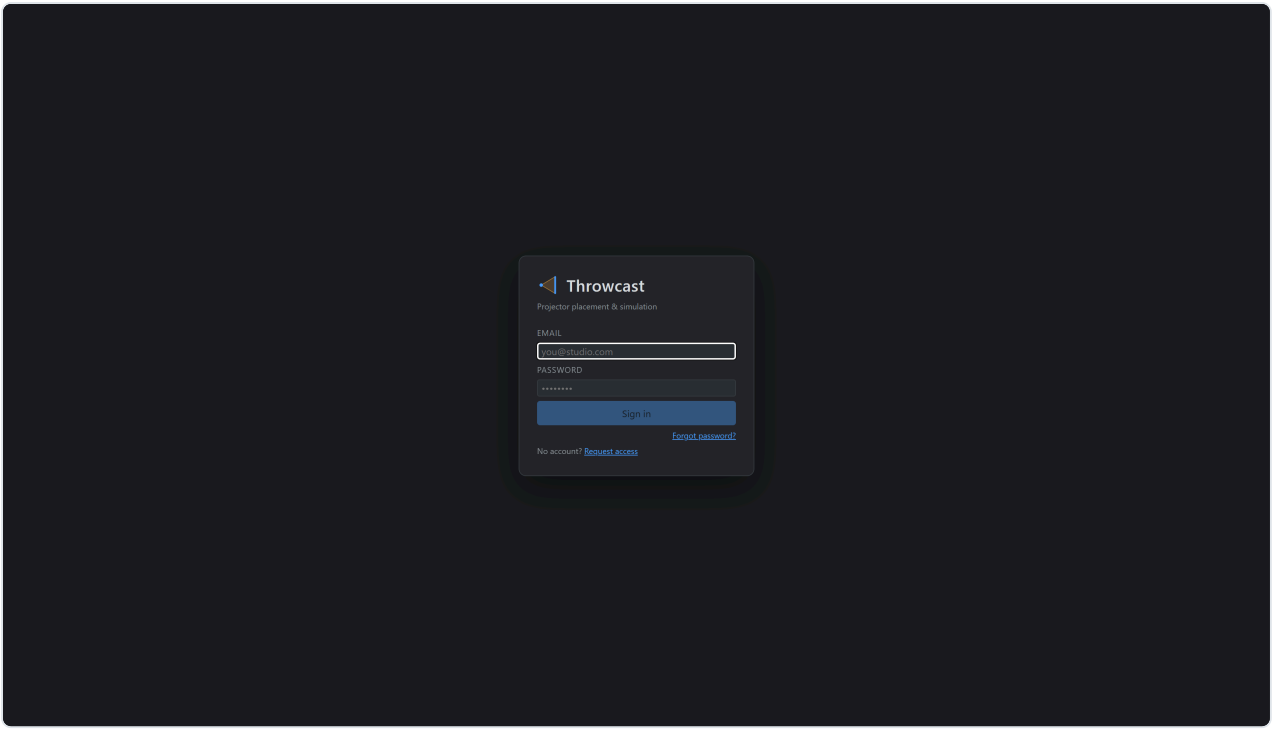


Figure 1.2 The sign-in screen. An address the server has not seen before creates an account, so there is no separate sign-up form.

> **NOTE**
>
> There is no password reset and no "sign in with…". An unknown email address *creates an account* rather than reporting an error — so a typo in your address signs you into a new, empty account rather than telling you the password was wrong. If your projects are missing, check the address first. (On a server where sign-ups have been closed, an unknown address is simply refused and you arrive by invitation instead.)

Arriving by invitation

An invitation email carries a single-use link that signs you straight in. Because that link is spent as soon as it is followed, the first thing you see is "**Choose a password**" — set one now so you can sign in again later.

The dialog will not be dismissed until a password exists: clicking outside it, pressing `Esc`, opening a new tab and reloading all leave it up, because dismissing it would leave you signed in with no way back once the session ends. It disappears for good the moment you save one.

1.3 Projects

After signing in you land on your projects. Each card is one project; open one by double-clicking it. "**New project**" asks for a name and creates an empty scene.

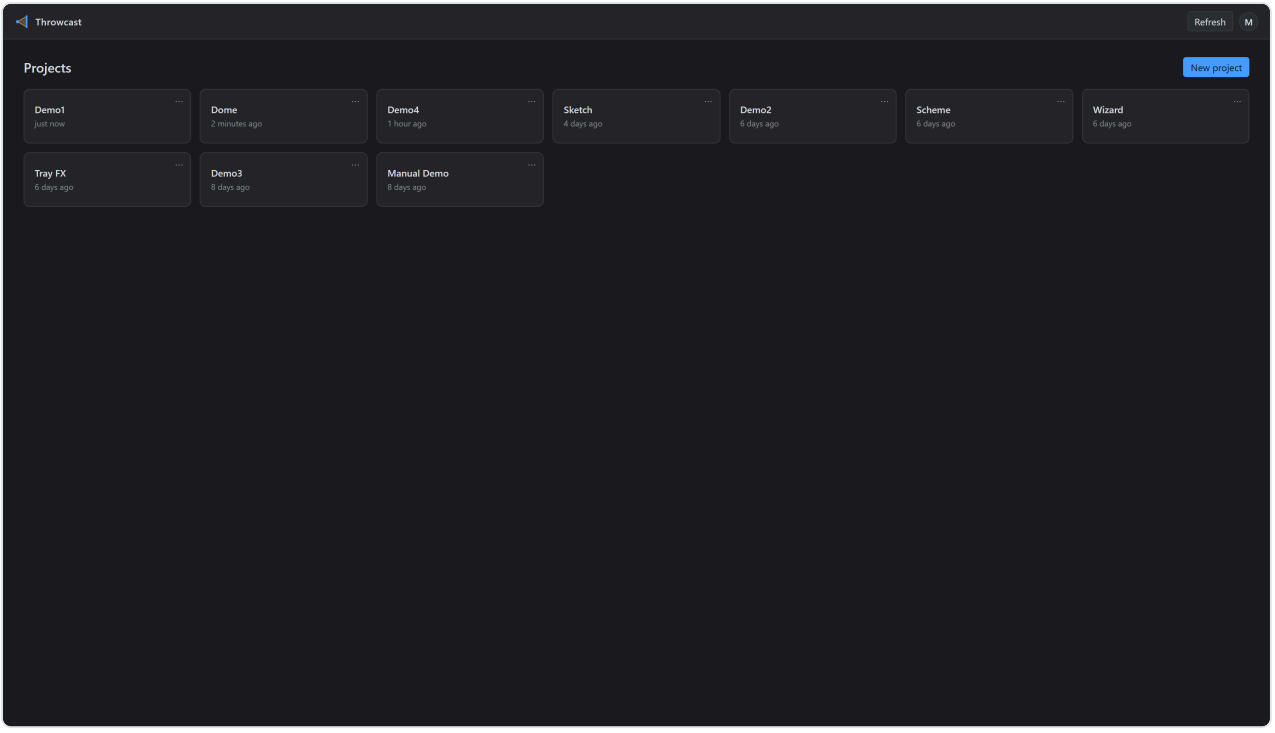


Figure 1.3 The projects screen: card grid, with "**Refresh**" and "**New project**". Rename and delete live on the cards themselves.

Inside the editor, the chip at the top left — "**‹ Demo1**" in the figures throughout this manual — takes you back here. Leaving a project unmounts the 3D viewport entirely, which is what frees the graphics memory a large venue was using.

1.4 Where your work is saved

Throwcast saves continuously to your account. The indicator at the top right of the editor reads "**All changes saved**" when everything has landed, and "**Unsaved changes…**" while a write is in flight. There is no Save command and no Save button.

1.5 The project in this manual

Figure 1.5 shows a small venue built so that there is something real to point at. It is worth reading once:

Table 1.1 The "Demo1" venue.

ELEMENT	WHAT IT IS
Room	A room object, 12 × 4 × 13.1 m — floor and three walls, open at the top
Curved screen	A curved surface, 12 m of screen bent on a 5.7 m radius, across the far end
Box / sphere / cone	Three objects standing on the floor in front of it, so the beams have something to fall across
Two Epson CO-FH01	3 000 lm, 1920 × 1080, each on its built-in 1.19 : 1 lens, both onto the curved screen
Trusses	Eight 1 m truss sections in a row, spanning the width of the room
Rack	A 22 U rack holding two players and a switch

Cable tray	One tray network — 16 m of run bent through four stretches
Four cables	Two HDMI, a player to each projector, and two Ethernet from the switch — all four routed through the tray

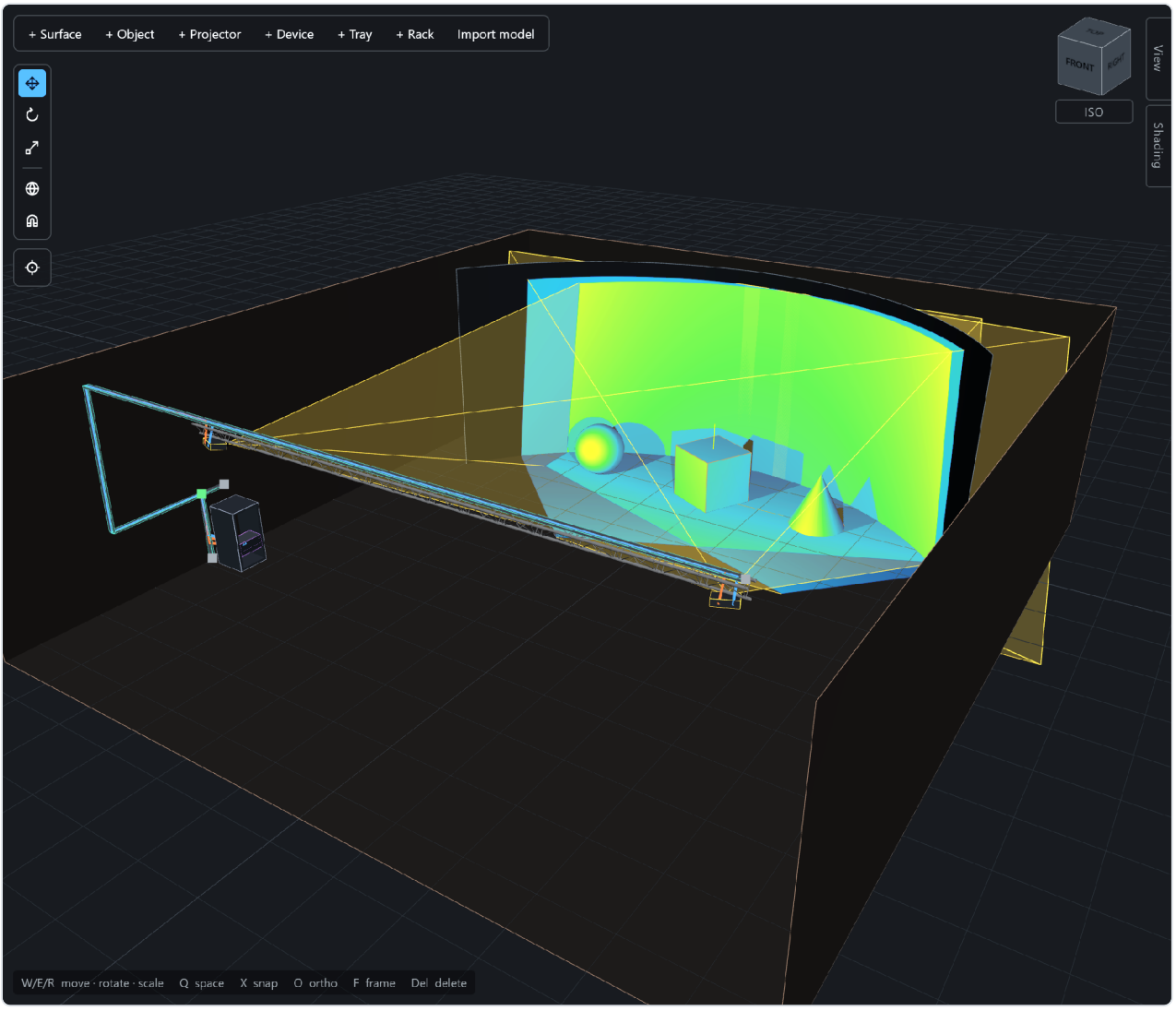


Figure 1.5 The demo venue, with the lux heatmap switched on — the green-to-blue wash is illuminance, not projected content. Both machines throw onto the curved screen at the far end, and the box, sphere and cone stand in the beams. The rack and the tray run are at the near end.

CHAPTER 2

The editor

One window, two side panels and a viewport. The rule that decides what goes where is simple: **the left panel is what exists in the project, the right panel is the properties of the one thing you have focused, and settings that belong to neither float over the viewport.**

2.1 Anatomy of the window

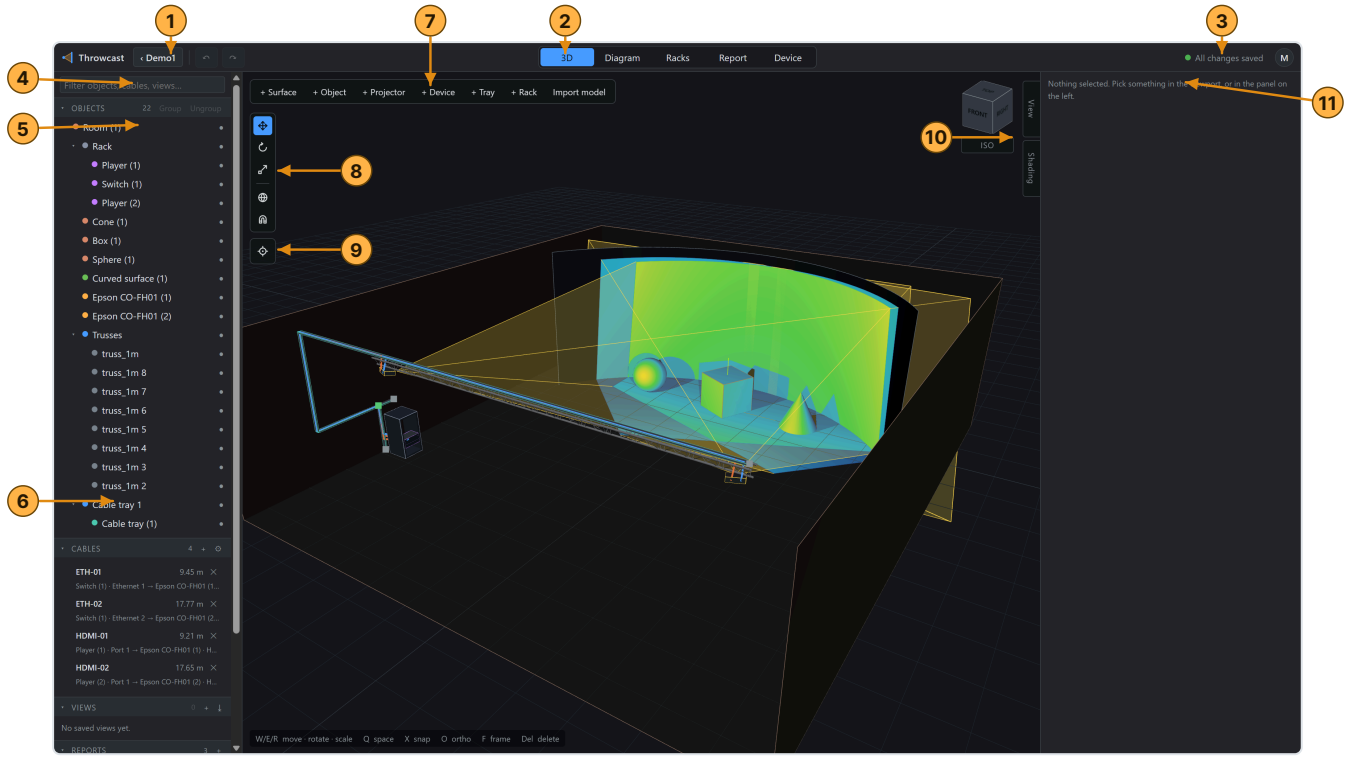


Figure 2.1 The editor.

1. **Project chip** — back to your projects. Beside it, undo and redo.
2. **Mode switcher** — 3D, Device, Diagram, Racks, Report, left to right in the order the work goes: the venue, then what stands in it, then how it is wired, then how it is racked, then what you hand over.
3. **Save state** — "All changes saved", or a spinner while writing. Beside it, a muted "**Manual**" link that opens this document in a new tab, and the account menu.
4. **Filter** — narrows every section of the browser at once.
5. **Objects** — the scene tree: what is in the venue, and what rides what.
6. **Cables, Views, Reports, Blend arrays** — the other four sections.
7. **Add cluster** — surface, object, projector, device, tray, rack, import.
8. **Gizmo rail** — move, rotate, scale; world ↔ local; snapping.
9. **Hover probe** — lux and pixel density under the cursor.
10. **Edge tabs** — the View and Shading panels.
11. **Inspector** — properties of whatever is focused.

Both side panels are draggable between 200 and 480 px, and their widths are remembered between sessions. They are one of only two things Throwcast keeps in your browser rather than in the project; the other is what the viewport draws (beams, footprints, cables, labels, grid).

2.2 The five modes

Table 2.1 What each mode is for.

MODE	WHAT IT IS
3D	The live venue with every overlay. Where placement happens.
Device	The library workshop: what a machine *is*, rather than where it is. It replaces both side panels.
Diagram	The same rig drawn as a 2D signal schematic. Fully controlled from the project — it is a second view of the same data, not a separate drawing.
Racks	Front elevations of your rack enclosures, with a palette of devices to mount.
Report	A **sheet** of paper you arrange content blocks on. One of those blocks is a frozen 3D view, and opening it binds the *same live viewport* to it, letterboxed to the block's rectangle.

Which mode you had open is not saved with the project — it is session state. Undo and redo work in every mode.

2.3 The project browser

Five collapsible sections in one scroll, under a sticky filter box.

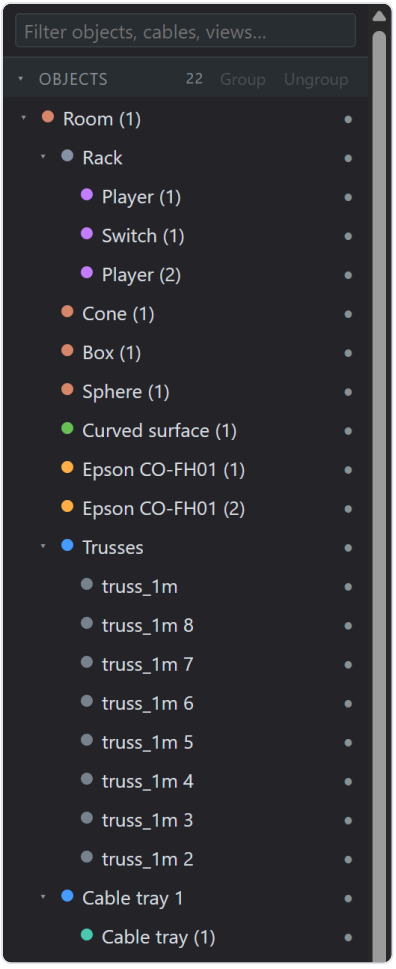


Figure 2.3 The browser. Note the nesting: the three devices sit under "**Rack**" because they are mounted in it, the eight truss sections under "**Trusses**" because they were grouped, and the tray network — one object however many stretches it runs — under "**Cable tray 1**". Where a blend array exists, its anchor carries an anchor badge.

- **Objects** — the scene tree. Drag to rearrange, group and ungroup, and toggle per-object visibility with the dot on the right.
- **Cables** — every cable with its live routed length, and a header "⚙" for project-wide settings (renumbering, default slack, port types).
- **Views** — saved cameras. "**+**" bookmarks the current one; "⤓" renders the current view to a 3840 px PNG.
- **Reports** — reports, with their pages nested underneath.
- **Blend arrays** — the parametric projector rigs, grids and rings alike.

A single click *focuses* a row. A double-click *opens* it, where that means something: a view flies the camera there, a report page enters report mode bound to it.

> **FILTERING**
> Typing in the filter box expands any section that has matches and dims the count on sections that have none. Clearing the box restores exactly the expansion state you left — the filter never overwrites it.

A FOLDED BRANCH STAYS FOLDED

Fold a group in the object tree and it is still folded after a trip to the Device tab, after a refresh, and the next time anyone opens the project — the fold is stored on the object, not in the session. Tidying a ninety-object venue down to a readable dozen rows is part of how the project reads, and it used to be thrown away by every reload.

It costs no undo step: a `Ctrl` + `Z` that unfolds a group instead of undoing the edit you were thinking about is one nobody can trust. A project opened read-only keeps its folds for the session instead, since folding is navigation rather than editing.

2.4 The inspector

The right panel shows the thing you have focused — the *last* one you picked, where you have picked several. For a projector it carries the transform, the device and lens, the throw ratio and shift, a **Metrics** block, and a **Tools** block.

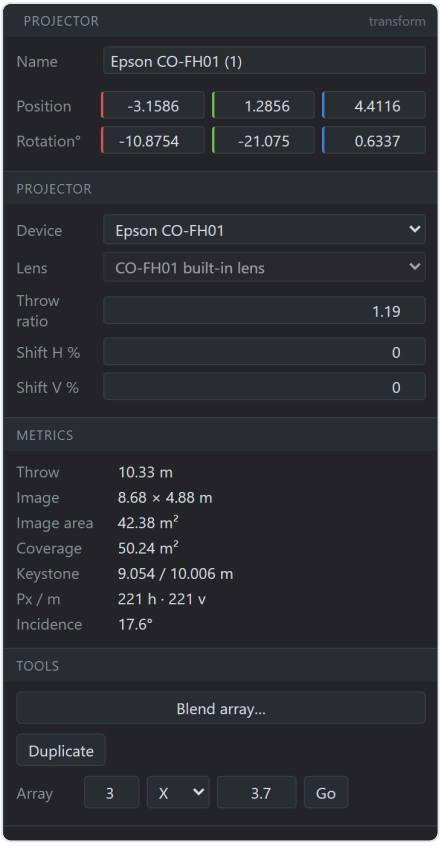


Figure 2.4 The inspector for a projector. §4.5 reads the Metrics block line by line.

INSTANCE VS MODEL

The inspector shows what is true of *this machine in this room* — which device, which lens, what throw ratio, what shift. It deliberately shows nothing that belongs to the *model*: no lumens, no body dimensions, no lens protrusion. Those live in the Device tab (Chapter 7), because changing them changes every instance.

A field edits everything you have selected

Select nine trusses, switch "**Section**" to Triangle, and all nine change. Type a "**Position Y**" and all nine hang at that height. The panel goes on showing the primary — the last one you picked — with a banded line at the top saying how many objects the next edit will reach, because the difference between editing one truss and editing nine is otherwise invisible in a panel that looks identical either way.

Two rules make that predictable:

- **Only the box you typed in travels.** Typing into Y moves nine trusses to that height and leaves each one where it stands along the run — the other two axes are not copied off the primary.
- **How far a field reaches depends on the field.** Position, rotation and scale are the vocabulary of every object, so a mixed bag of projectors and devices can be aligned to one height in one edit. Everything else reaches only objects of the primary's own type, because "**Length**" is a stock length on a truss and a run on a cable tray.

Values are absolute, never nudges — the number you typed reads the same on everything it reached — and the whole fan-out is one `Ctrl` + `Z`. "**Name**" is the one field that never fans out.

> **A DASH MEANS THEY DISAGREE**
>
> A field the selection does not agree about reads **—** rather than the primary's value: a box reading 2.00 while it writes to nine objects, six of which are 3 m, is the one lie a designer taking measurements off the panel cannot survive. Number boxes go blank with a dimmed placeholder, a segmented row lights nothing, a dropdown shows a blank entry, and a checkbox takes its indeterminate state.
>
> Typing over the dash sets them all — including to the primary's own value, which is exactly the case a naive "only write what changed" rule gets wrong.

A control whose result is *derived* from the object it lands on runs that derivation per object. Switching five screens to Cylinder conserves each screen's own width and works its radius out of that, giving five cylinders of five sizes rather than five copies of the primary.

Cables (§8.2) and report blocks (§11.3) work the same way, over their own selections.

2.5 Viewport overlays

Three floating clusters sit over the viewport, plus two edge tabs flush against its right side.

- **Add** (top) — the ways to put something in the scene.
- **Gizmo rail** (left) — move / rotate / scale, the world ↔ local toggle, and snapping.
- **Probe** (left, below the rail) — the hover readout.
- **View tab** — what the viewport draws: beams, footprints, cables, labels, measurements, cutters, grid.
- **Shading tab** — the simulation: off, lux heatmap, or content preview.

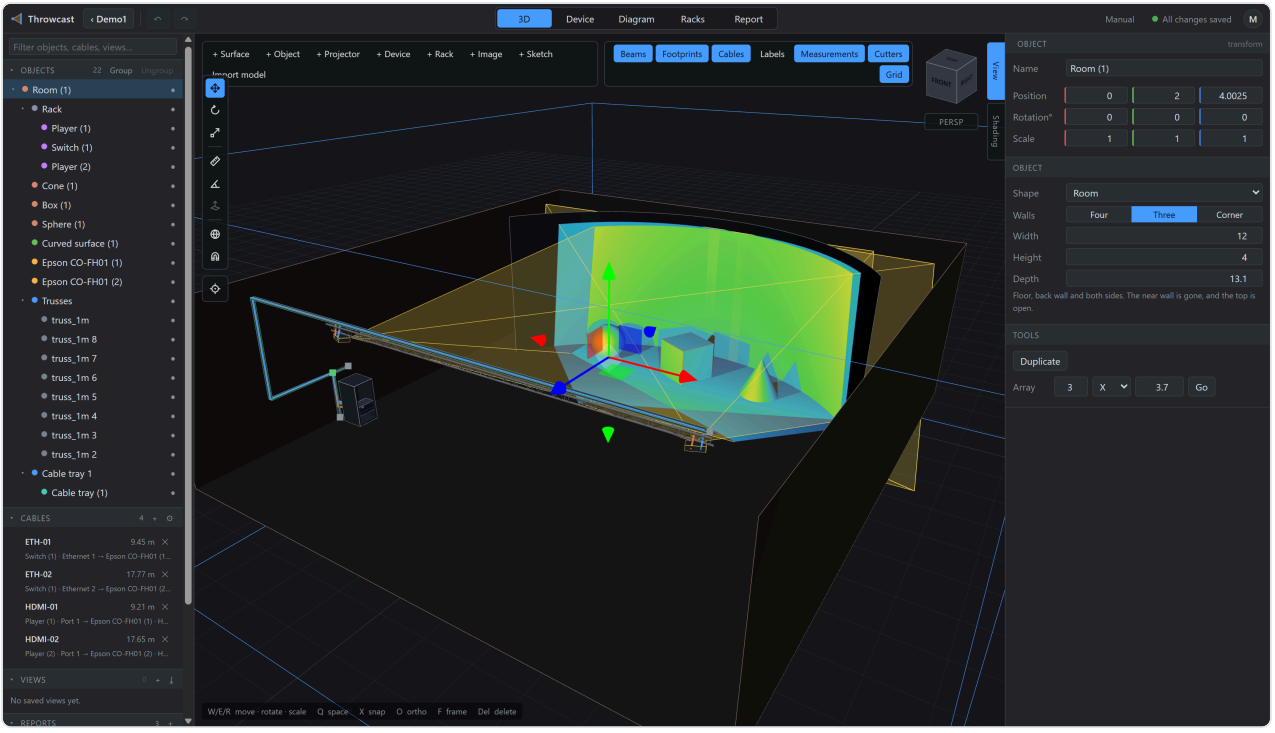


Figure 2.5 The View panel. These are display settings, not properties of anything, which is why they float rather than living in a panel. They are remembered in your browser, not in the project — and "**Labels**" is dark here because that is how a project opens.

The projection is not in here: **PERSP / ISO** is the plate under the ViewCube, where the rest of the camera lives, rather than a second switch for the same setting three panels away. And a sixth button, "**Drawing**", joins the cluster only while a report's 3D view is being framed — a drawing is a way of presenting rather than a way of working, so it belongs to the block that prints it (§11.5).

> **LABELS OPEN OFF**
>
> "**Labels**" is the one display overlay whose cost grows with the rig — one name chip per object, drawn over the geometry — so a venue would open under a drift of callouts hiding the things they name. It starts *off*; beams, footprints, cables, measurements and the grid start on. If you have ever touched a toggle yourself, you keep what you chose.

A lit edge tab means one thing only: *that panel is open*. It does not indicate which shading mode is active — the venue itself is repainted for that, which is hard to miss.

2.6 Navigating the viewport

Table 2.2 Getting around.

INPUT	DOES
Left-drag	Orbit around the target
Wheel	Zoom
Middle-drag	Pan — keeps working while an annotation tool is armed
F	Frame the selection

`M` / `A`	The ruler / the angle — press again to put it away
`O`	Orthographic ↔ perspective
ViewCube	Click a face for an axis view, an edge for 45°, a corner for isometric
PERSP / ISO	The plate under the cube: the projection (ISO being the orthographic one), the same switch as `O`. It moves no camera — the cube above it is what aims the view

CHAPTER 3

Building the venue

Before a projector means anything you need something for it to hit. Throwcast gives you two families of geometry — *surfaces*, which are screens, and *objects*, which are everything else — plus imported CAD. And when none of those is the shape you need, §3.7 onwards is the other way in: put the architect's drawing in the venue, scale it to life, and trace what you actually have to build.

3.1 Scene conventions

- **Y is up.** The floor is the XZ plane; height is Y.
- **Everything is metres**, rotations are degrees in the inspector.
- **FRONT is +Z** and RIGHT is +X, as the ViewCube shows.
- **A projector throws along its own −Z** — its forward. An unrotated projector faces the front wall.

3.2 Objects

"**+ Object**" asks which one first. It holds everything that stands in the venue without being a screen — five *shapes*, under *Rigging* the two truss sections and cable tray, and under *Cut* the cutter that opens doorways in them (§3.14).

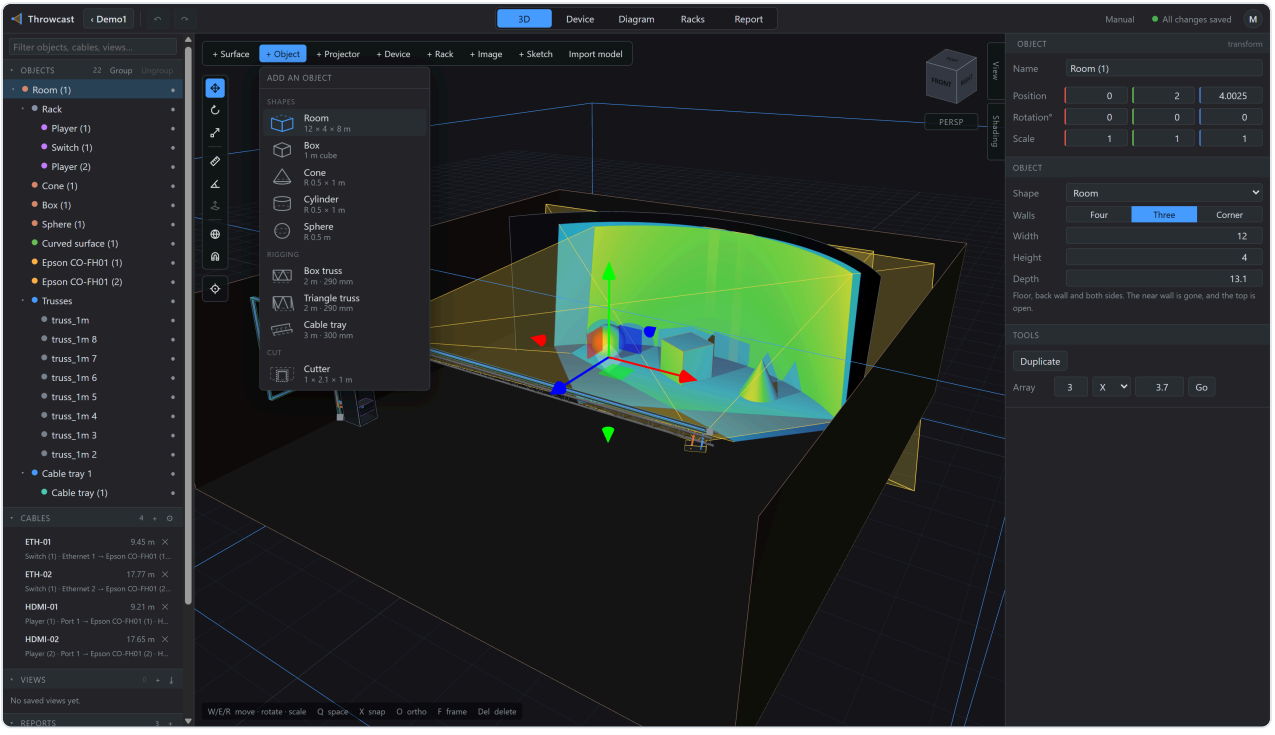


Figure 3.2 The nine rows and the size each one arrives at. Truss and tray had buttons of their own until they moved in here: they are the same kind of answer as the shapes — a thing that stands in the venue and is tuned in the inspector afterwards. The cutter sits under a heading of its own because it is the one row that takes geometry away rather than adding it.

Table 3.1 What "**+ Object**" offers.

ROW	IS	ARRIVES AS
Room	Floor and four walls, deliberately *no ceiling*	12 × 4 × 8 m
Box	A plinth, a riser, a flight case	1 m cube
Cone	A cone on its base	R 0.5 × 1 m
Cylinder	A solid column	R 0.5 × 1 m
Sphere	A globe	R 0.5 m
Box truss	Four chords in a square — the general-purpose stick	2 m · 290 mm
Triangle truss	Three chords, single chord on top — how truss hangs	2 m · 290 mm
Cable tray	One stretch to build a run from (§9)	3 m · 300 mm
Cutter	A body that dissolves what it overlaps — a doorway, a window, a bore (§3.14)	1 × 2.1 × 1 m

Where a row lands differs, and it is not arbitrary. A **shape** arrives at the origin, because a room is what every other position in the venue is measured from and it wants a known, repeatable spot. **Truss and tray** arrive under whatever you are looking at: they are rigged into a venue that already exists, and the point you are orbiting around is the one place on screen the app knows you mean.

A room has no ceiling on purpose: a closed box is something you cannot see into and cannot rig above. Open at the top, a normal orbit looks straight down into it and a rig hung over it lights in.

Every object is a projection target, an occluder and a measurable body from the moment it exists. Aim a projector at a box and the Metrics panel reads throw distance and image size off whichever face the beam lands on.

3.3 Projection surfaces

"**+ Surface**" asks the same way. A surface is a *screen*: its width is measured **along** the surface rather than across it, and it can be made rear-projection.

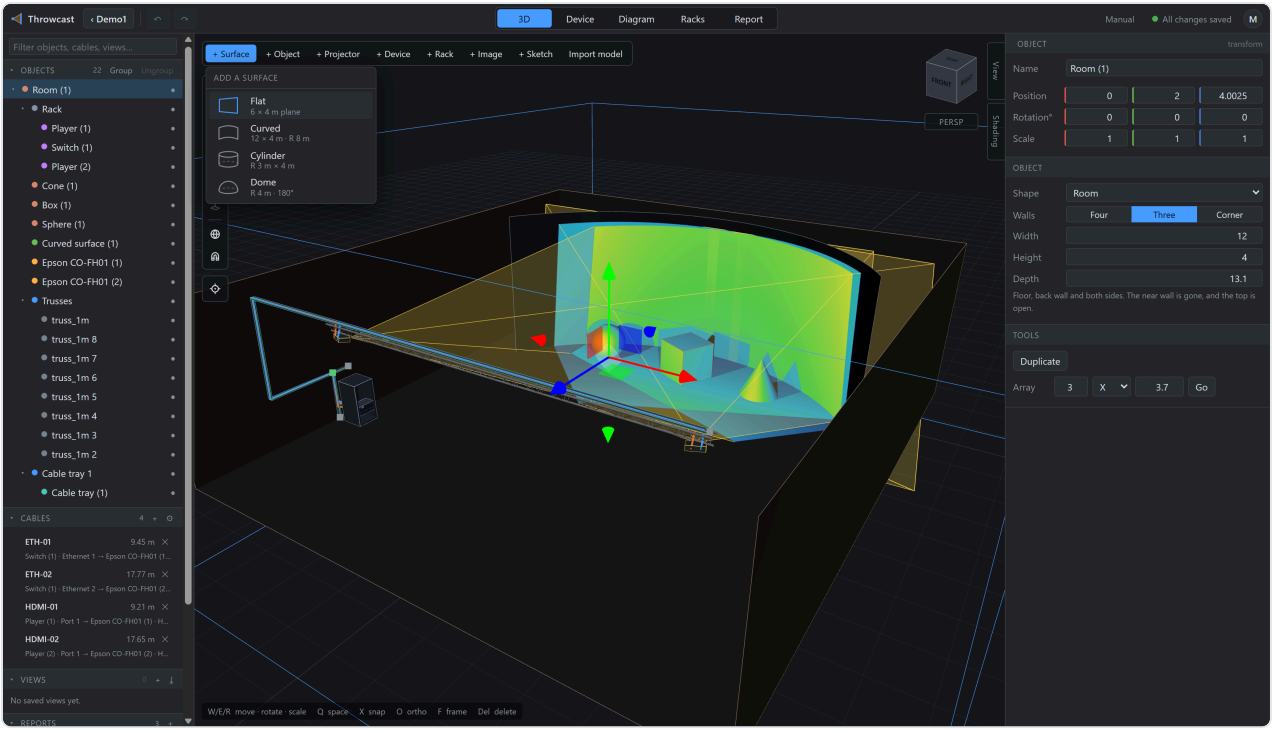


Figure 3.3a The surface shapes. **Flat** is a plane; **Curved** is an open arc; **Cylinder** is that arc closed into a 360° tube; **Dome** is that same arc swept round instead, zenith up.

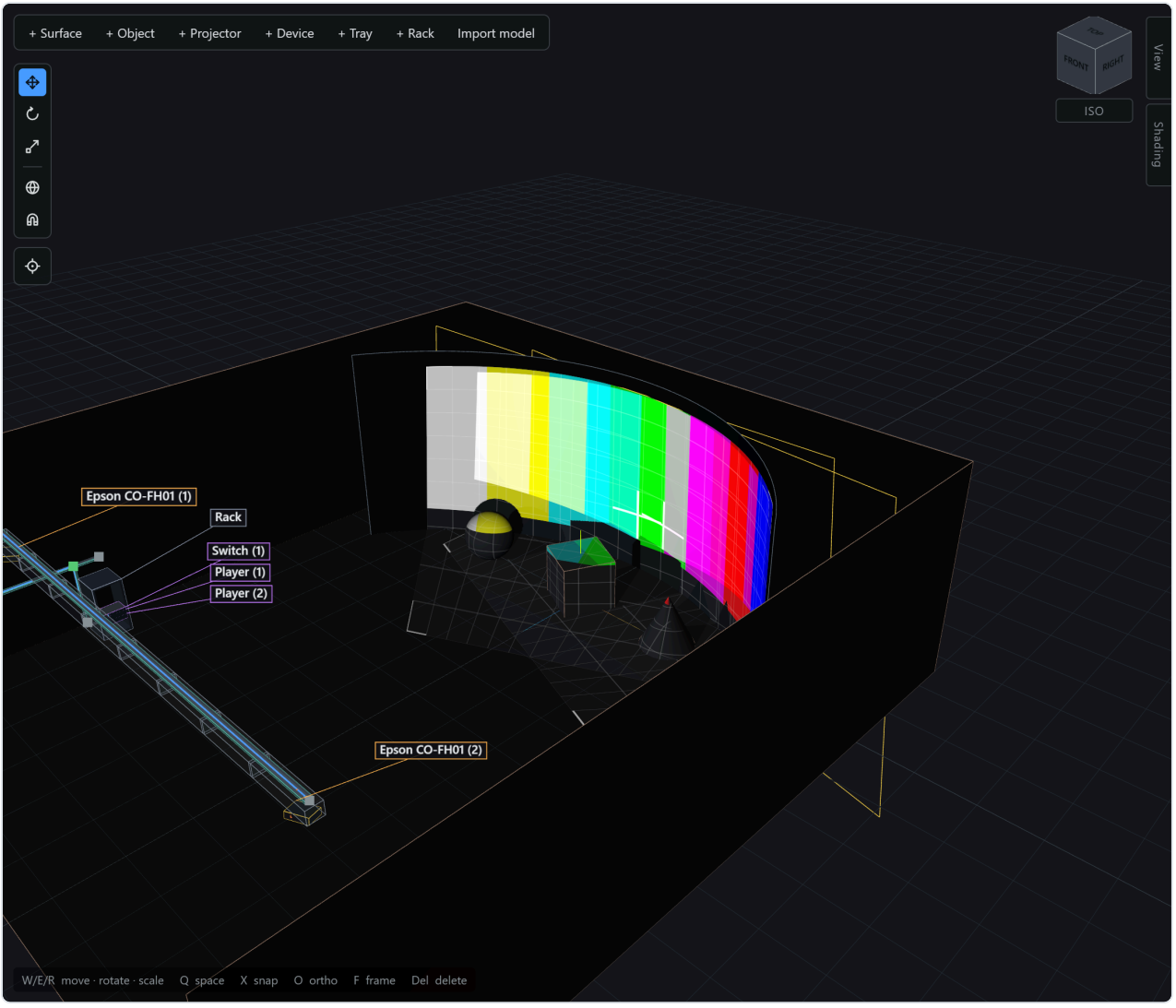


Figure 3.3b The demo project's curved screen: 12 m of screen on a 5.7 m radius, lit by two machines from across the room. **Content preview is switched on here** (§6.3) — an unlit surface photographs as a black rectangle, whereas the test card's grid bends around the curve and the trapezoid shows the tilt's keystone. Metrics fit a tangent plane at the point the beam actually hits.

The shape can be changed at any time from the inspector, and doing so keeps the surface's position, name and any annotations pinned to it.

> **THE RULE**
>
> Switching shape **conserves width** — how much screen you have never changes. Only the radius moves, because a closed tube holding a given width of screen has exactly one radius (`width / 2π`). A cylinder therefore shows you a radius field rather than a width field that would snap back.

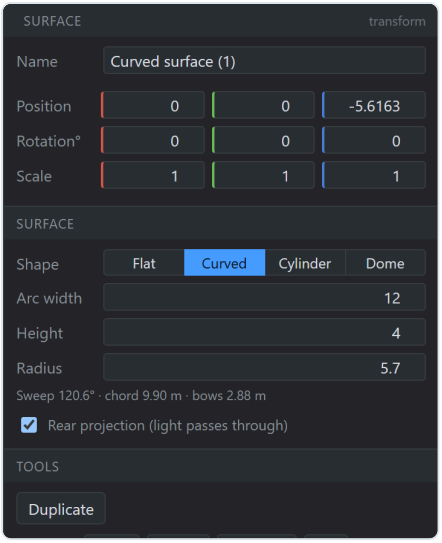


Figure 3.3c The surface inspector. The shape buttons sit above the dimensions; the fields a shape does not use are hidden rather than disabled, and their values are kept.

Domes

A **Dome** is the curved screen swept round rather than extruded: a planetarium. It is described by two numbers — its **radius** and its **dome angle**, which is the angle the trade quotes. **180°** is a hemisphere; more than that is an overshoot dome, wrapping below the spring line to a narrower rim; **360°** closes it into a sphere, at exactly the width that closes a cylinder. Under the two fields the inspector reads back the rim diameter, how deep the bowl is, and how much screen that comes to.

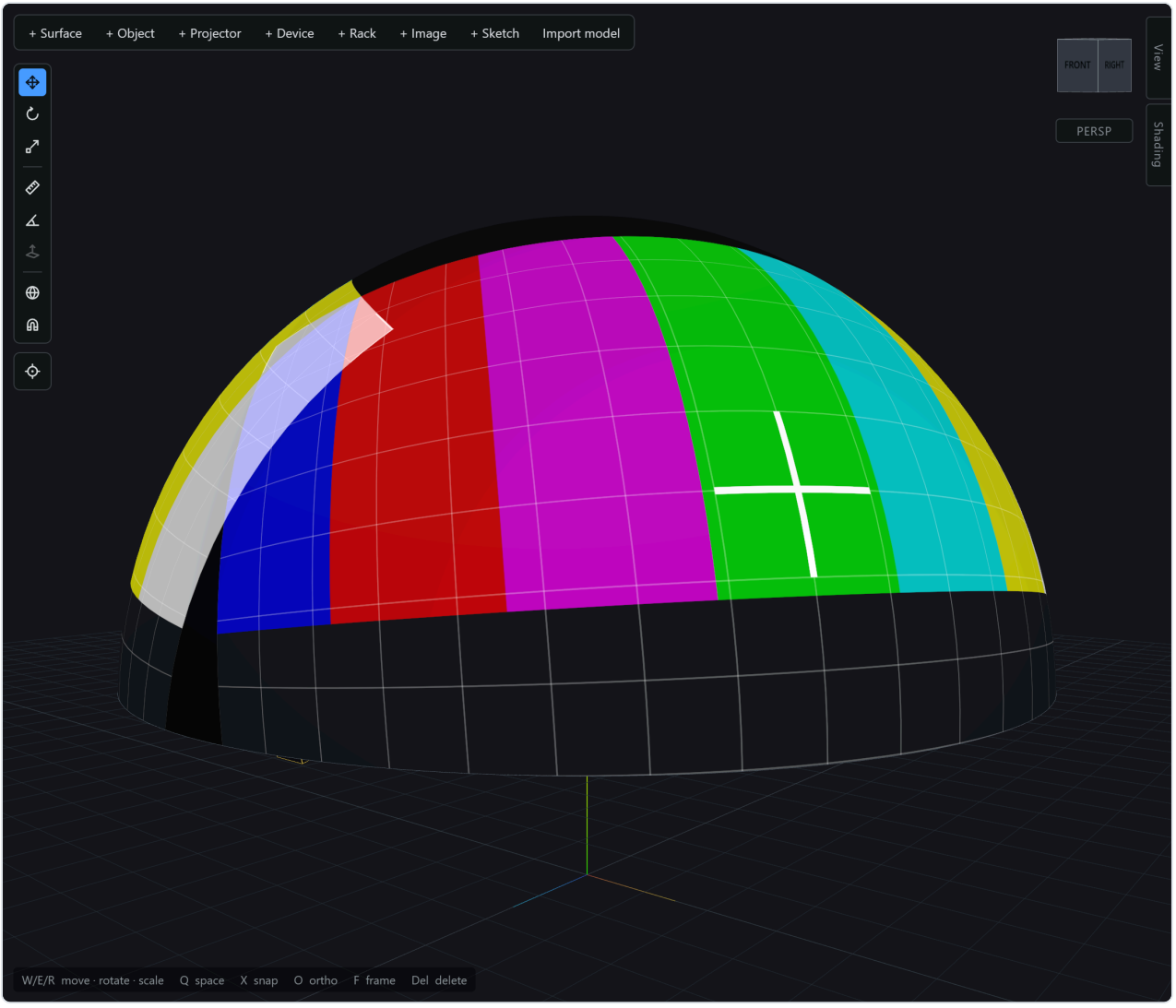


Figure 3.3d A 12 m dome — R 6 m at 180° — with three machines at the cove throwing across it. **Content preview is on** (§6.3), and the test card is what makes the shape read: its grid bends in *two* directions at once, where a curved screen bends it in one. The dark band above the rim is simply where no image lands. The screen is set to rear projection so the interior reads from outside — the rig is inside a shell the camera is not.

A dome is placed by its **centre**, not its skin — as a cylinder is placed by its axis. The object's origin is the centre of curvature, with the zenith straight up from it, which is where the audience sits and where a survey works from. So a **tilted** dome is just the object rotated, tilting about the point it really tilts about, and a dome of radius *r* sits exactly on a cylinder of radius *r* if you want a silo. Height is not a dome's to set — a surface of revolution has none — so that field is hidden, and the value comes back if you make the surface flat or curved again.

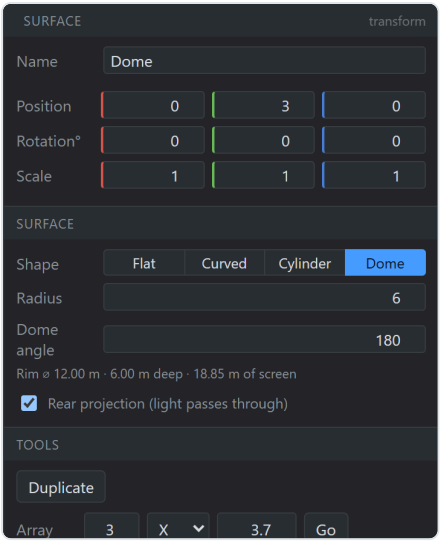


Figure 3.3e The same inspector, on the dome in Figure 3.3d. Width and Height are gone; **Radius** and **Dome angle** stand in their place, and the derived row reads back the rim diameter, the depth and the screen area that pair comes to. Compare Figure 3.3c: one panel, four shapes, and only the fields a shape really has.

Everything else is unchanged: a doorway is a cutter pushed through it, the heatmap and content preview land on it, the probe reads it, and metrics fit a tangent plane at the point the beam hits.

Rear projection

A surface is *opaque* by default: it catches light on the face turned toward each projector, and you see that light only from the same side. Turning that off makes it a rear-projection screen — light passes through, and any projector lights the face you are looking at. Occlusion does not change either way; a screen blocks light from both sides regardless of which face shows the image.

3.4 Importing models

"**Import model**" accepts `.obj`, `.gltf` / `.glb` and `.fbx`. You are asked for the file's **unit** (mm / cm / m) and its **up axis** (Y or Z), because neither is reliably recorded in the formats themselves. Materials are preserved, and a spatial index is built so that picking and raycasting stay fast on venue-sized CAD.

You can also drag model files straight onto the viewport.

3.5 Moving things

Select an object and a gizmo appears at it.

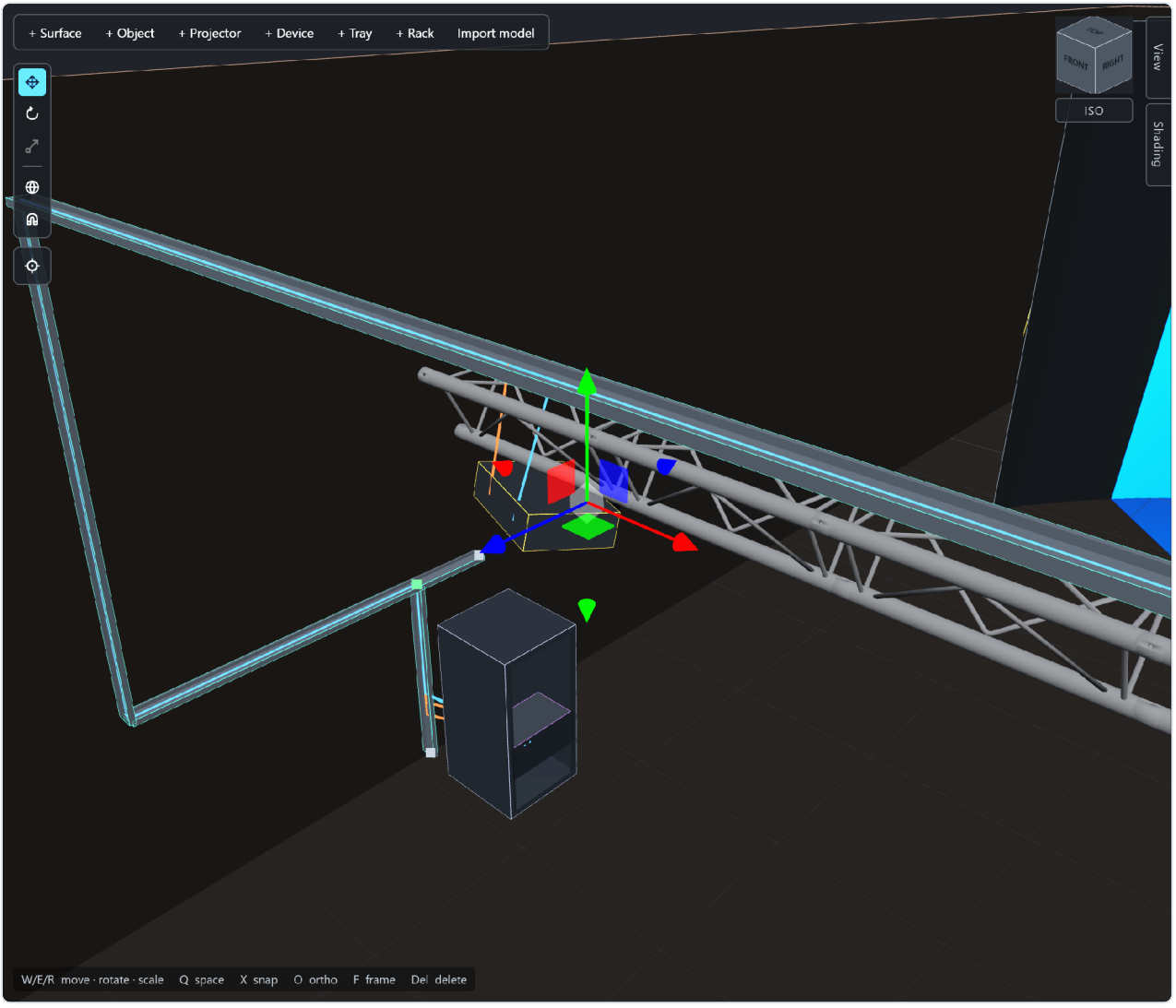


Figure 3.5 The move gizmo. Drag an arrow for one axis, a plane handle for two.

Table 3.2 Gizmo controls.

KEY	DOES
`W` or `G`	Move
`E`	Rotate
`R`	Scale
`Q`	World ↔ local space
`X`	Snapping — 0.1 m translate, 15° rotate

You can always type exact numbers into the inspector instead. For a rigging job that is usually faster: set the height and the throw distance numerically, and use the gizmo only for the last nudge.

3.6 The object tree

Objects can be parented to each other. **A child's transform is stored relative to its parent**, so moving a parent carries everything under it, keeping every offset and aim.

> **READING POSITIONS**
> This is why a projector inside a blend array shows a position like `-2.2249, 0, 0` rather than its world position: the group holds the common part, and the projector holds its offset from it. Select the group to see where the whole rig is.

Rearranging

Drag a row and it tells you what will happen *before* you let go. The row is read in vertical thirds:

Table 3.3 What a drag will do.

POINTER	MARKER	RESULT
Middle of a row	The row lights up	The objects become its children
Top or bottom quarter	A line at that edge, indented to the level it lands at	They become siblings, exactly there
A refused destination	The same marker in red, with a reason in the status bar	Nothing — and you knew before releasing
Empty space below the tree	A line under the last row	Out to the root, at the end

- **A line under the last child of a group is ambiguous** — stay in, or come out? Drag *left* to step out a level, *right* to stay in. The line's own indentation shows which you are getting.
- **Drags carry the selection.** Start on a selected row and the whole selection moves, as one undo entry.
- **Rest on a folded group for a moment and it opens**, so a deep drop does not need you to let go first.
- **Racks and shelves take no line.** Their children are ordered by slot, so a drop inside one becomes a drop *onto* the rack — which mounts the device.
- **A folded branch stays folded** — in the project, not just for the session (§2.3).

Naming

The first one of a kind takes the bare name and the second one is numbered: `Cutter`, then `Cutter (2)`. An index on the only one of a kind is noise, and it promises a `(2)` that may never exist. Duplicating counts *on* rather than growing a tail — `Custom Projector` → `(2)` → `(3)` — and only a parenthesised bare number reads as an index, so a machine you called *Mark II (Rev 2)* keeps its name. The array tools are deliberately left alone: *Wall 2*, *Wall 3* is what those units are called on site.

3.7 Reference images

The rest of this chapter is the other way to build a venue: instead of assembling it out of the shapes above, put the drawing of it in the scene and work from that. Place the architect's plan, scale it to life, draw on a plane over it, and push what you drew into a solid — a riser with a notch in it, a set wall in plan, a plinth whose footprint is the real building's.

"**+ Image**" puts a picture in the venue: a floor plan, an elevation, a seating chart. It arrives the same four ways a sketch does — flat on the floor (**Ground**), upright at the origin facing the camera (**Front**), upright along +X (**Side**), or lying **in a face you click**, upright on it and parented to it, so an elevation taped to a wall rides that wall.

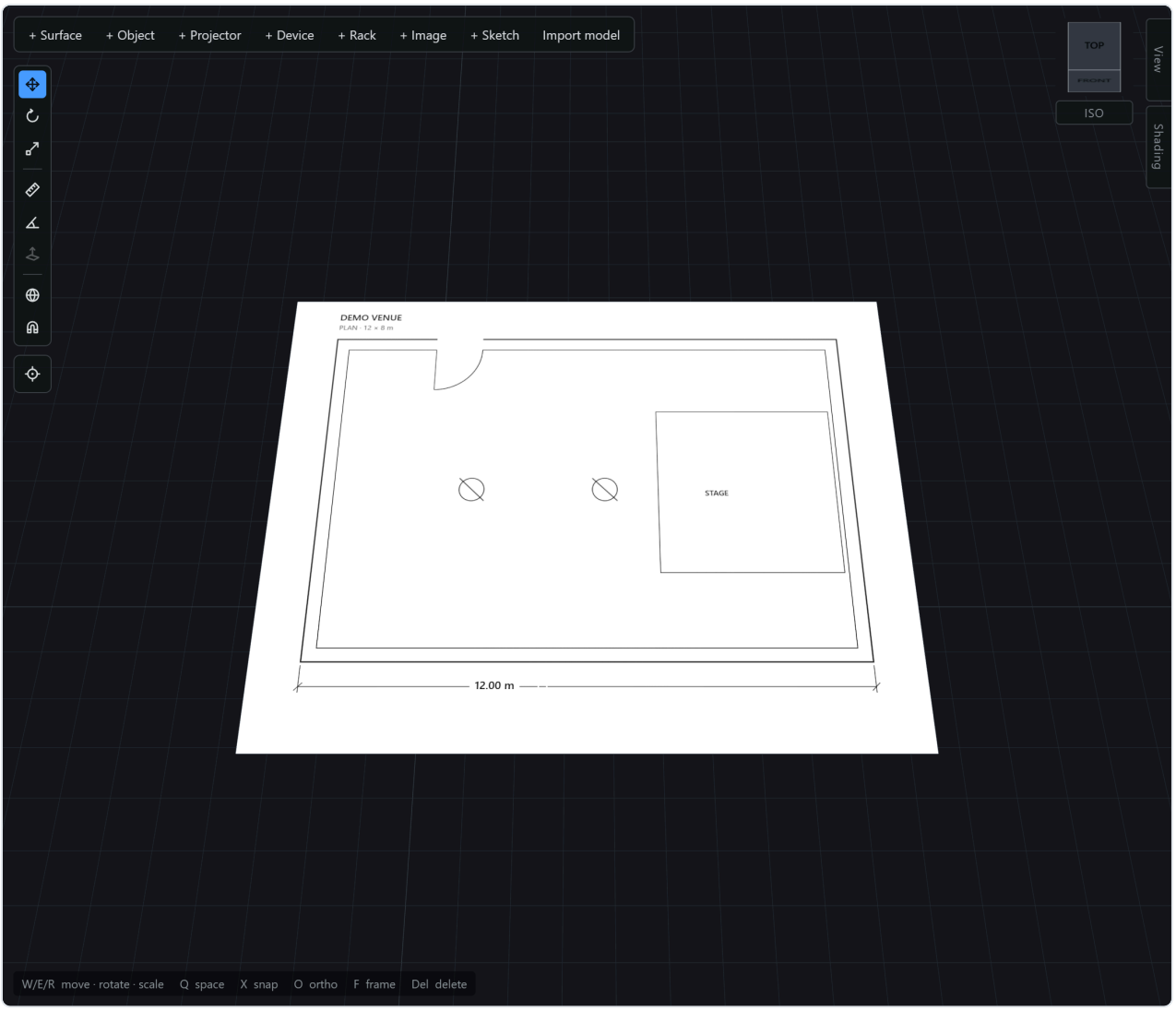


Figure 3.7 A plan on the floor, scaled to life. The 1 m grid running under it is the check that the calibration took: the room the drawing calls 12 m covers twelve squares.

> **THE RULE**
>
> **A picture is a drafting aid, not a screen.** No beam lands on it, it casts no shadow, and it never appears in the heatmap or the content preview — aim a projector at a floor plan and Metrics says *no surface* rather than measuring to it. What it *is* is a **pick target**, because dimensioning to a plan is the entire reason it is in the venue.

3.8 Scaling a picture to life

A picture arrives at a placeholder width, which is almost never its real one. “**Set scale…**” in the inspector fixes that in two clicks and a number: click two points on the picture whose real separation you know — a dimension line, a door width, a column grid — and type what they actually measure.

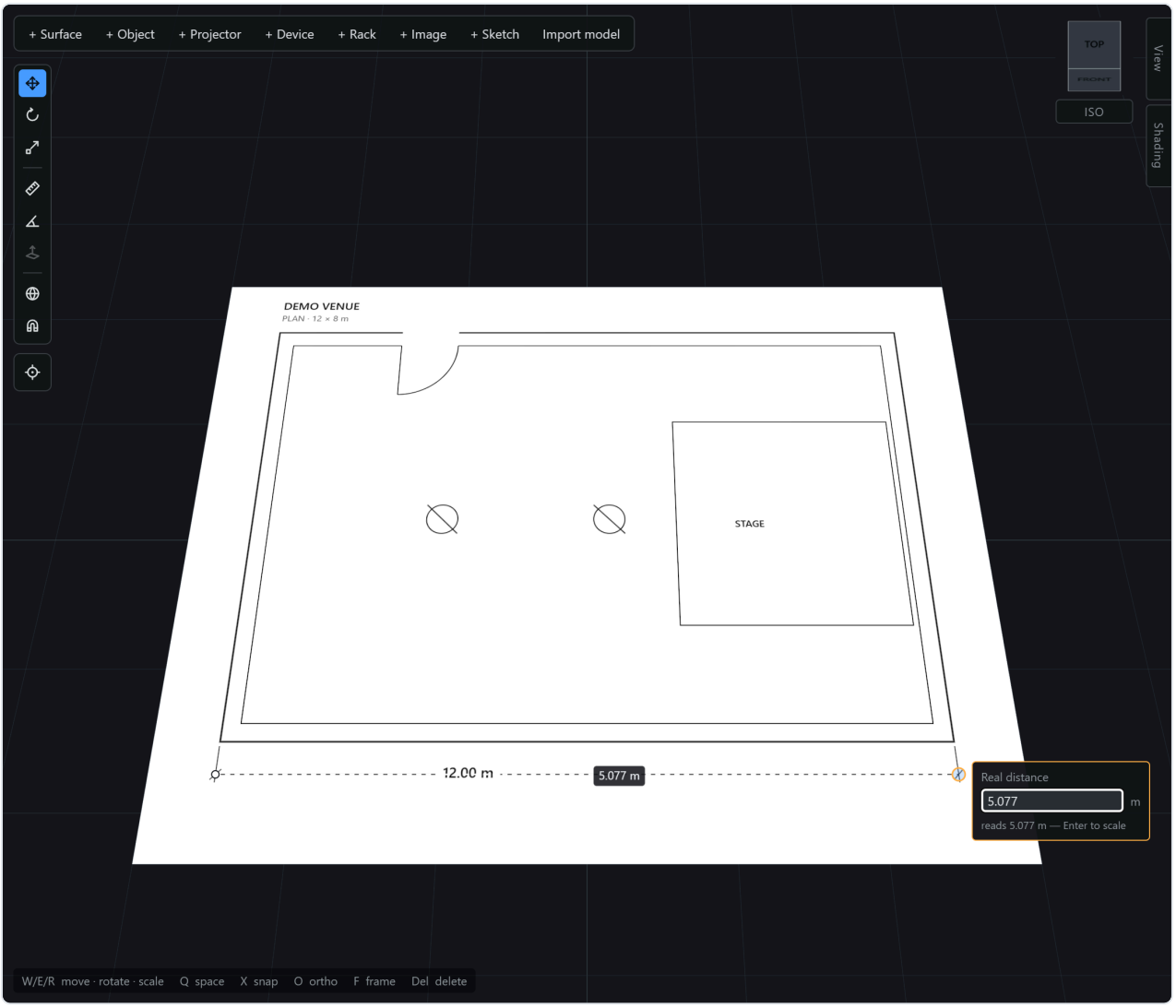


Figure 3.8 Calibrating against the drawing’s own dimension line. The two clicks land on its end ticks, the dashed measurement reads what the picture *currently* spans — 5.077 m, because it arrived at a placeholder width — and the box takes the number the drawing prints beside it: **12.00 m**. Type it, press `Enter`, and the plan is life size.

The plane then rescales **uniformly about the first point you clicked**. That anchor matters: it is what lets you calibrate a plan you have already positioned without it walking off where you put it. The resize and the compensating move go into the document as a single change, so one `Ctrl` + `Z` puts back both.

> **WHAT YOU CALIBRATE IS WHAT YOU CLICKED**
>
> The number you type is the distance between *those two points*, not the width of the picture. A dimension line that measures a 12 m room across a sheet with margins leaves the picture wider than 12 m — correctly so. Pick the longest known distance you can see: the same half-millimetre of click error spread over 12 m is a tenth of the error it is over 1 m.

3.9 Drawing on a plane

"**+ Sketch**" puts a drawing plane in the venue — the same four ways, so a sketch can lie on the floor over a plan or in the face of a wall you are detailing — and goes straight into the editor, because there is nothing to do with an empty sketch except draw on it. "**Edit sketch…**" in the inspector re-opens one later.

Three things happen when the editor opens: the camera eases square-on to the plane and switches to orthographic, the projection *decoration* goes quiet — beams, footprints, labels, cables, the probe — and the drawing tools take the pointer. The venue itself stays exactly where it is, because drawing against it is the point.

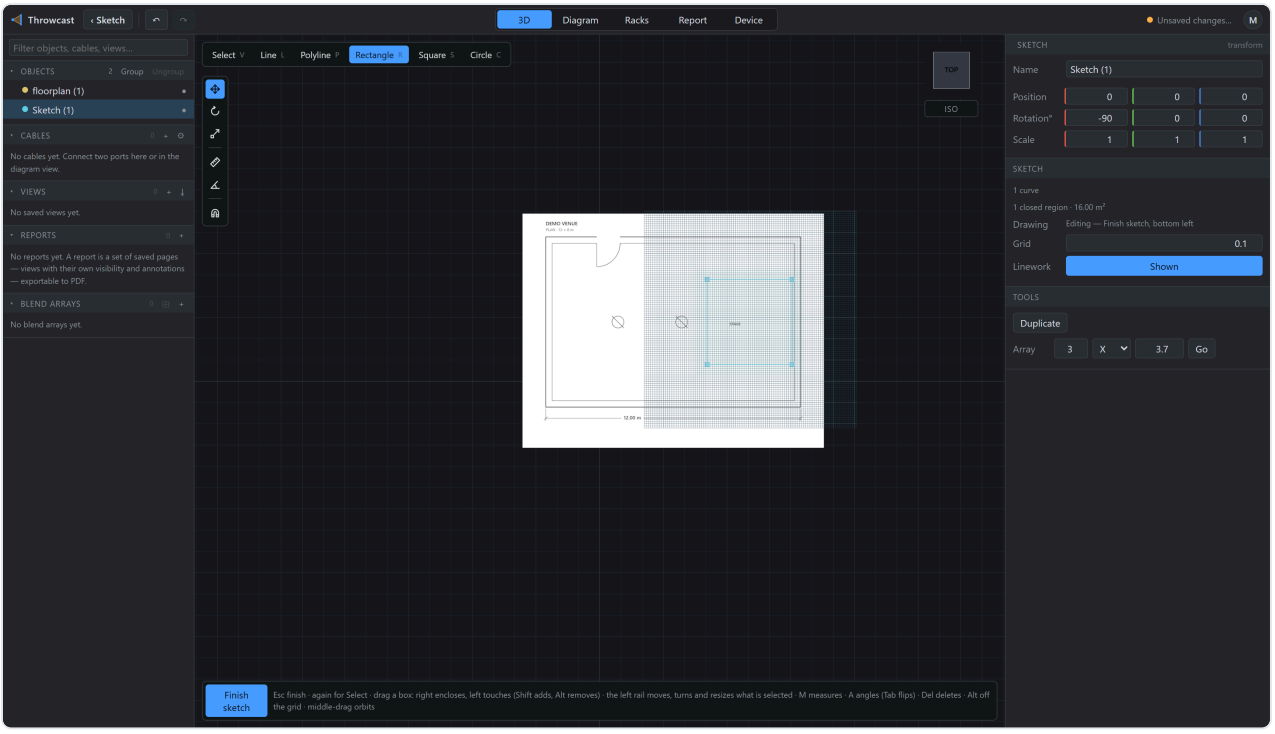


Figure 3.9a The editor open on a ground sketch over the calibrated plan, with the stage traced. The inspector reads **16.00 m²** — the 4 × 4 m the drawing prints, which is what a calibrated plan buys you. The **tools** take the top-left corner, the slot "**+ Surface**" and the rest occupy the other 99 % of the time, and they have stood down. **Finish sketch** sits bottom-left, where the shortcut strip normally lists `W` / `E` / `R`: those keys do nothing while a tool is armed, so the strip says the ones that work instead. What is a form — how many curves and closed regions, a corner's radius — reads in the inspector.

The pen is six tools, one letter each: `V` select, `L` line, `P` polyline, `R` rectangle, `S` square, `C` circle. A line and a polyline mint the same thing and differ only in when they stop. Middle-drag and the wheel still orbit and zoom, so you can look round mid-gesture.

Points snap to an endpoint, a centre, the axes through where the gesture started, then the grid — most specific first, so an endpoint 4 mm away beats a grid line 1 mm away. That order is deliberate: the grid is a convenience, a shared endpoint is what closes a region. Hold `Alt` to draw off it entirely.

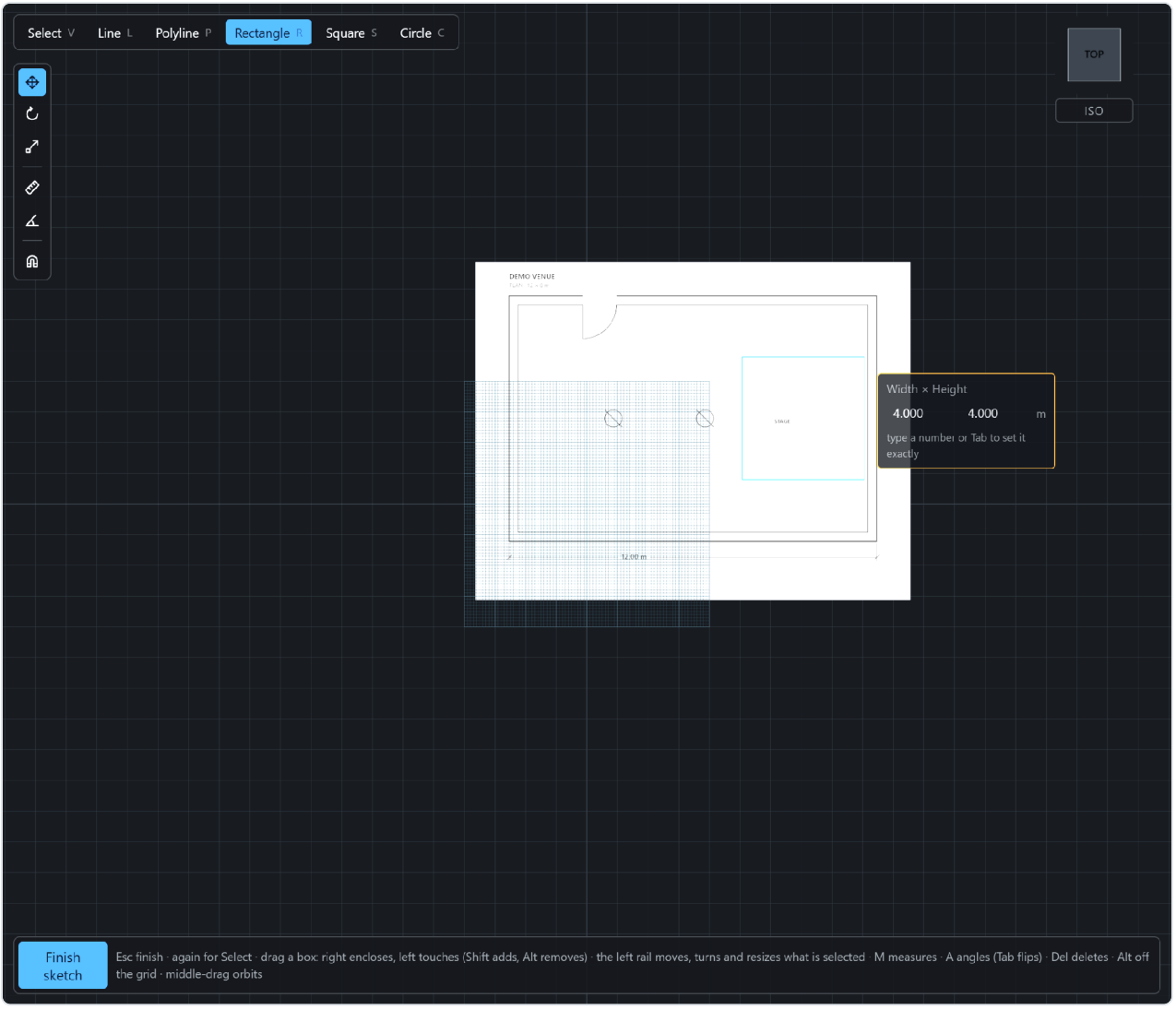


Figure 3.9b **The size shows as you draw**, riding just above the cursor — a length for a line, a radius for a circle, two sides for a rectangle. Here it reads the stage off the plan as it is traced. Type a digit or press `Tab` and the box takes the keyboard: the number becomes yours to set exactly, `Tab` moves between a rectangle's two sides, and `Enter` draws it at that size in the direction the pointer was already pointing. It stops following the moment you type in it.

> **FINISHING A POLYLINE**
>
> A polyline is the only tool that does not know when it is done. It **ends itself** when a click lands on something already drawn — that is the moment a region closes — and clicking its own first point again closes it into a ring. Otherwise `Esc`, `Enter`, reaching for another tool, or "**Finish sketch**" all commit it. Nothing throws a run away; `Ctrl` + `Z` is how you take one back.

To select rather than draw, put the pen down first. A tool stays armed after it finishes a shape — you rarely draw exactly one wall — so a click meant to pick something up starts another shape

instead. `Esc` steps out one level at a time: finish the gesture, then back to **Select**, then leave the editor. From Select, clicking a curve selects the whole curve and `Del` deletes it.

Drag a box on empty plane to take several at once, and which way you drag says what you mean. Dragging **rightwards** takes only what is entirely inside the box, so a wall running out of it is left alone; dragging **leftwards** takes anything the box touches, which is how you grab a run of lines without framing all of them. It is the rule every drafting program uses, and the box changes colour to say which one you are running. `Shift` adds to what you hold, `Alt` removes. Corners and grips inside the box come with it — a point is either in or out.

3.10 Moving what you have selected

The transform rail on the left works on the selection: **move**, **rotate** and **scale**, with the arrows standing at the middle of what you hold and lying in the drawing's plane. There is no third axis, because a drawing has nothing off its plane to say — move runs along the two directions of the sheet, rotate is one ring about it, and scale takes either direction or both at once from the square between the axes. Snapping puts a move on the drawing's own grid.

The rail is shorter in here for the same reason. World/Local has gone (a drawing has one frame, its own), so has "**Extrude**" — pressing it closes the editor to arm itself, and the way out is "**Finish sketch**" — and so has the hover probe, which opening a drawing switched off with the rest of the projection decoration. The three transform buttons carry no letters here either: `W`, `E` and `R` belong to the pen, `R` being Rectangle.

> **THE RULE**
>
> **A shape keeps what it is.** Turn a rectangle and it is a turned rectangle, with its width and height still yours to type and its grips still resizing it — never four loose lines. Two cases cannot be exact and say so rather than pretending: a turned rectangle scaled unevenly stays a rectangle with its own two sides scaled, and a circle scaled unevenly stays a circle of the area the ellipse would have had. Both are exact for an even scale and for anything square to the grid.

Corners move as corners — pick up a dozen and every line welded to each one follows, exactly as dragging one does. A **grip** still resizes its shape when you move it, because that is what a grip is for; select the rectangle itself to move the rectangle. The whole drag is one `Ctrl`+`Z`.

3.11 Dimensions on a drawing

`M` takes the ruler out inside the editor, and two clicks put a dimension on the drawing. It snaps to what the pen snaps to — corners, centres, the axes through the first point, then the grid, with `Alt` to suppress — so corner to corner is exactly corner to corner rather than however near you clicked. `Esc` abandons the one in progress, and again puts the ruler away and the pen back in your hand.

A dimension **belongs to the drawing**. It is measured in the sketch's own coordinates, so it rides the plane: move the wall a layout is drawn on and the numbers go with it, and deleting the sketch

takes them too. It looks and behaves like a scene ruler otherwise — click it and press `Del` to remove one, in the editor or out in the venue. Clicking one takes the **Select** tool, and only Select: with a pen in your hand a click near a dimension draws, and with the ruler out it measures. That is why taking the ruler out also puts the pen down — measure, `Esc`, click what you just made, and it is yours.

A selected dimension grows a grip at each end, and dragging one re-measures from there — onto a different corner, or out to a centre, snapping to everything the ruler snapped to and to the axes through the other end. `Alt` takes it off the grid, and the drag is one `Ctrl` + `Z`. The grips appear only while it is selected, and that is deliberate: an end usually sits exactly on the corner it was measured to, so ends you could always grab would mean that measuring a corner quietly cost you the ability to drag it.

Set "**Dimensions**" to Hidden in the inspector to put them away without deleting them; it is the same switch as "**Linework**", for the other half of the drawing, and it is stored in the file rather than in your viewport preferences.

3.12 Corners and closed regions

A corner is a coincidence, not a link. Two endpoints within a millimetre of each other *are* one corner: drag it and every line attached follows. Nothing records that they are joined — snapping makes the coincidence exact on every commit — so there is no connection to create and nothing to go stale after an undo. It is the rule cable trays already live by (§9.1), at drawing precision.

A rectangle's and a circle's handles are **grips** rather than corners: dragging one resizes the shape and keeps it that shape. That is also the one place a click can surprise you — a handle takes the click ahead of the curve it belongs to, and a circle's grip sits on its rim, so select a circle by clicking the rim somewhere else. Selecting a real corner offers a **radius**, which rounds it.

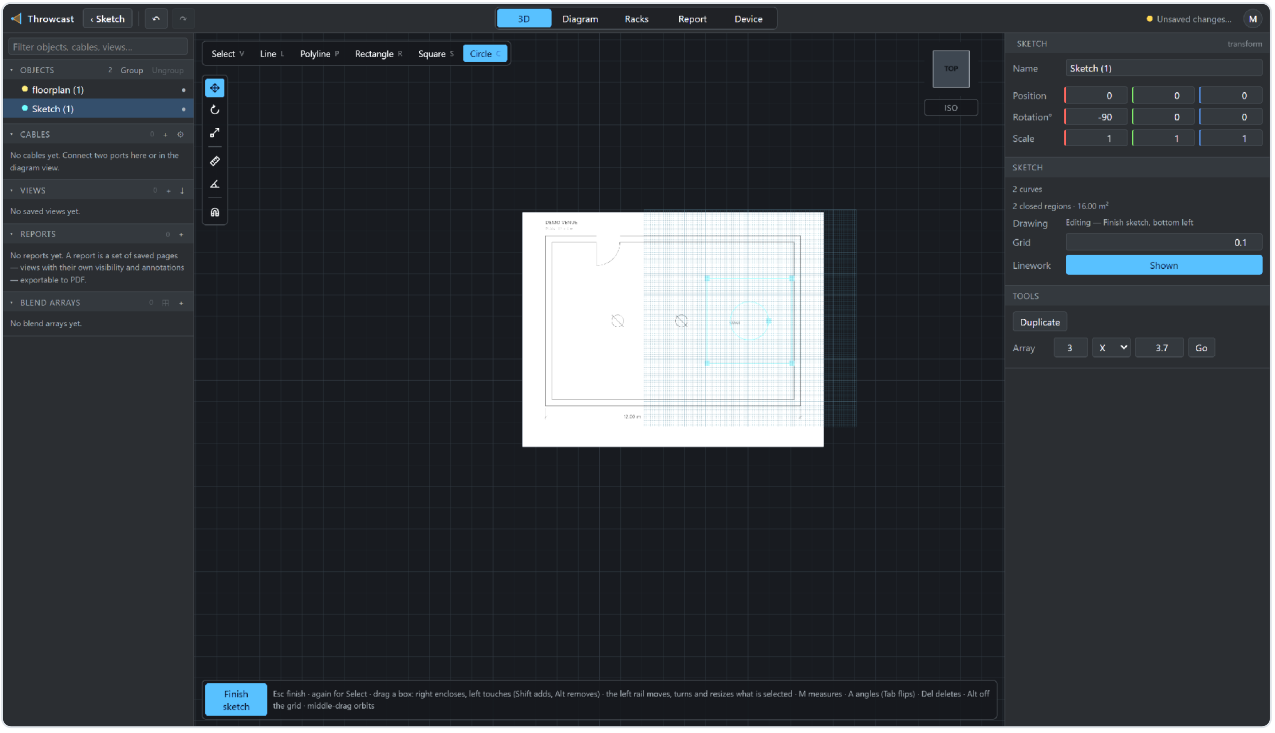


Figure 3.12 A circle dropped into the traced stage — a trap in the deck — and the drawing now reads **two** closed regions: the deck with a hole in it, and the disc filling the hole. Closed regions fill as you draw, and the inspector counts them with their total area. That number is the thing you are drawing toward: finding out the outline never closed *after* you have put the tools away is finding out too late.

Regions are worked out from the lines, never stored, and two consequences are worth knowing. **Regions tile the drawing**: a circle inside a rectangle gives you both the plate with a bore in it *and* the disc filling that bore, so either can be pushed up. And **crossing is not joining** — two rectangles that merely overlap are two regions, not three. An intersection counts only where the drawing says so with a shared point, which is the same rule as two tray runs that cross without touching.

3.13 Extruding a region

Extrude is the last icon on the transform rail, under move / rotate / scale / measure — it is the same kind of thing as the ruler beside it, a cursor mode you switch into and back out of. It acts on the sketch you have selected, or on the one you are drawing: pressing it mid-drawing finishes the sketch first, since choosing what to build wants the pen out of the way. It stays dark until a sketch with at least one closed region is in play, and its tooltip says which of the two is missing.

Armed, every closed region fills faintly and **clicking one builds the solid there and then** — there is no Create step. The panel's **Depth** re-cuts every body you have made as you drag it, so the number is attached to the thing it describes rather than to a promise about it; **Side** puts it front, back, or half each way. Clicking a region again takes its body away, and closing the panel is just putting the tool down.

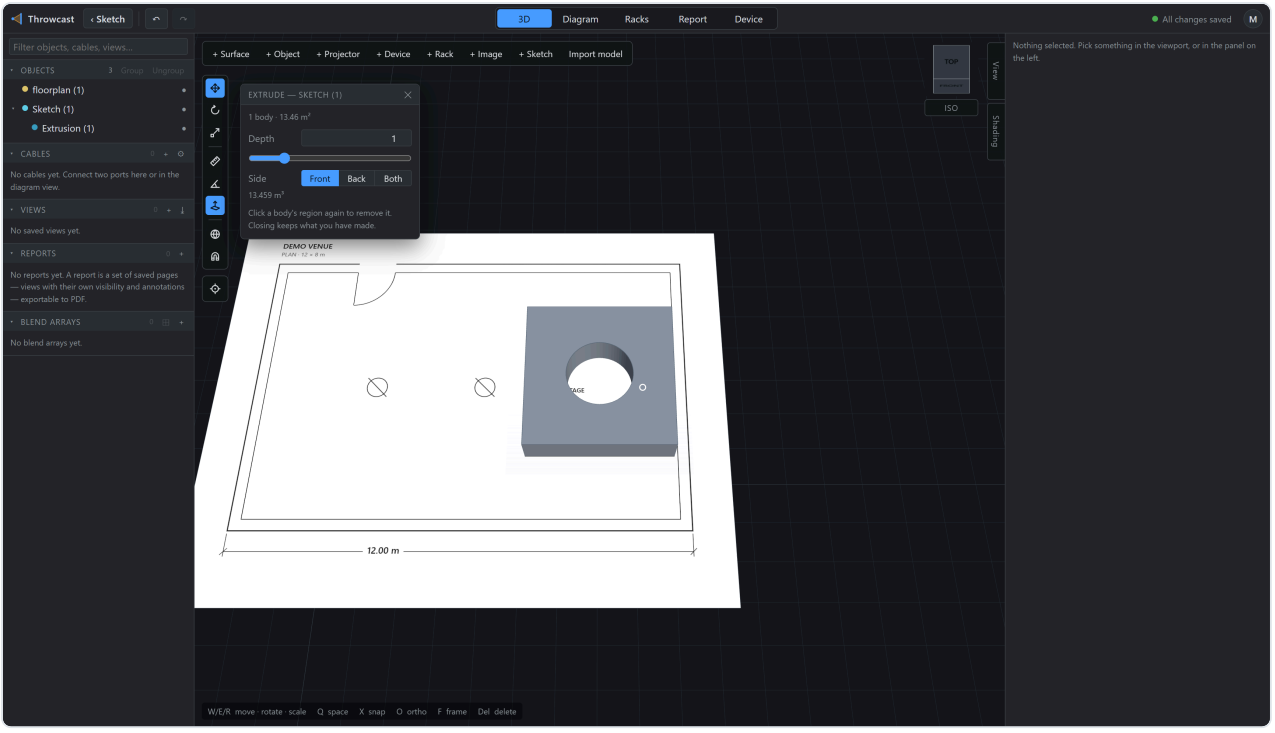


Figure 3.13 The traced deck pushed up a metre, its trap cut straight through — one click on the region, no Create step. The panel counts what you have built and totals its volume; the depth slider covers 0–5 m and the field takes anything taller. The plan is still on the floor underneath, and the body is still tied to the lines drawn over it.

> **THE RULE**
>
> **A body is live-linked to its drawing.** Edit the sketch and every solid cut from it re-cuts — drag a corner and the wall follows, in one undo step. It finds its region again by a point stored *inside* it rather than by a list of lines, because a list of lines is orphaned by exactly the edits this feature exists to make. If the region genuinely goes away the body draws nothing and says *region lost*, and the inspector offers "**Re-pick…**" — it never silently vanishes and never silently bakes.

Unlike the drawing it came from, a body is ordinary scenery: **lit, occluding and measurable**, light-tight with no opt-out. It carries no scale, because its outline is its sketch's and its thickness is the depth you set — a scale would pull the solid off the drawing it is supposed to be.

Deleting a sketch deletes its bodies, since they are its children; one `Ctrl` + `Z` brings the lot back. To get the drawing out of the way without that, set "**Linework**" to Hidden — a different control from the eye in the object tree on purpose. The eye hides the whole assembly, bodies included; this hides the pencil and keeps the wall. A finished sketch shows only its lines anyway: the plane and the fills are working apparatus, and they come back when you select it.

3.14 Doors, windows and holes

A venue has openings in it, and they matter: light goes through a doorway, the wall beside it still throws a shadow, and a machine aimed through one measures its throw against the far surface rather than the one it is shooting past. "**+ Object ▸ Cutter**" is how you make them.

A cutter is a **body that dissolves whatever it overlaps**. Push it through a wall and you have a doorway. Move it and the doorway moves; scale it and the doorway resizes; hide it and the wall closes up again.

> **THE RULE**
> **A cut is *where a thing is***, so it is placed the way everything else in this app is placed — with the gizmo, against the model, by eye and by snapping — rather than by typing four numbers into a panel that owns it. Nothing is stored about what cuts what: a cutter and a wall are related by overlapping in space and by nothing else, so there is no link to go stale on a delete, an undo or a reparent.

That is also what lets one cutter open **everything it touches at once**: pushed into a corner it opens both walls, and sunk into the floor line it notches the floor and the wall together. A hole across two bodies is one hole, which a list of openings hung off either of them could not express at all.

It arrives 1 × 2.1 × 1 m — doorway-sized, and a metre deep so it comes out the far side of any wall it meets. Four shapes, switched in the inspector, which is *all* the inspector has, because everything about *where* lives in the transform section above it:

Table 3.4 The four cutter shapes.

SHAPE	FOR
Box	Doorways, windows, service hatches — the common case
Cylinder	A cable bore, a porthole, a lighting penetration
Cone	A splayed opening, a countersink
Sphere	A dished recess

> **PUSH IT RIGHT THROUGH**
> A cutter face left exactly flush with the wall it is cutting is the one case the geometry underneath handles badly. Overshoot: a body you can see makes that obvious in a way a depth field never did.

It draws as a **pink ghost** — a wireframe cage over a barely-there fill. The cage is what you aim by, because the thing you are positioning is an edge (the jamb of the doorway) and an edge on a solid body is exactly what you cannot see through to the wall behind. Nothing else in the scene is pink, so a cutter never reads as scenery at a glance.

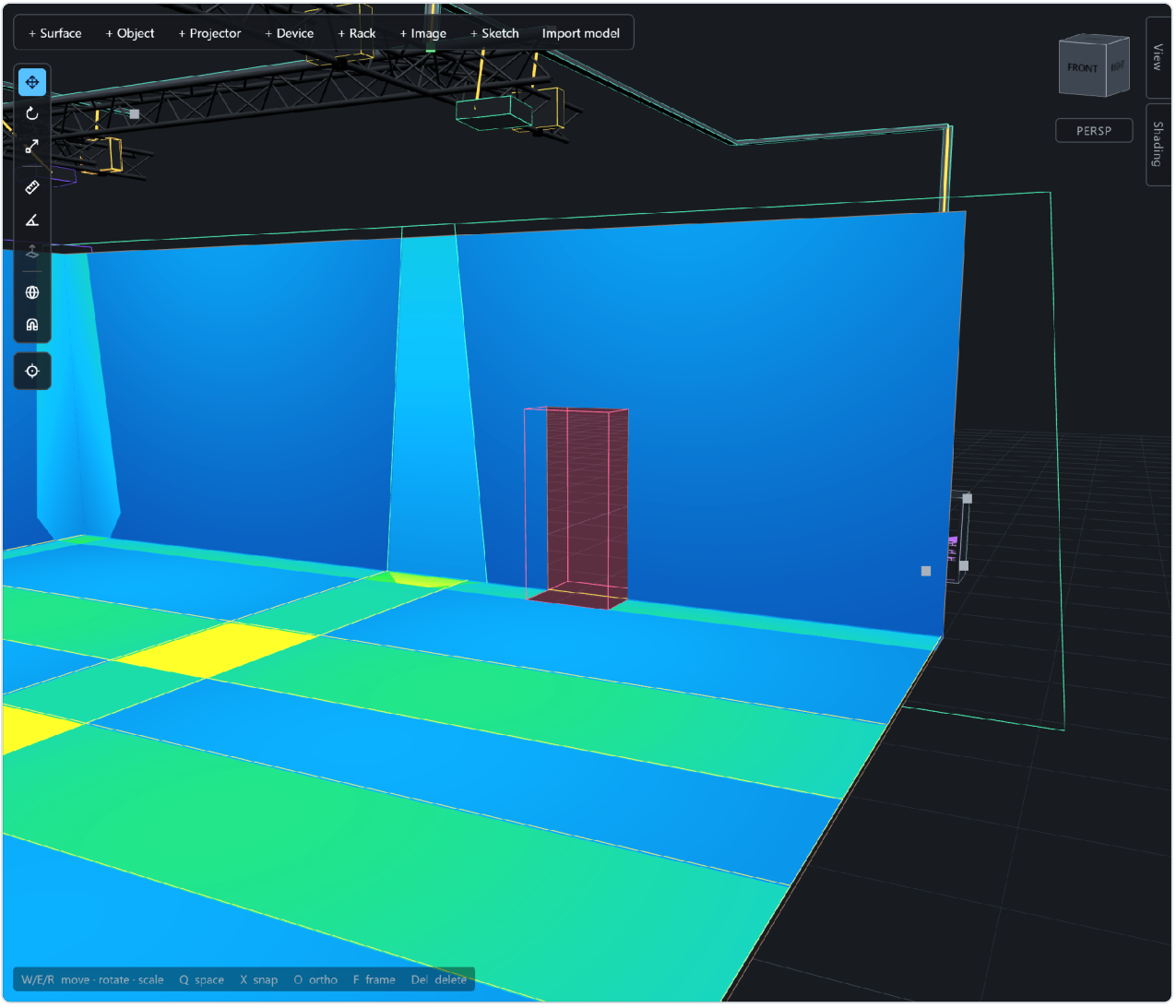


Figure 3.14a One cutter, pushed through the back wall of a lit room and standing on the floor. It arrived at this size — 1 × 2.1 × 1 m — and nothing was typed to place it: it is a body, moved with the gizmo like everything else. The wall is already open where the cage stands; the cage is simply in front of the opening.

What it cuts: projection surfaces in all three shapes, room walls, the four solid objects, and extrusions. Not imported CAD models.

What it is not: part of the simulation. A cutter catches no light, casts no shadow, has no footprint and never becomes a throw distance. It is a tool standing in the venue, not a body in it.

> **PUTTING THE GHOSTS AWAY**
>
> A venue with its doorways placed is a venue full of pink boxes standing in the way of looking at it. "**View ▸ Cutters**" stops drawing them, and stops there — every opening stays exactly as it was. That is *not* the same as hiding a cutter with the eye in the object browser, which takes it out of the venue and closes the wall up again. (A ghost you cannot see is also one you cannot click: select it in the browser, as with anything else that is not on screen.) A report's 3D view freezes the toggle like any other display setting, so the sheet you hand over can be of the finished room while the one you keep shows what shaped it.

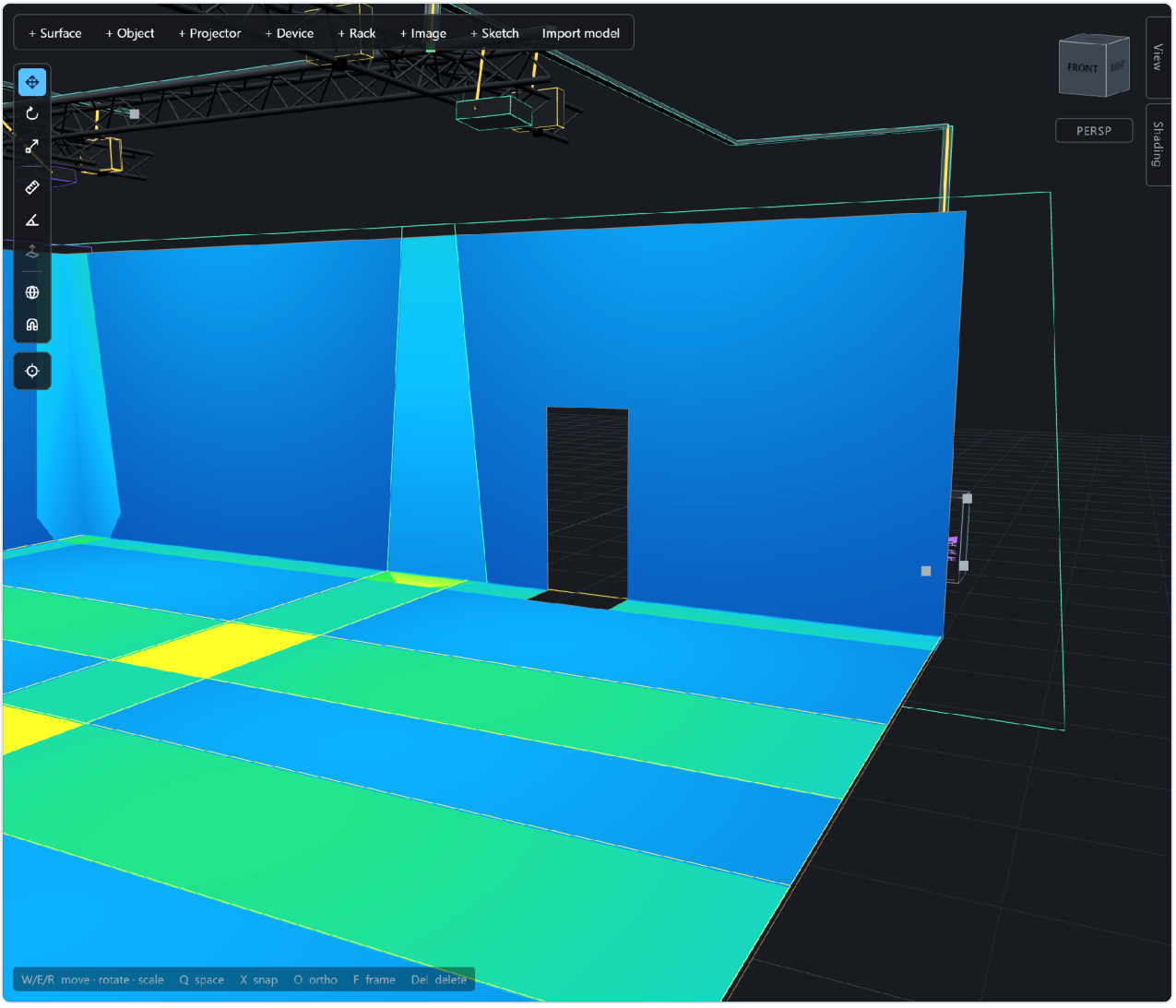


Figure 3.14b The same camera, one toggle later: "**View ▸ Cutters**" off. The doorway is inked at its jambs like every other edge in the venue, the wall around it is still catching light, and the room beyond it is dark because there is nothing out there to light. Nothing about the cut changed — only whether the tool that made it is drawn.

Everything else about a cut wall is unchanged, and that is the surprising part: **the light, the shadows, the probe and your clicks all pass through the opening**. A hole is an absence of triangles, and occlusion here is geometric rather than analytic, so the bright rectangle on the floor beyond a doorway is simply what the raycasts find. Coverage figures are the one deliberate exception — "**Footprint area**" still measures the keystone quad and does not subtract the holes the beam passes through.

CHAPTER 4

Projectors and optics

A projector in Throwcast is a body, a lens and a handful of numbers. Get those right and everything downstream — coverage, blending, the heatmap, the reports — follows.

4.1 Adding a projector

"**+ Projector**" opens the definition picker. It offers what **this project** already owns, and a read-only **catalogue** of real machines.

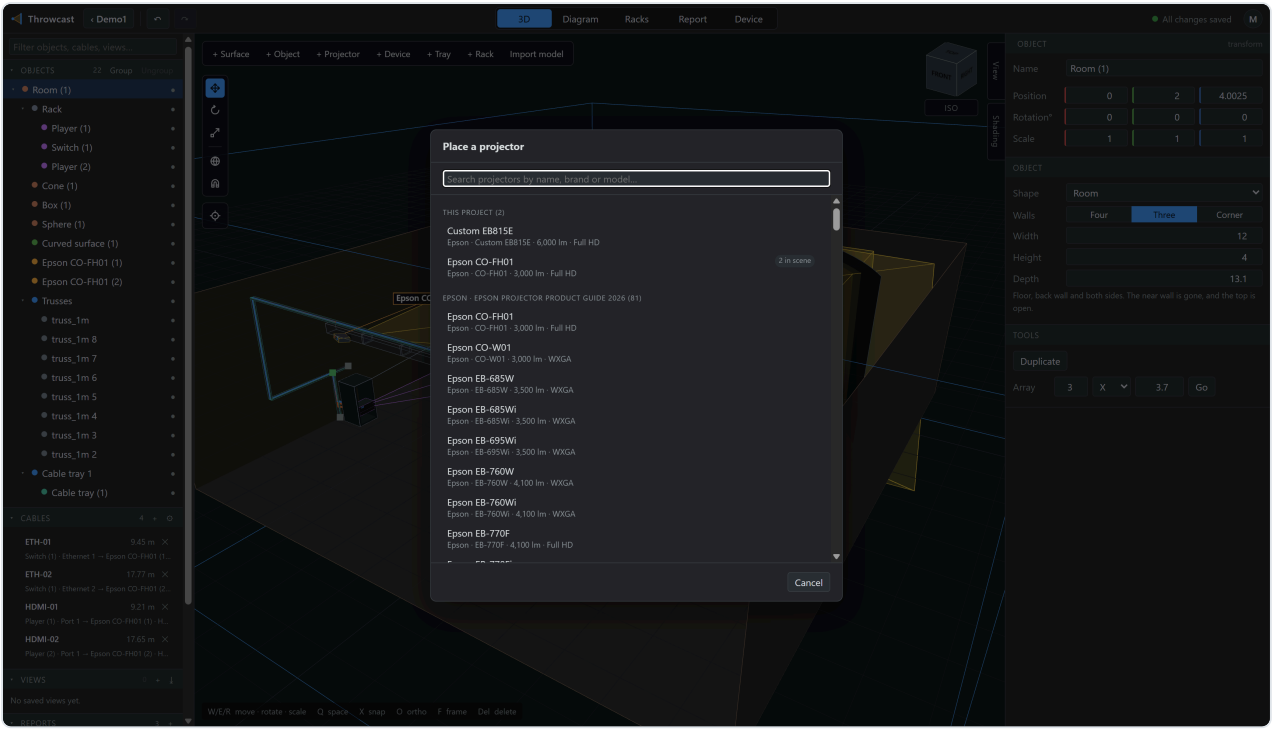


Figure 4.1 The picker. Search matches name, brand and model; each row carries a spec line and, where relevant, a badge saying how many are already in the scene.

The bundled catalogue covers 81 Epson bodies with 48 lenses, and 96 Panasonic bodies with about 200 lens entries. It is the same for every user and cannot be edited. "**Custom Projector**" gives you a blank machine to fill in yourself — see Chapter 7.

4.2 Throw ratio and image size

The throw ratio is the one number that ties distance to image size:

> **THE FORMULA**
>
> **Throw ratio = axial distance ÷ image width.** So at a distance `d`, the image is `w = d / TR` wide, and `h = w / aspect` tall.

A zoom lens has a *range* of throw ratios; the inspector clamps what you type to the range of the lens actually mounted. A shorter ratio means a bigger image from the same spot.

> **WHERE THE BEAM STARTS**
> Light does not leave from the middle of the chassis. Throwcast throws from the **lens** — the projector's position plus the lens protrusion along its forward axis (0.06 m on the reference machines). Ignoring that makes footprints roughly 2.5 % too wide, which is enough to lose a blend. Folded optics — ultra-short-throw, right-angle and rear-throwing lenses — carry a full exit pose instead, so the beam can leave sideways or from a virtual point inside the body.

4.3 Lens shift

Lens shift moves the image without moving or tilting the projector. It is expressed as a **percentage of image size**, not as a distance:

- **Shift V** is a percentage of image *height*. Positive is up.
- **Shift H** is a percentage of image *width*. Positive is to the projector's right.
- `Shift V = −50` puts the top edge of the image exactly on the lens axis.

Shift is almost always better than tilt. A shifted image stays rectangular; a tilted one becomes a trapezoid (§4.6).

WORKED EXAMPLE — DEMO2'S SHIFTED MACHINES

Machine	`Epson EB-L695SE, WUXGA 1920 × 1200`
Throw ratio	`0.50`
Throw distance	`3.448 m`
Image (3.448 ÷ 0.50, ÷ 1.6)	`6.896 × 4.310 m`
Shift V	**`−50 %`**
What −50 % buys	`the image's top edge lands exactly on the lens axis`
Pixel density	`278 px/m`

These machines are also tilted, at 37.0° incidence, which is why their keystone widths do *not* come out equal — 6.922 m across the top against 7.772 m across the bottom (§4.5). The eight machines of the main array are the other case: no shift, no tilt, and top and bottom both 6.557 m.

4.4 The frustum

Every powered projector draws its beam: a wireframe pyramid from the lens to the surface, capped where it lands.

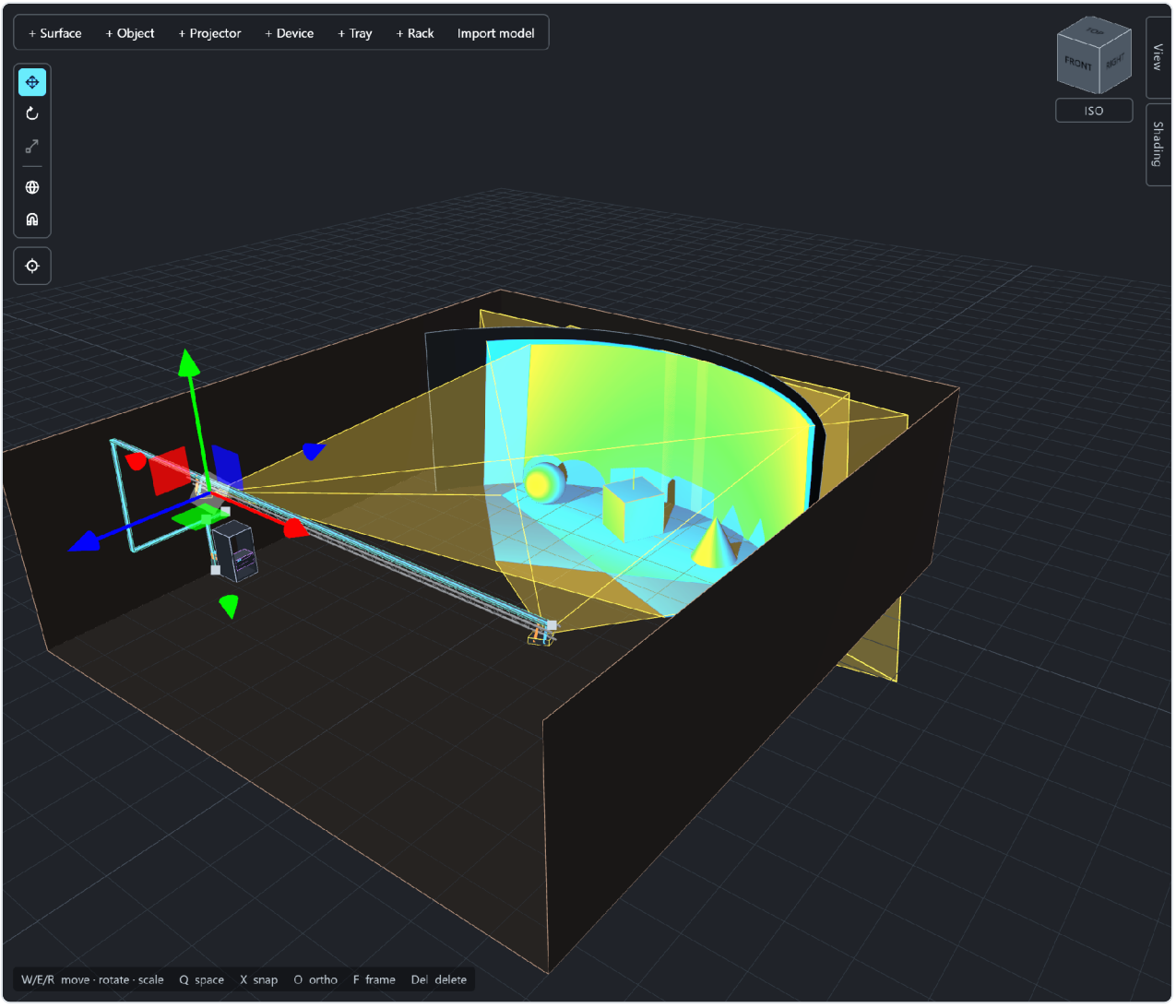


Figure 4.4 Beams from the two main machines. Where they cross is the blend region; the footprint outline on the screen is the image each one actually makes.

Beams and footprints can be turned off independently in the "**View**" panel — useful once a rig has more than a handful of machines.

4.5 Reading the Metrics panel

Metrics are recomputed every frame from a real raycast: a central ray plus the four corners of the frame, fired at whatever geometry is in the way.

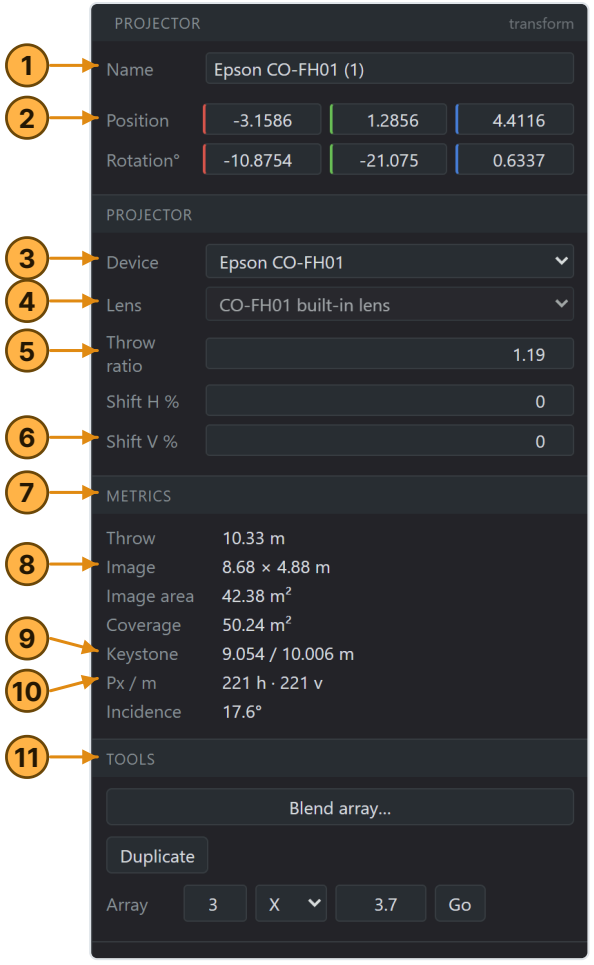


Figure 4.5 The projector inspector, top to bottom.

1. **Name** — this instance, not the model.
2. **Transform** — position and rotation, local to the parent.
3. **Device** — which definition this is an instance of.
4. **Lens** — which of the device's compatible lenses is mounted.
5. **Throw ratio** — clamped to the mounted lens's range.
6. **Shift H / V** — percent of image width / height.
7. **Metrics** — everything below is measured, not entered.
8. **Image** — width × height where the beam lands.
9. **Keystone** — true top and bottom edge widths (§4.6).
10. **Px / m** — delivered pixel density, horizontal and vertical.
11. **Tools** — blend array, duplicate, linear array.

Table 4.1 What each metric means.

METRIC	MEANING
Throw	Distance from the *lens* to the surface along the optical axis.
Image	The image the optics make at that distance: `w = d/TR`, `h = w/aspect`.
Image area	Area of that rectangle — the divisor in the lux calculation.
Coverage	Area of the polygon actually landing on geometry. Lower than image area when the beam spills off an edge.
Keystone	True top / bottom edge widths. Equal on a square-on machine; unequal when tilted.
Px / m	Delivered pixels per metre. The number to argue about viewing distance with.
Incidence	Angle between the beam and the surface normal. Light falls off with its cosine.

If a projector hits nothing, the panel says so and gives the reason rather than showing stale numbers.

4.6 Keystone

Tilt a projector and its footprint stops being a rectangle: the far edge is further away, so it is wider. Throwcast reports the true trapezoid rather than a planar approximation.

WORKED EXAMPLE — THE DEMO PROJECT'S CYC MACHINE

Throw ratio / tilt	`1.44 · 12° down`
Throw distance	`7.54 m`
Nominal image	`5.24 × 3.27 m`
Top edge (further away)	`5.12 m`
Bottom edge (nearer)	`5.62 m`
Difference	`0.50 m — about 9 %`

Half a metre of taper across a 5 m image. In a blend that is the difference between a seam you cannot see and one you can.

CHAPTER 5

Blend arrays

A blend array is a parametric rig of projectors: clone one machine into a **grid** or a **ring**, overlap them by a stated percentage, and keep driving that shape for the life of the project.

5.1 Creating one

Select a projector that is already aimed where you want it, then open the tool from "**Tools → Blend array…**" in the inspector, or the "⊞" in the Blend arrays section header.

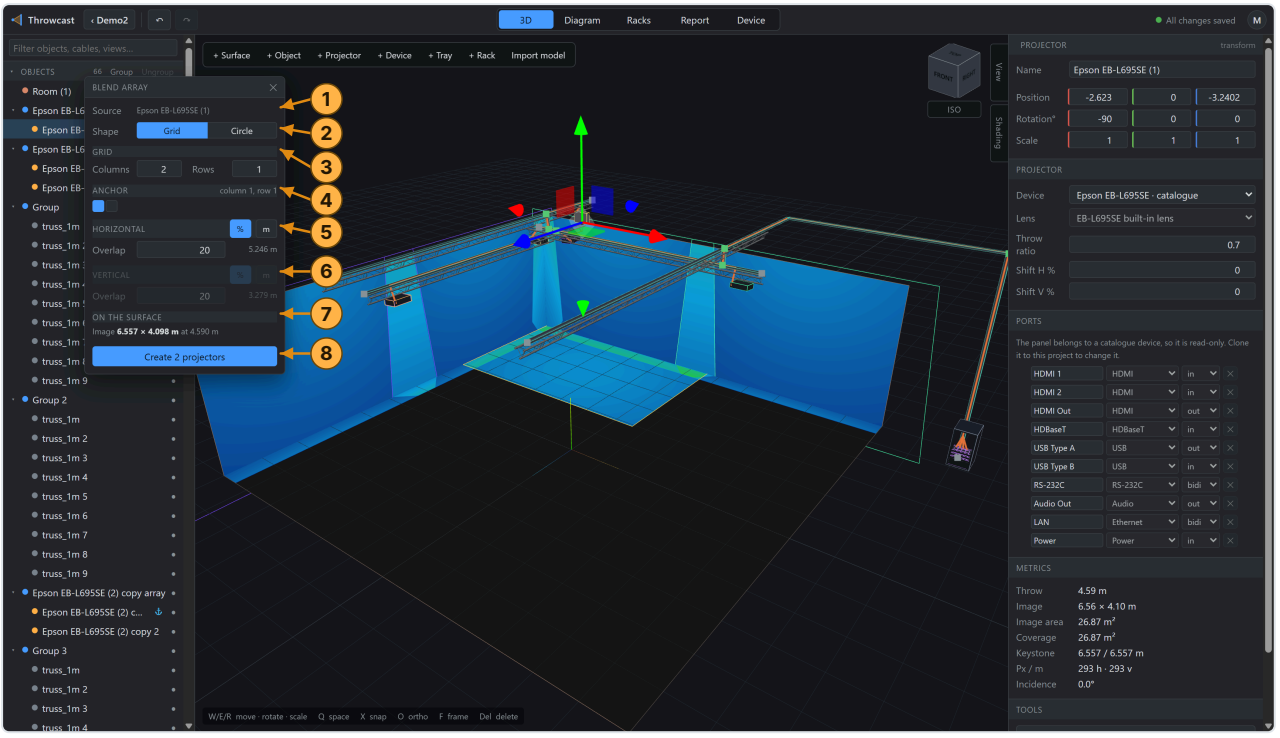


Figure 5.1 The blend-array tool.

1. **Source** — the projector every cell is a clone of.
2. **Shape** — a grid, or a ring (§5.4). Only before "**Create**".
3. **Grid** — columns across, rows up.
4. **Anchor** — which cell the source machine occupies. It never moves.
5. **Horizontal** — overlap % or step in metres; the other is a live readout.
6. **Vertical** — the same, for rows.
7. **On the surface** — the image the source actually makes, at its real throw.
8. **Create** — builds the rig, in one undo entry.

The panel is a *modeless* floating window: it can be dragged by its title bar, and keyboard shortcuts keep working while it is open so you can go on nudging projectors underneath it.

5.2 Overlap or step

Each axis is driven by one of two numbers, and shows the other:

- **Overlap %** — the share of the image that two neighbours share. This is the number a blend is specified in.
- **Step (m)** — the physical spacing between machines.

The percentage is solved against the image the anchor *actually makes on the surface it hits*, recomputed live — so 20 % means 20 % of a real image, not of an assumed one.

WORKED EXAMPLE — DEMO2'S 4 × 2 ARRAY

Anchor image	`6.557 × 4.098 m`
Horizontal overlap	`17.63 % (1.156 m)`
Step across (6.557 × 0.8237)	`5.401 m`
Vertical overlap	`35.48 % (1.454 m)`
Step up (4.098 × 0.6452)	`2.644 m`
Total width (2 × 6.557 − 1.156)	`11.958 m`
Total height (4 × 4.098 − 3 × 1.454)	`12.030 m`

Eight machines, two columns by four rows. The vertical overlap is twice the horizontal because the rows are stacked up a surface the beams meet head-on, where the cost of overlap is cheap.

5.3 The grid is the anchor's own

Cells march along the **anchor's local right and up**, not world X and Y. That is the layout you would see standing behind the rig, and it survives a tilted or rolled machine. Clones are parallel to the anchor, so moving *or rotating* the anchor carries the whole array with it.

The anchor and the four corners of the grid are drag handles. Dragging a corner re-derives the step along the axes it spans — and on an overlap-driven axis it back-solves the percentage, so dragging a corner literally edits the number in the panel. Only the component along each grid axis is used; the corner snaps back onto the lattice, which is the feedback that says the array stays rectangular.

5.4 Rings

A great many rigs are not walls. A cyclorama, a drum, a 360° screen: the machines stand on a circle and throw outward at it, or ring a central object and throw in. Switch "**Shape**" to "**Circle**" and the same tool builds that — same spec, same undo, same everything below. "**Units**" is the count around the circle and "**Rows**" becomes *tiers* stacked along the axis.

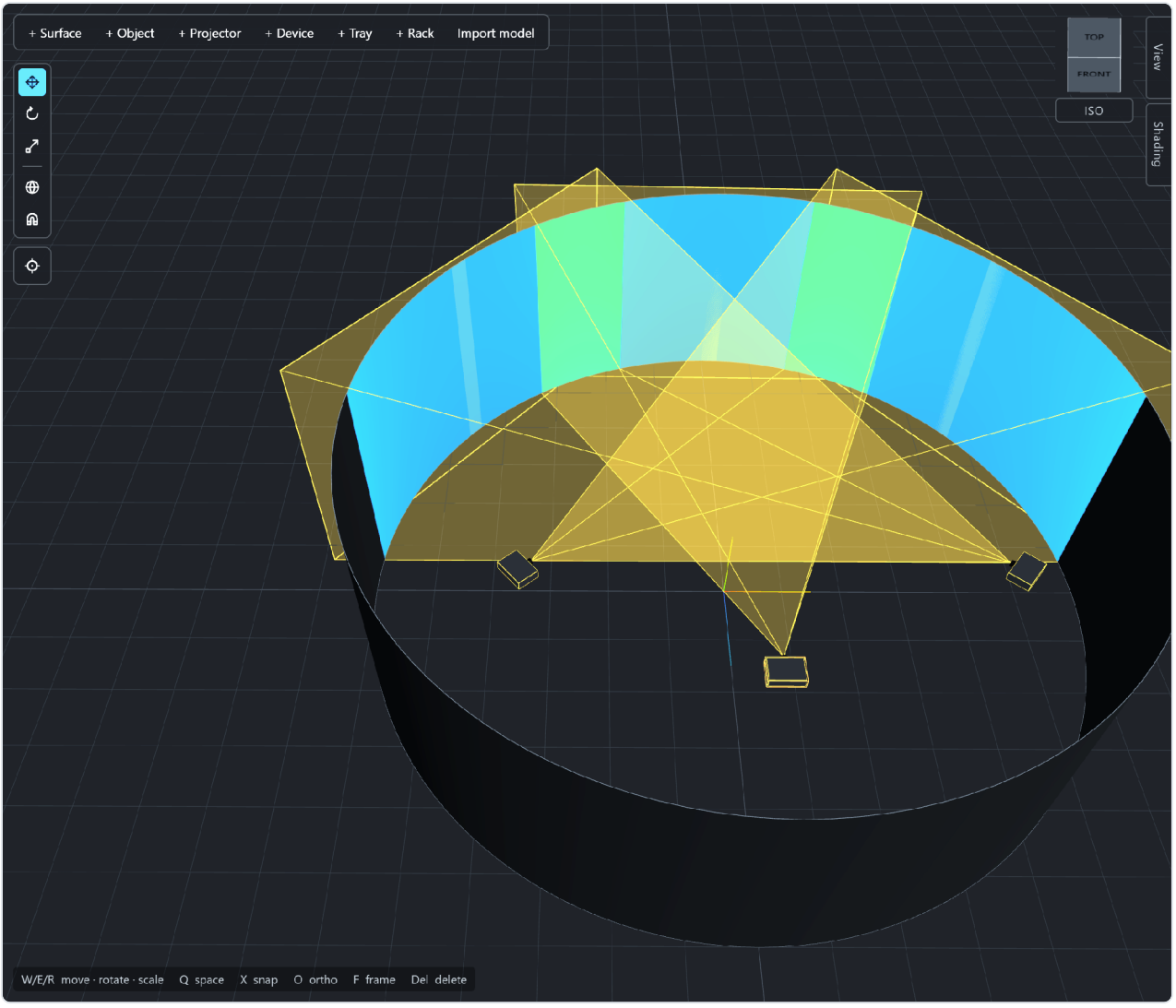


Figure 5.4a Three machines on a 100° arc inside a drum, throwing outward. The green bands are where neighbouring images meet — the blend seams the overlap percentage sets.

> **IT IS BORN WHERE THE MACHINE ALREADY STANDS**
>
> Aim one projector at the middle of the drum and press "**Circle**": the centre is seeded where the beam actually lands, and the radius is read off as the distance the machine already stands at. "**Create**" then rings the venue *without moving the projector you placed by hand*. Type a different centre or radius afterwards and the whole rig re-seats — including the source, because on a ring the **circle** is what holds still, not the anchor.

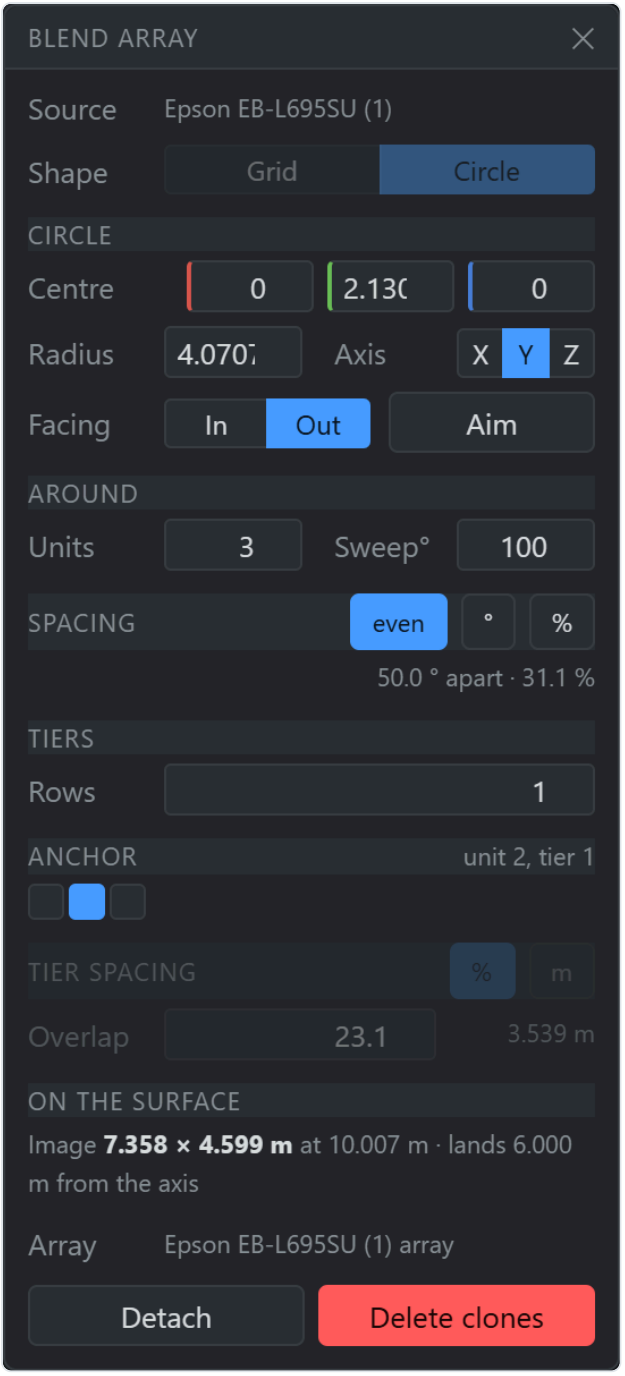


Figure 5.4b The ring controls. "**Centre**" and "**Radius**" place the circle; "**Axis**" is the line it turns about (Y is a level ring); "**Facing**" says whether the machines look at the centre or away from it, and "**Aim**" turns them onto it. "**Sweep°**" is the shape: 360 is a closed ring, anything less an arc. Here the source is the *middle* unit of three — "**unit 2, tier 1**" on the anchor pad.

Spacing round the circle

Three ways to drive it, and the two you are not driving are live readouts:

MODE	WHAT IT MEANS
"**even**"	Divide the sweep by the count. A closed ring always closes — twelve units over 360° sit 30° apart.
"**°**"	A typed angle between neighbours.
"**%**"	Solve the angle from a blend overlap, against the image the source actually makes.

"**Sweep°**" is what you *said* the rig is, and Throwcast leaves it alone. Type 25° into a twelve-unit closed ring and the panel tells you `12 × 25.0 ° covers 300.0 ° — a 60.0 ° gap against the 360 °`

`sweep` rather than quietly re-opening the circle behind you. Fix it by units, by angle, or by pressing "**even**".

> **WHERE THE LIGHT LANDS, NOT WHERE THE MACHINES ARE**
> An overlap is a length *on the screen*, so the angle that delivers it depends on how far from the axis the image falls — which the panel reports beside the image size (`lands 6.000 m from the axis`). Aim every unit straight at the axis and there is no arc to overlap along at all: the images stack on the centre line rather than blending, the "**%**" button greys out, and the panel says so.

Tilt, tiers and handles

- **Tilt is carried round, not coned.** Every unit is the source's pose swung rigidly about the axis, so a machine pitched 20° down is still pitched 20° down at the far side of the circle.
- **Rows are tiers** stacked along the axis, spaced by the same overlap-or-metres control a grid's rows use.
- **The handles are a cross through the source**: drag it to re-derive the radius and the start bearing (lift it and the whole ring rises with it), or drag an end of an open arc to re-space. A closed single-tier ring has no end to pull, so the source is its only handle and the count, sweep and radius are numbers in the panel.
- **A closed ring may only lose whole tiers.** Taking one unit out of a circle leaves a gap no circle describes, so you are asked to stand the rig down instead. An open arc can lose an end, exactly as a row can.

> **TWO ANCHOR EDITS, AND THEY ARE OPPOSITES**
> The **anchor pad** moves the source machine into a different slot and re-lays the rig *around* it — the source holds still, everybody else takes a new bearing. The "⚓" beside a member in the array inspector moves the *role* to another machine and nothing moves at all. Both leave the venue looking the same on a closed ring; what they change is which machine the rig is measured from next time.

5.5 The edge-overlap readout

Select the array in the browser and the inspector reports each adjacent pair.

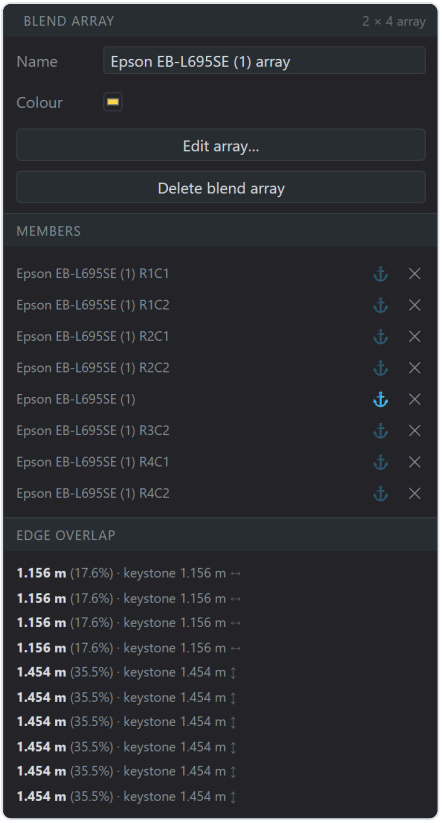


Figure 5.5 The array inspector: members, the anchor badge, and the measured overlap for every adjacent pair — `1.156 m (17.6 %)` across each row ("↔") and `1.454 m (35.5 %)` up each column ("↕"), each with its keystone figure beside it.

Adjacency is worked out locally — the nearest machine ahead along the run whose offset *across* the run is smaller — so a grid yields within-row and within-column pairs and no diagonals. A single line behaves exactly as you would expect.

A **ring** is read from its own spec instead, because no single straight axis describes a circle: the units face a dozen different ways onto a curve. That also means the seam where the last unit blends back onto the first is reported like any other — the pair a chain along an axis could never find.

> **ON A CURVE THE TRUE OVERLAP READS UNDER THE PLANAR ONE**
> On a flat wall the keystone figure comes out *above* the planar one. Round a drum it is the other way: each image is angled away from the straight line between two centres, so it only spans part of it. That shortfall is the curve, and it is the number the blend has to be trimmed to.

5.6 Drift, and re-solving

A clone is a clone. Change the source's throw ratio, device, lens, aspect or lens shift and every unit follows in the same undo entry — shift included, because shift is how a rig is *aimed*, and a rig is aimed as one thing: drop the truss and they all shift up together. A clone you then trim by hand keeps that trim until you edit the machine it is a copy of, which is the moment you meant the whole rig to follow.

Position is a separate decision. Re-stamping the optics does not move anything — changing the picture and moving eight machines are two different intentions — so the spacing stays where it is and no longer matches the intent you set. Throwcast **reports that rather than silently**

correcting it: the panel says something like `H is −28.0 % against a 20.0 % target` and offers "**Re-solve**", which re-spaces to the target.

5.7 Editing and standing down

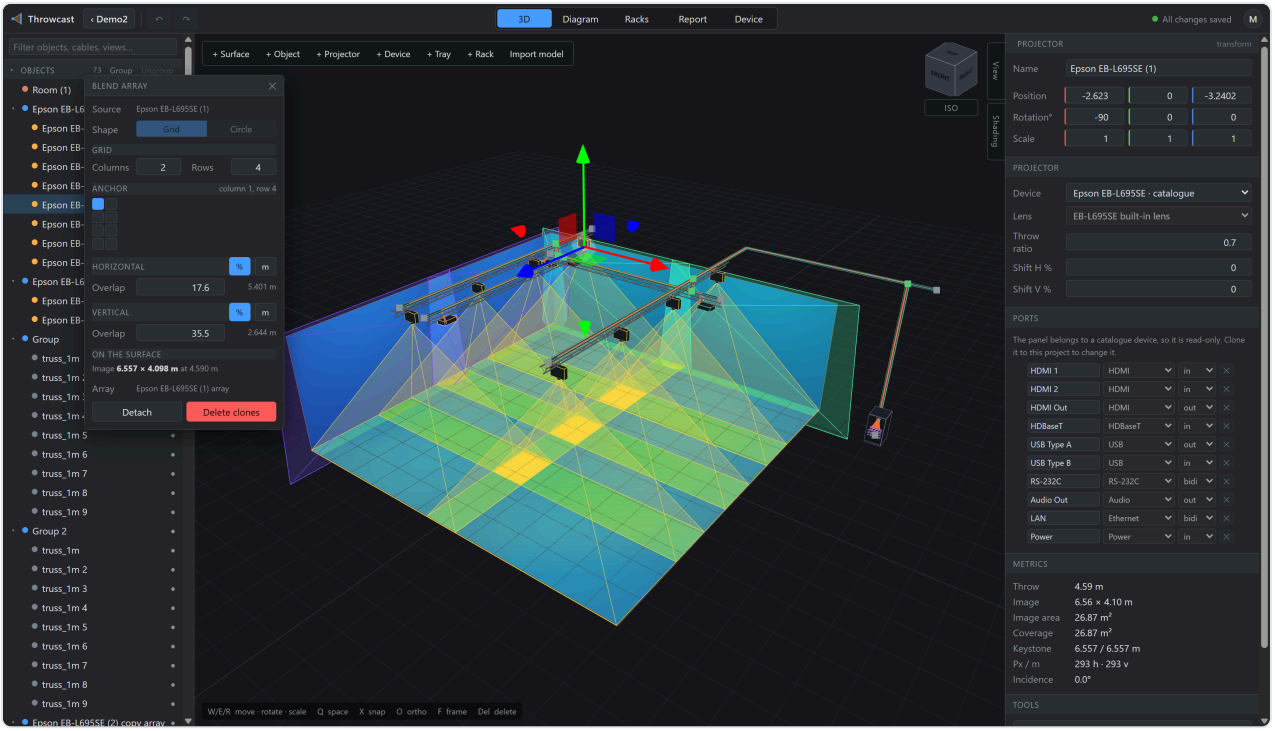


Figure 5.7 The same panel once the array exists: "**Detach**" stands the rig down and keeps every machine where it is; "**Delete clones**" removes them.

- **Move one unit by hand** and it stays where you put it, reported as off the array with a "**Re-apply**" offered. Hand trim survives until you ask for it not to.
- **Change the column, row or unit count** to resize by the numbers. Survivors keep their offsets and nothing moves — and on a ring driven "**even**", changing the count re-divides the sweep so the circle still closes.
- **Delete a unit, or drag one out of the rig's folder**, and if that would tear the grid you are asked once: stand the whole array down, or abandon the edit. It is one question because it is one decision.
- **Standing down keeps the folder.** The blend group goes; the scene group holding the machines — which you may have renamed — stays.

CHAPTER 6

Simulation

Two pictures of the same physics: a false-colour map of how much light lands where, and a preview of the actual image. Both are real-time, and neither needs baking.

6.1 Shading modes

Open the "**Shading**" edge tab.

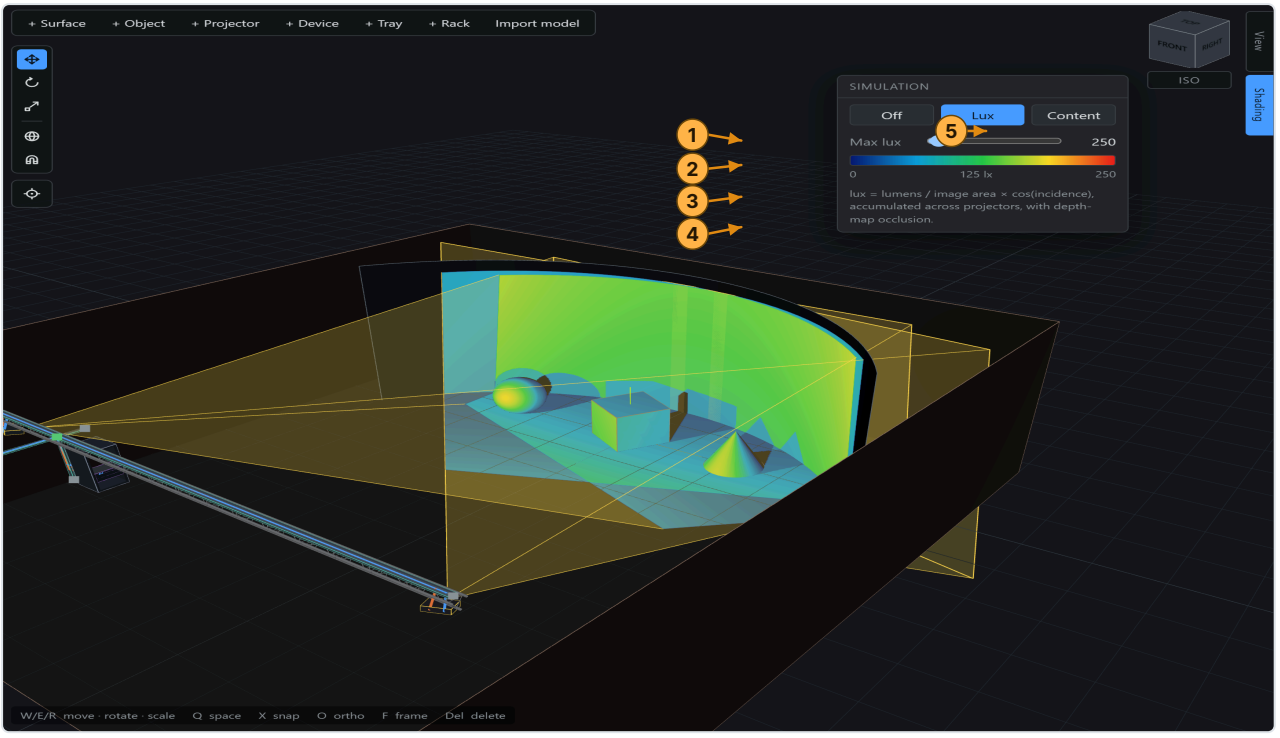


Figure 6.1 The simulation panel, with the heatmap on the demo rig.

1. **Mode** — Off, Lux, or Content.
2. **Max lux** — the top of the colour scale, 100 to 5000.
3. **Legend** — the colour ramp with its numbers.
4. **The formula** — stated on the panel, so the picture is never a mystery.
5. **The tab** — lit because the panel is open, not because a mode is on.

6.2 The lux heatmap

> **WHAT IT SHOWS**
>
> The light actually landing on a surface: each machine's lumens spread over the image it makes *there*, discounted for a beam that arrives at a slant, and added up over every projector that reaches the point. Whatever is in shadow gets none of it.

Three consequences worth internalising:

- **Bigger image, dimmer picture.** The same lumens are spread over more surface, and it is the *area* that grows — so brightness falls away faster than the extra width suggests.
- **Angle costs light.** The more obliquely a beam lands, the further it is smeared across the surface, and the dimmer it reads.
- **Overlaps add.** A blend region gets both machines' light, which is exactly why it shows up hotter.

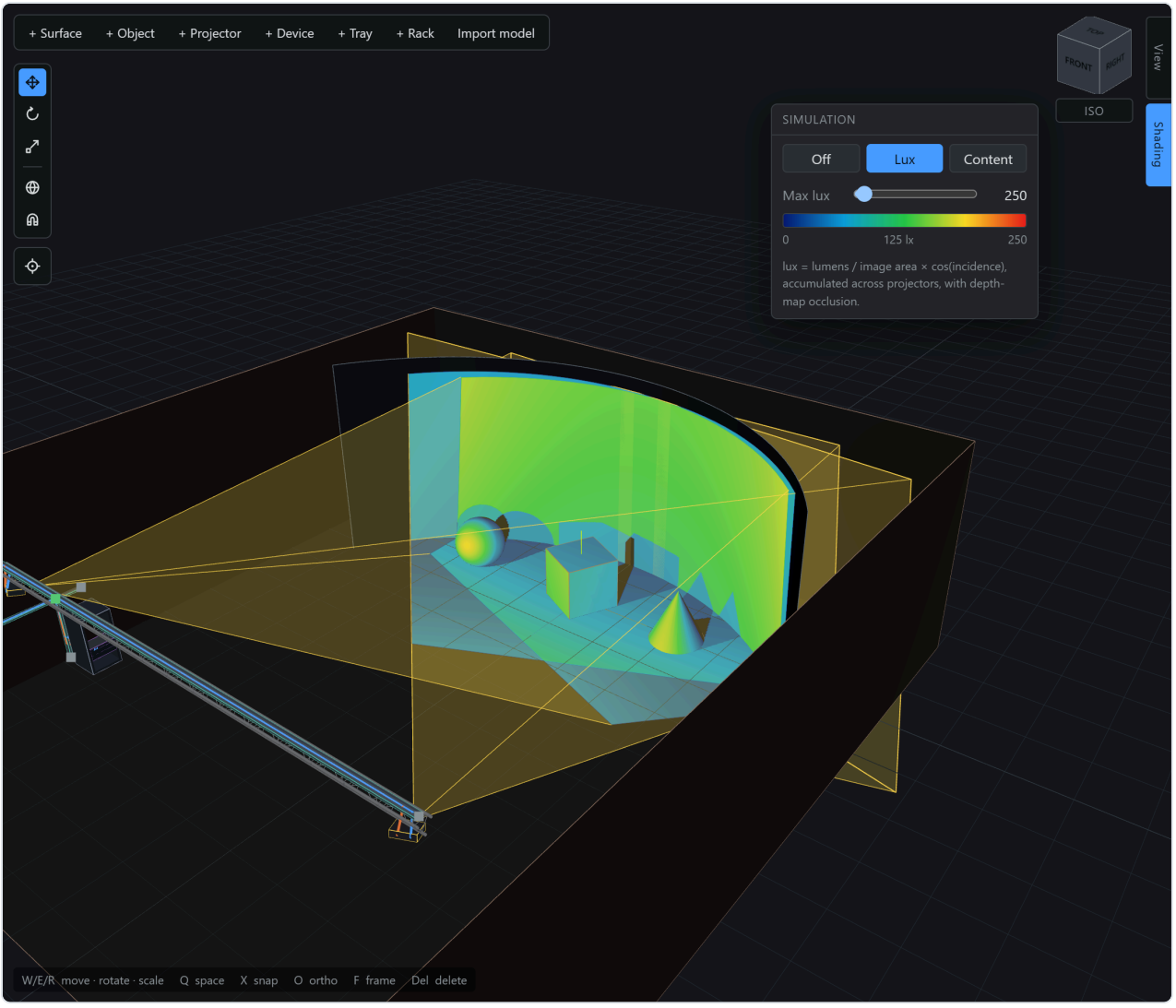


Figure 6.2 The demo rig's main screen. The two images read the same green; the vertical band where they overlap is visibly hotter — that is the blend region, carrying both machines' light.

Set **Max lux** to the top of the range you care about. The default of 1000 lx puts a typical rig's overlap bands in the middle of the scale.

6.3 Content preview

Switch to "**Content**" and each projector projects an image instead of a colour. With no image loaded you get a test card whose corner markers land exactly on the frustum corners, so keystone stretch is visible directly.

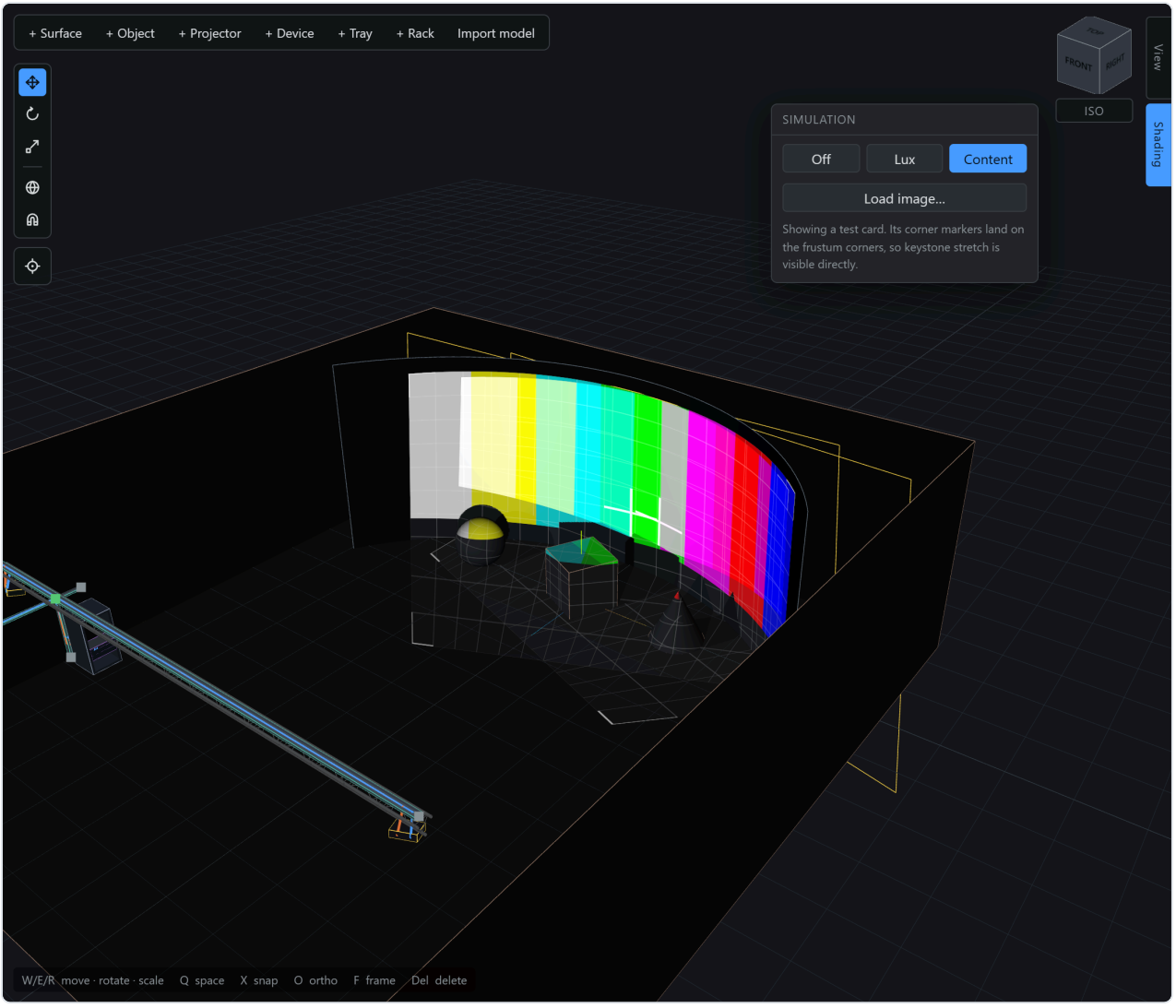


Figure 6.3 Content preview on the blended pair. Both test cards are visible, and the region where they overlap is brighter and doubled — which is what an un-blended overlap looks like in the room.

"**Load image…**" replaces the test card with your own still. The image is stored by content hash rather than inside the project, so a large still does not get re-uploaded every time you nudge a projector.

6.4 Occlusion and light-tight bodies

The simulation renders a depth map per projector, so anything in the way casts a real shadow — a truss, a column, a person-sized box.

> **THE RULE**
>
> Bodies are **light-tight per projector**: each machine lights only the face turned toward it, and you see that light only from the same side. Rear-projection screens opt out. The rule does not depend on which way a surface normal happens to point, which is what makes it safe on imported venue CAD, where normals are often inconsistent.

Openings need no special handling for the same reason. Cut a doorway (§3.14) and the light goes through it, the wall beside it still shadows, and the bright rectangle lands on the floor

beyond — a hole is an absence of triangles, and the depth map simply does not find one there.

6.5 The hover probe

Turn the probe on from the rail on the left of the viewport, then move the cursor over any surface.

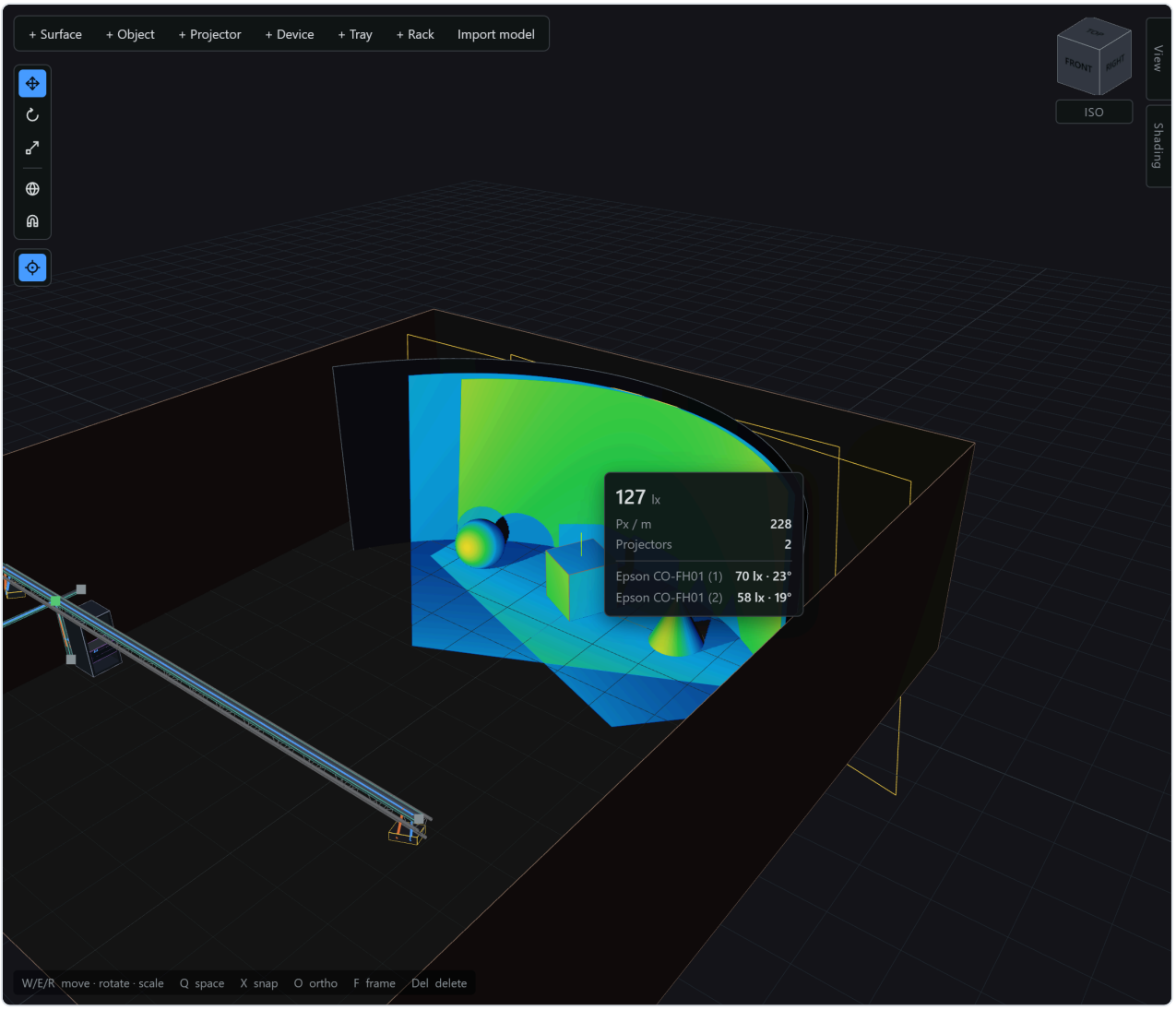


Figure 6.5 The probe: total illuminance, best available pixel density, how many projectors reach the point, and each machine's contribution with its incidence angle. Here 127 lx at a point both machines reach — 70 lx from "**Epson CO-FH01 (1)**" at 23° and 58 lx from "**(2)**" at 19°.

The per-projector breakdown is the useful part. In a blend it tells you how the total is being split; on a spill it tells you which machine is responsible.

Where nothing reaches, the probe says "**No light reaches this point**" rather than reporting zero — a shadow and an unlit surface are different problems.

CHAPTER 7

The device library

The Device tab answers a different question from the rest of the app: not *where is this machine*, but *what is this machine*.

7.1 Definitions and instances

A **definition** is a model — a body, its optics, its lens options, its socket panel. An **instance** is one of those placed in the venue, carrying only what is true of that unit: where it is, which lens is mounted, what throw ratio and shift it is set to.

Editing a definition updates every instance of it. That is how “edit one, update thirty” works.

7.2 The Device tab

Device mode replaces both side panels: the left becomes the library of definitions *this project owns*, the right becomes the editor for the selected one.

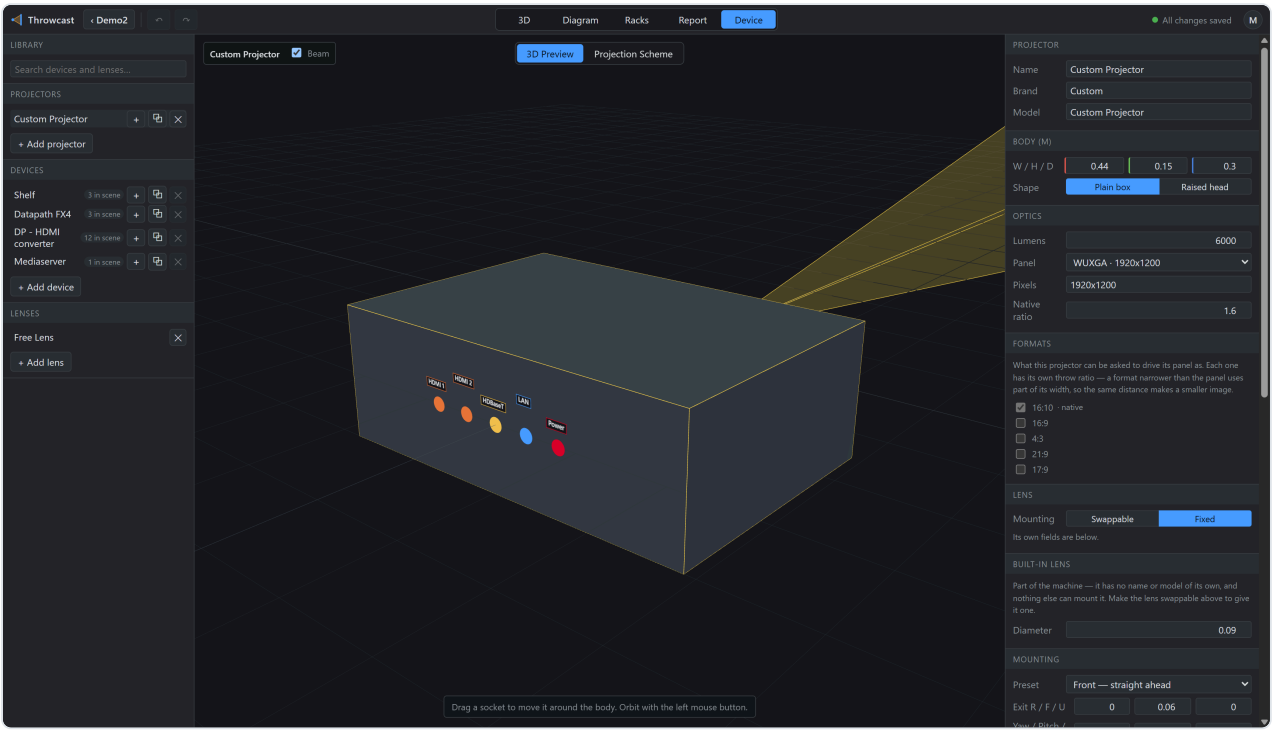


Figure 7.2 The Device tab. Left: projectors, devices and lenses this project owns, each row showing how many are in the scene. Centre: a 3D preview with the socket panel laid out on the body. Right: the definition editor.

The editor covers the body (width, height, depth, and whether the machine wears a raised ultra-short-throw head), the optics (lumens, panel, native ratio), the image **formats** the panel can be driven in, lens mounting (fixed or swappable, and which lenses are compatible), and the ports.

A lens is authored as a **casting**: which of four shapes it is — a straight barrel, an elbow that turns the light 90°, a rear fold that throws back over the cabinet, or an ultra-short-throw — and how long each of its sections is. Where the beam starts and which way it leaves (§4.4) is *derived* from that, rather than typed as a set of offsets: say what the part is and the light leaves where the part says it does. Where it bolts on is the *machine's* figure, not the lens's — one "**Lens mount R / U**" on the body, so a lens that fits two chassis does not carry one of them around.

> **WHY FORMATS MATTER**
> A projector does not have "an aspect ratio" — it has a panel and a list of formats it can drive it in, and **the throw ratio changes with the choice**. Driving 4:3 on a 16:10 panel uses only part of the panel's width, so the same distance makes a narrower image and the ratio rises by exactly `panel ÷ chosen`. Vendors tabulate a separate throw-ratio column per format for this reason, and so does Throwcast.

7.3 The catalogue

The bundled catalogue is global and read-only. The two "**+ Add …**" buttons in the Device tab ask a different question from "**+ Projector**" in the 3D tab:

Table 7.1 Two questions, one dialog.

WHERE	ASKS	OFFERS
3D tab, "**+ Projector**"	Which definition do I place?	This project + Catalogue
Device tab, "**+ Add …**"	What new definition do I create?	Custom + Catalogue (cloned to edit)

Cloning from the catalogue copies it under a fresh id, so your edits never touch the catalogue and a project stays self-contained.

7.4 Ports

A socket's position is part of the definition: a face, plus an offset from that face's centre. Storing it that way means a body that gets resized *clamps* its panel back on rather than leaving sockets floating in mid-air.

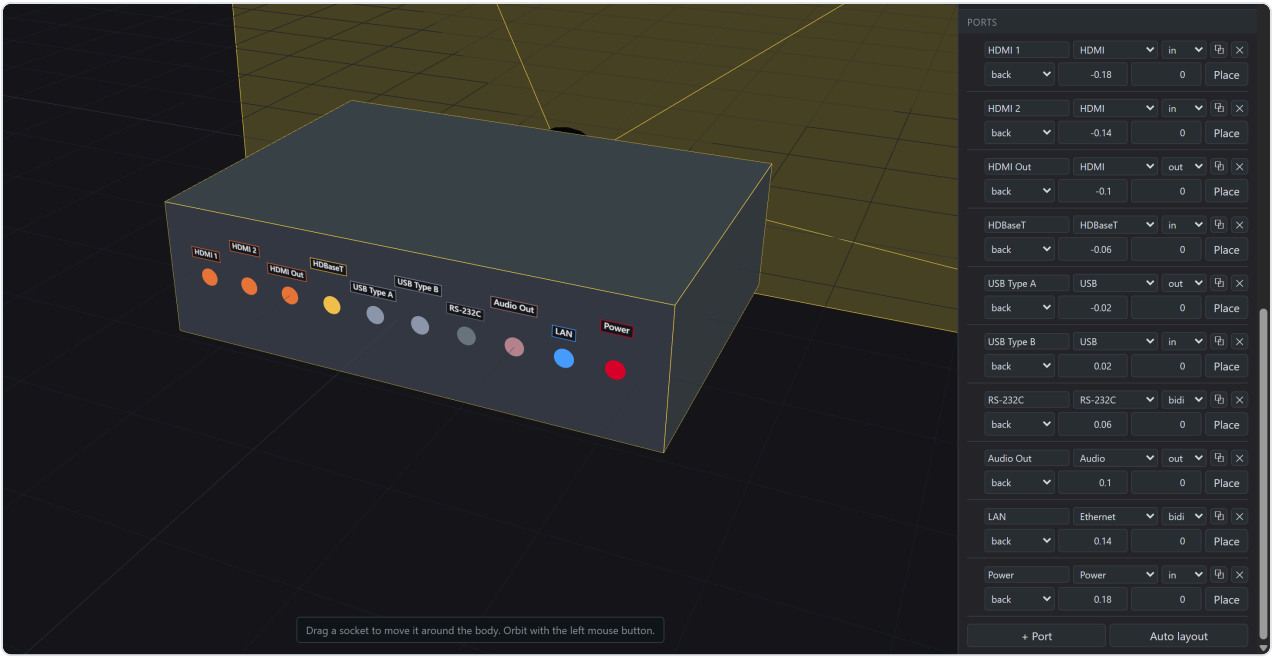


Figure 7.4 The EB-L695SE's panel in both the places it exists: ten sockets on the machine's own back face, and the rows that put them there. Each row takes two lines — a name, its type and its direction, then the face it sits on and its offset across and up from that face's centre. The two are one thing seen twice: **drag a socket on the body and the numbers follow it**, "**Place**" arms a click for putting one down by pointing, and "⧉" duplicates a socket beside its twin — which is how a row of four HDMIs gets built. The preview opens on the machine's front; this is it orbited round to the back, where this projector keeps everything.

Faces are measured on the cabinet. A projector whose lens head stands proud of the cabinet has a back panel only cabinet-high, and the head's own faces are not offered.

"**Auto layout**" in the Ports section re-spreads every socket across the back to fit the body's current size: one centred row at 40 mm pitch, wrapping to further rows when the panel is wider than the machine. It is a button rather than an automatic response to a dimension edit, so a panel someone arranged by hand stays arranged.

7.5 The projection scheme

For a projector or a lens, the centre of the Device tab offers a second tab beside the 3D preview: a vendor-style **Projection Scheme** — a 2D throw diagram in side and top elevation, with dimension lines.

It is driven by a calculator that is not part of the project: distance and image width solve each other, zoom slides through the throw-ratio range, and shift and aspect are live. A "**Fix**" pin on either distance or image width makes zoom drive only the other, which goes read-only.

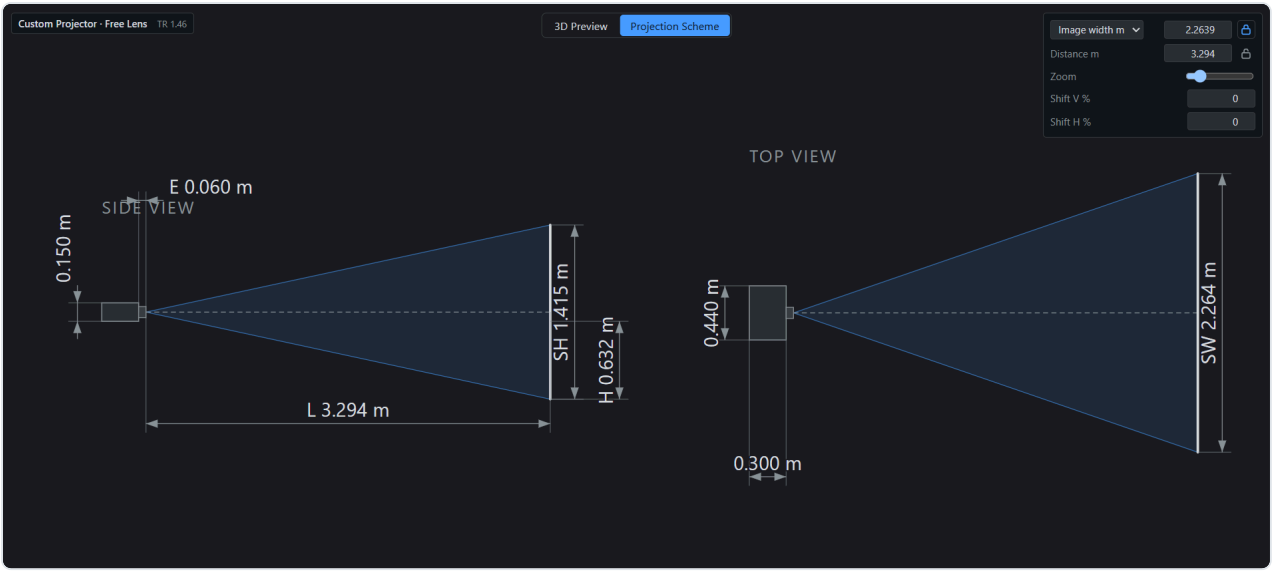


Figure 7.5 A 100″ image from an EB-L695SE, its zoom halfway along the range at TR 0.60 — so a 2.154 m picture from `L 1.292 m`. The side elevation dimensions the throw, the image height and the exit; the plan does the same across the room. Both are drawn from the machine's own figures, which is why the cabinet's 0.440 × 0.122 m face and its 0.304 m depth are on the drawing too. The padlock beside the size is the pin: with it down, zoom moves the distance and leaves the picture alone.

Folded lenses are drawn honestly: an ultra-short-throw marks its exit window, a right-angle lens shows the cabinet side-on to the beam, and a rear-throwing lens puts the chassis between the apex and the screen.

The canvas is a viewport, not a picture — the wheel zooms at the cursor while text stays at screen size, dragging pans, each elevation can be moved by its caption, and a double-click refits.

7.6 Solving a scheme from vendor data

A machine the catalogue does not carry still has to throw the right picture, and the numbers the app needs are not the numbers a vendor prints. Throwcast wants a throw ratio and an exit point — where in the machine the light appears to come from. A datasheet gives you a **table**: at this screen size, stand it this far away. A throw calculator gives you the same thing one size at a time.

The **projection-scheme wizard** converts the second into the first. Open the Device tab, select the projector — or, where the lens comes off, the lens — and in the "**Mounting**" section press "**Solve from vendor data…**". It runs in four steps: the machine, the lens's shape, the scheme itself, and a review.

> **THE RULE**
>
> **Vendor data in, one undo entry out.** Every number you type lives in the dialog until "**Apply**", which writes the lens *and* any body edits the first step staged as a single change. One `Ctrl` + `Z` takes all of it back off.

Step 1 — the machine. Every distance ahead is measured against a face of this box, and the exit the wizard solves is placed from its lens mount, so the box is asked about first. It is drawn in

proportion, with the mount marked and the raised head, on a machine that wears one, standing on the cabinet. The figures are editable here, which matters more than it sounds: a fit solved against the wrong depth lands the apex a body's length out. A lens welded into a machine is pinned to it; a lens that comes off gets the picker in the figure below, because the same lens on a different chassis is a different fit.

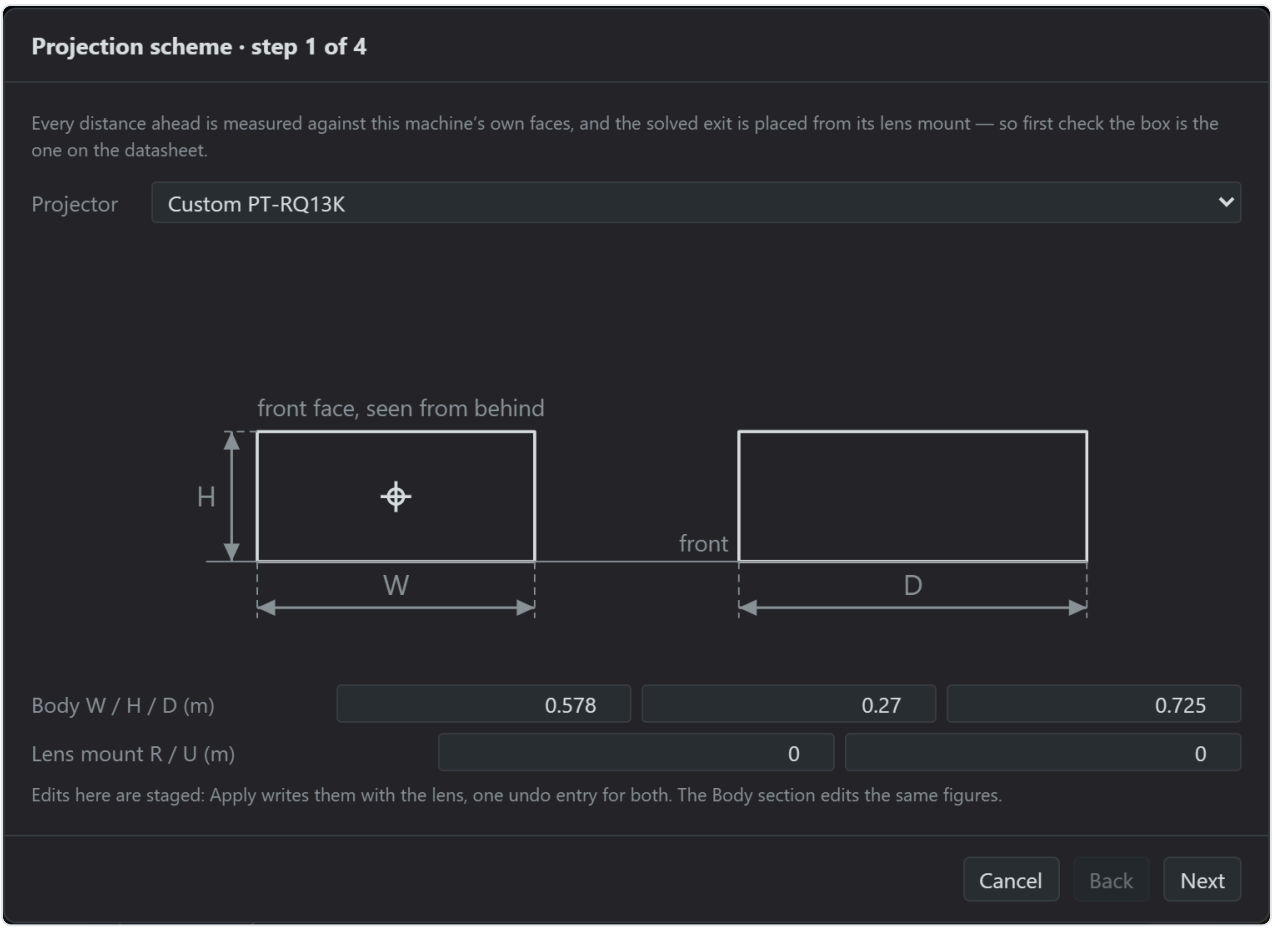


Figure 7.6 Step 1, on a PT-RQ13K. The body the fit will stand on, drawn to its own proportions — front face and side elevation, with the lens mount marked. Edits here are staged, not written.

> **A CATALOGUE MACHINE IS READ-ONLY HERE TOO**
>
> If the host is a catalogue entry its figures are disabled behind "**Clone for editing**", which is the Device tab's own clone (§7.3) — fresh id, your project, the catalogue untouched. The wizard then follows the clone: the picker, the drawings and the fit all move to it, and cancelling afterwards keeps it, since you can delete it from the library like anything else.

Step 2 — the shape. Four cards, because four shapes cover what a fit can actually solve. The choice sets which way the beam leaves the machine, which face a vendor's distances are conventionally measured to, and which columns the next step asks for. They are the subject of the next section.

7.7 The four shapes

A lens is one of four castings. Two of them are the same part hung different ways round, so picking those cards reveals a segment for the handedness — a card each would have been four cards for one lens.

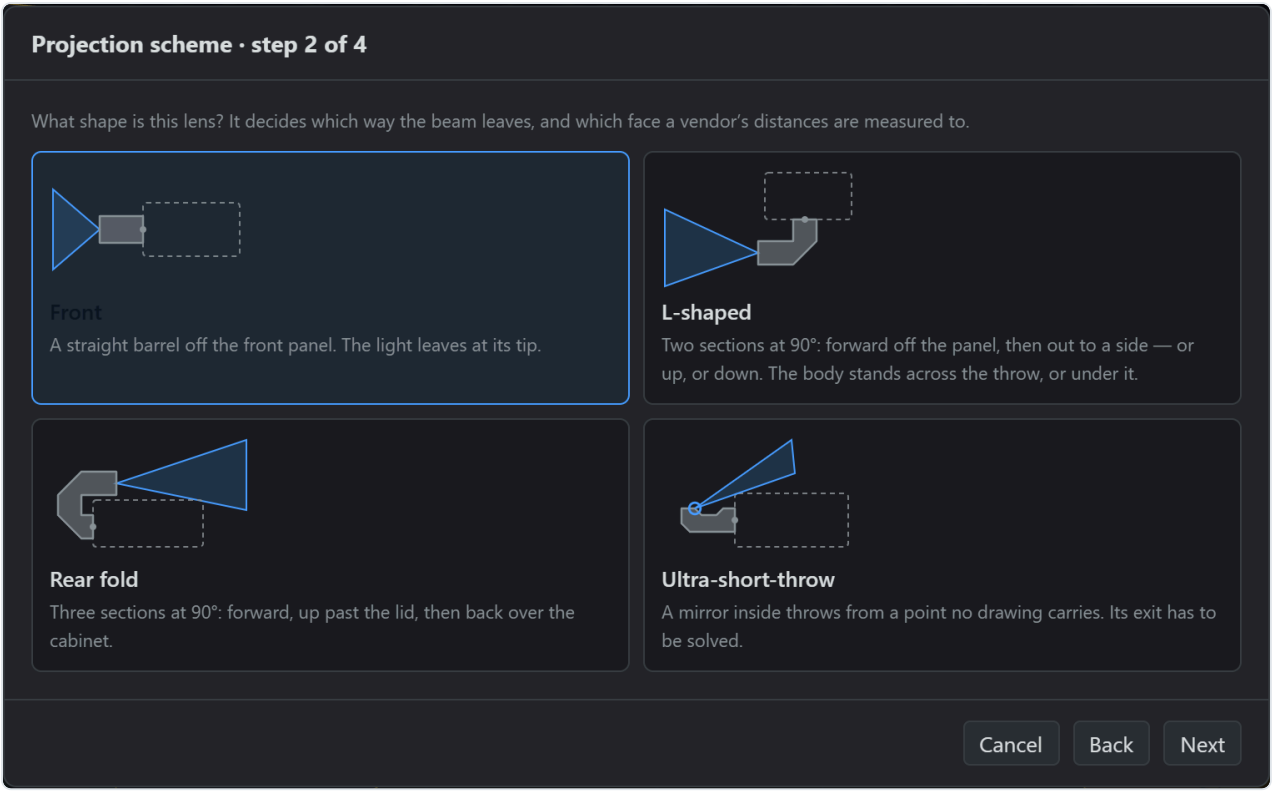


Figure 7.7a **Front** — a straight barrel off the front panel, the light leaving at its tip. Every ordinary zoom and prime, and the shape to pick when nothing about the machine turns the beam. It is asked for a distance column, a "**Tele**" column if the lens zooms, and optionally where the image can sit.

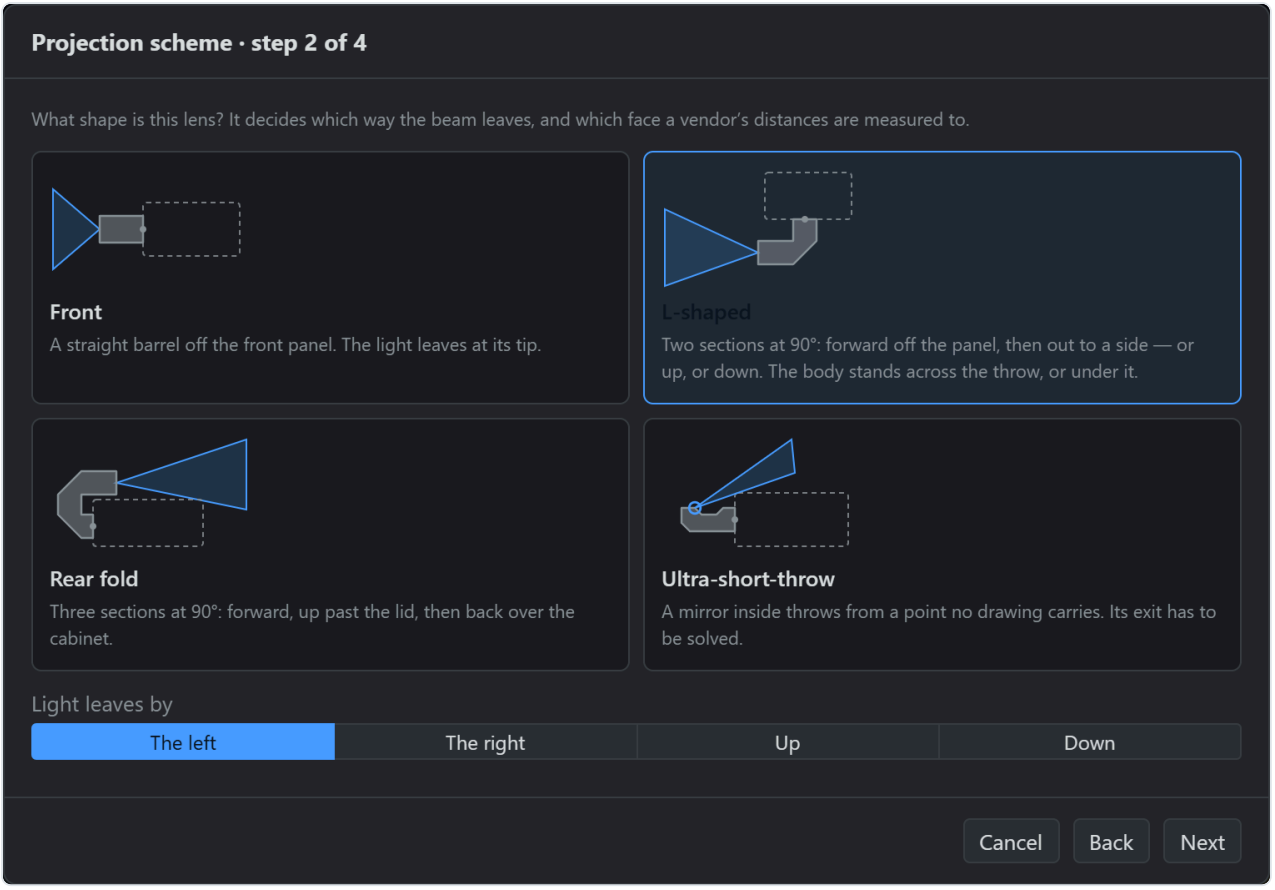


Figure 7.7b **L-shaped** — two sections at 90°: forward off the panel, then out to a side. Picking it reveals "**Light leaves by**", four ways to hang one part: the left, the right, up or down. Its two legs are typed, and they are the authority for the geometry — the fit contributes the throw ratio and a cross-check. It is the one shape asked nothing about the vertical, its vertical being its own arm.

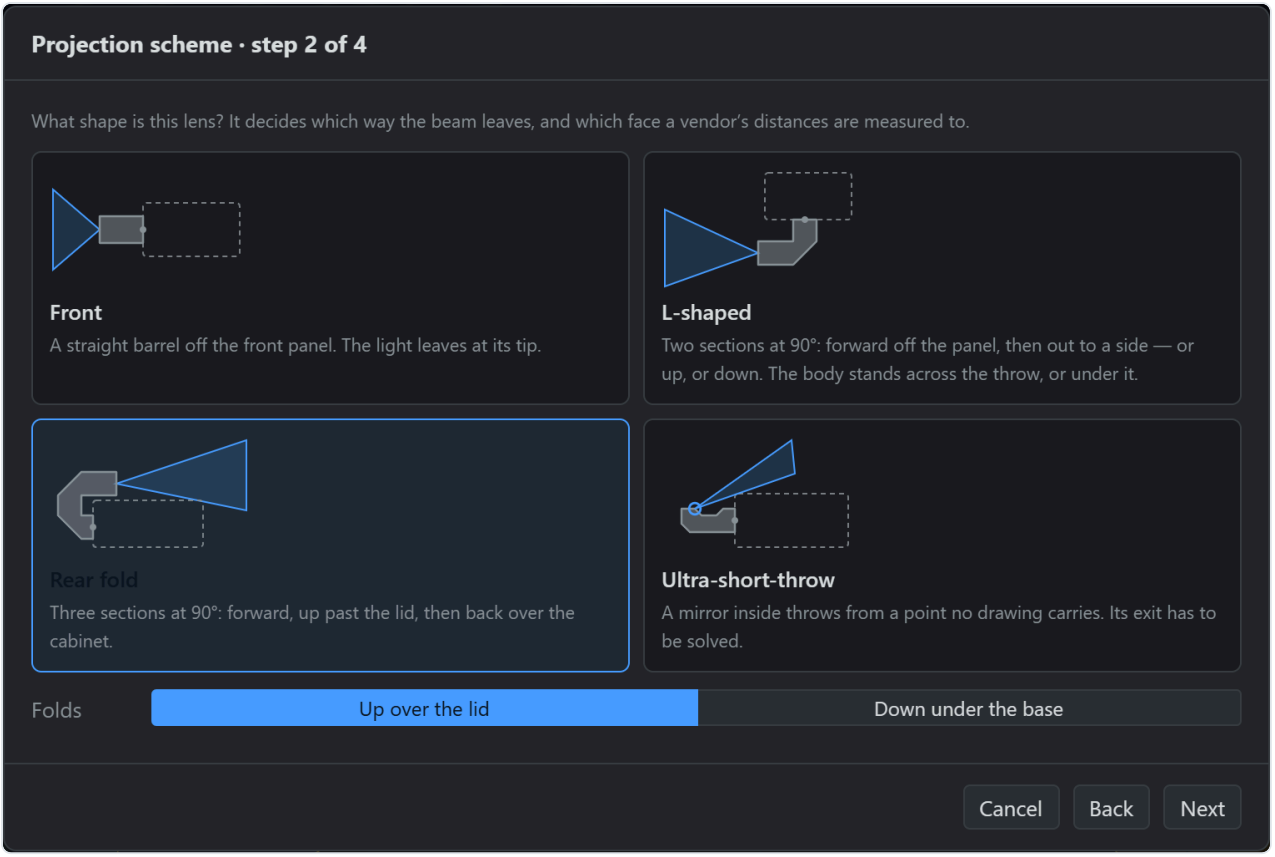


Figure 7.7c **Rear fold** — three sections at 90°: forward, up past the lid, then back over the cabinet, so the light leaves the far end and flies back over the machine it came out of. "**Folds**" chooses up over the lid or down under the base. All three legs are typed; the glass sits at leg 1 minus leg 3 from the front face, which is why this family is not *also* asked where its lens is.

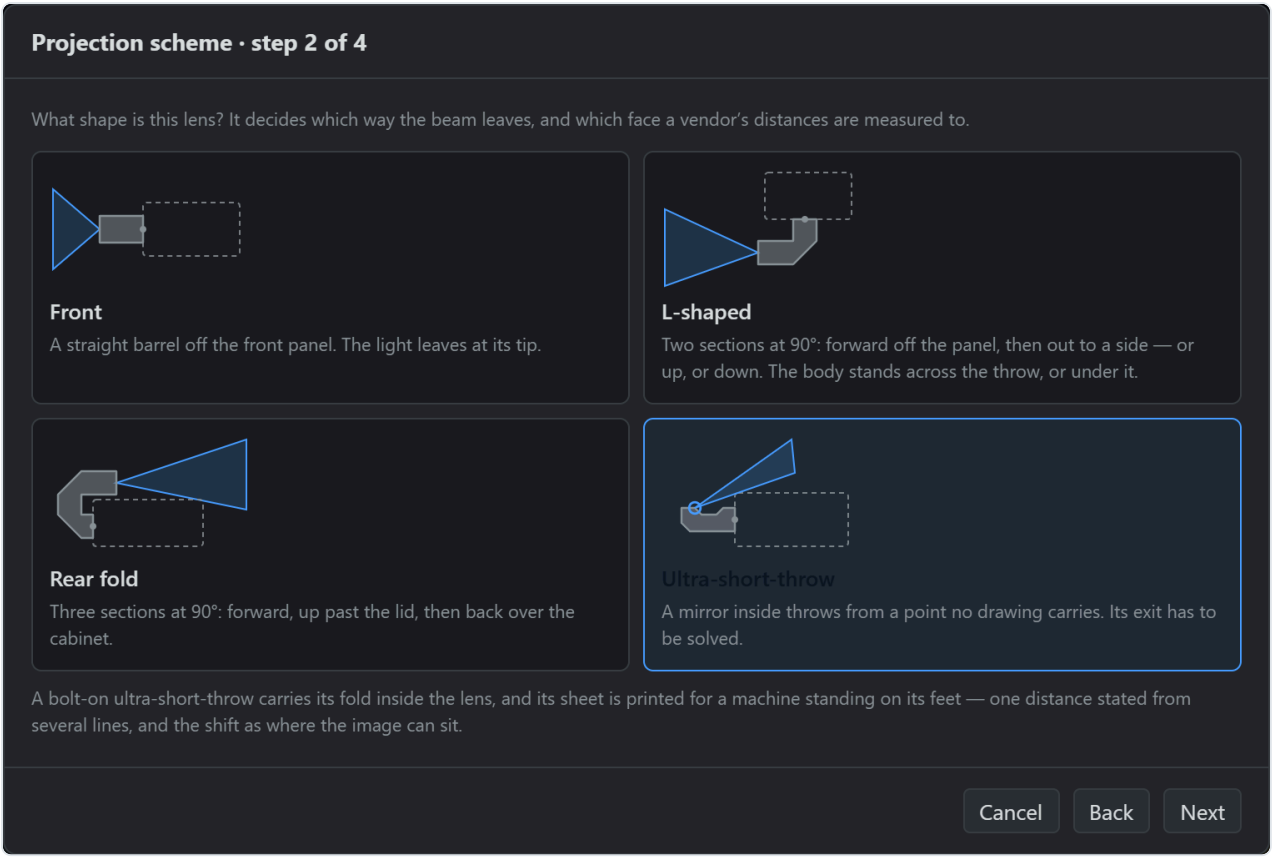


Figure 7.7d **Ultra-short-throw** — a mirror inside throws from a point no drawing carries, which is exactly the point the wizard exists to solve. Whether it is welded into the body or bolted onto a lens mount is read from the machine, not asked: the two are printed from different conventions, and they get different columns.

What each is asked for follows from the sheets it is transcribed from — and a column a family could never fill is one it is never shown:

Table 7.2 What the third step offers, by shape.

SHAPE	TYPED SECTIONS	COLUMNS IT GETS
"**Front**"	None — the barrel is what the fit solves.	Distance, Tele, and the image-position pair.
"**L-shaped**"	Leg 1 mount → elbow, leg 2 elbow → glass, with its own datum.	Distance and Tele, measured to the near face, the lens tip or the far face.
"**Rear fold**"	Three legs — forward, up past the lid, back over the cabinet.	Distance, Tele, image position.
"**Ultra-short-throw**", welded in	The barrel that gets drawn.	Distance, Tele, **Drop** and "**Plate → datum**".
"**Ultra-short-throw**", bolted on	Barrel (front face → throw point) and Head (past it to the end).	Distance and image position. No Tele — every lens of this kind is a prime — and no Drop.

Where sections are typed they are the **authority**, not a hint: the casting is what you typed, the beam is derived from it, and the rows contribute the throw ratio, the zoom walk and a cross-check. Where the rows put the throw point somewhere the legs do not, the disagreement is reported in millimetres rather than resolved silently in either direction.

WHY A FIXED-LENS MACHINE IS OFFERED ONLY TWO

A machine whose lens is welded in gets "**Front**" and "**Ultra-short-throw**" — no elbow, no rear fold. Those are lenses, not bodies: parts that bolt to a lens mount, which is why they come off by construction. Both bundled catalogues agree — 98 bodies carry a welded-in lens and not one of them turns its light, while all 17 right-angle entries are separate lenses. Offering the card there would offer a machine nobody builds, and picking it would silently yaw the beam 90° into a wall. A layout the four cards cannot express is still typed by hand into the Mounting rows.

7.8 The table, the fit and the review

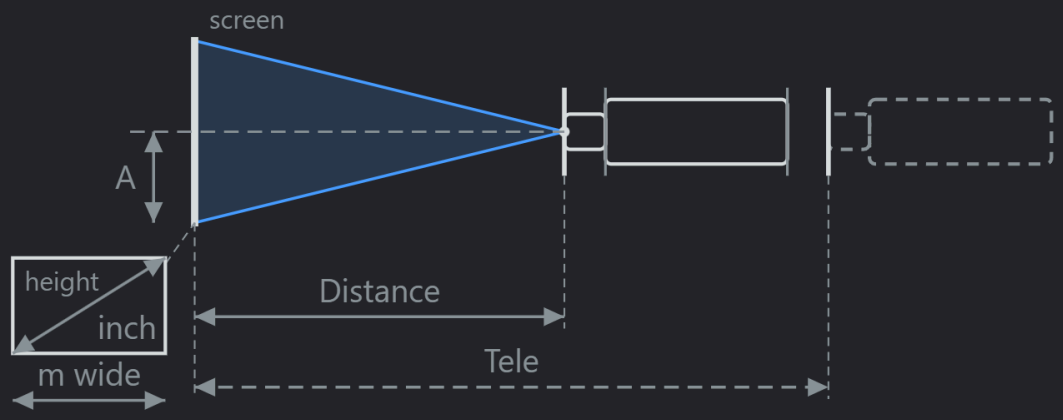
Step 3 is where the paperwork gets typed. "**Working from**" switches between "**A distance table**" and "**I know the throw ratio**" — the second for when the number is simply published, and it takes the ratio at both ends of the zoom, the park, the shift travel as typed percentages, and the casting's sections. "**Lengths in**" opens on millimetres, because that is what a drawing prints; switching converts what you have already entered rather than reinterpreting it.

Above the columns is a drawing of the machine with the run each column names dimensioned on it. Point at a header — or land in one of its cells — and that run lights up. It is schematic and never to scale: the question it answers is *which run is this*, not how long. The faces a distance can be measured to carry clickable markers, the chosen one solid, so "**Distance measured to**" can be answered on the drawing instead of in the dropdown.

Projection scheme · step 3 of 4

Working from: A distance table | I know the throw ratio

Lengths in: metres



Size	The screen that row is for, read as the unit beside it says: a diagonal in inches, or the image width in metres. Never its height.
Distance	What the guide prints against that screen, zoom at its wide end: the lens tip out to the surface.
Tele	The same row with the zoom at its long end — the same picture, from further back. Leave it empty on a prime.
A min / max	Where the image can sit against that row's screen: the printed pair from the chosen line up to the image's bottom edge or centre, machine standing on its feet. A range here IS the vertical shift travel, stated as geometry — it writes the lens's travel, not its park — and pairs at two sizes pin the axis height with it.
Lateral	The same across the room, when the guide prints one: image centre from the body's centreline, + to the machine's own right. Solves the horizontal travel — a fold swaps the sign itself, since its beam-right is the body's left.

Two rows solve it; more tighten it.

Size		Format	Distance	Tele	A min	A max	
200	inch	16:10(WQXGA)	7.70	11.20	—	—	×
400	inch	16:10(WQXGA)	15.48	22.52	—	—	×

+ Row

"Distance" measured to: The lens tip

Lens from front (m): 0.121

The distance stops at the glass, so this is what carries it the rest of the way onto the body: the vendor's *Protrudes* figure on a normal lens, and a negative number for a lens recessed into the cabinet — a well under the lid is behind the front face, not ahead of it.

Image position (optional) — when the guide prints where the image can sit rather than a shift percent: one pair per screen size, in the A columns above, machine standing on its feet; one value twice for a fixed position. This solves the shift *travel* — and pairs at two sizes pin the axis height with it.

Measured from: The lens axis

To the image: bottom edge

Lateral min / max (m):

Throw ratio 1.806 – 2.6278 against 16:10(WQXGA)
Apex R 0.0 mm · F 41.0 mm · U 0.0 mm
Worst miss 0.0 mm on distance — against the rows you typed

Cancel | Back | Next

Figure 7.8a Step 3 for a **front** lens, with two rows off the vendor's throw calculator. The drawing dimensions Distance and Tele — the same picture from two camera positions, which is what a zoom range is — and the inset draws the screen square-on, since an elevation sees it edge-on and could only ever dimension its height. The readout at the foot is the fit.

Two rows solve it; more tighten it. Two points fix a line and nothing fewer does — one row cannot separate a machine that throws wide from one that stands closer.

WORKED EXAMPLE — AN ET-D75LE20 ON A PT-RQ13K, FROM THE VENDOR'S CALCULATOR

Rows (16:10, metres, to the lens tip)	`200″ → 7.70 · tele 11.20  ·  400″ → 15.48 · tele 22.52`
Solved throw ratio	`1.806 – 2.6278`
Solved apex, past the front face	`41.0 mm`
Worst miss	`0.0 mm`
The catalogue's own entry	`1.806 – 2.6278, apex 37.8 mm`

The throw ratio lands exactly on the shipped one, and the apex 3 mm past it. That 3 mm is the calculator's own two-decimal printing: the same rounding on both rows is a *constant*, and a constant moves the datum rather than bending the line — which is why the fit can miss the rows you typed by nothing at all and still sit three millimetres from the entry the same lens ships with.

The readout shows the throw ratio, the solved apex, the shift where the columns solve one, and the **worst miss** in millimetres — how far the fitted line sits from the rows you typed.

A BAD FIT IS SHOWN, NEVER BLOCKED

"**Next**" is gated on a fit *existing*, not on it being good. A large miss is evidence, and it usually diagnoses one thing: the guide measures to a different face from the one you picked. Blocking would hide the very number that tells you so — move the datum and watch it collapse. The example above is the same story in miniature: had those rows been read to the front face instead of the lens tip, the apex would have landed 80 mm *inside* the machine, with the worst miss still reading zero.

The same step, the other three shapes

Step 2's card is not a label — it is what this step becomes. Each family gets the drawing its own guides are printed against, the columns those guides carry, and the datums that make sense of them. Below is the step as it opens for the other three, before a row is typed.

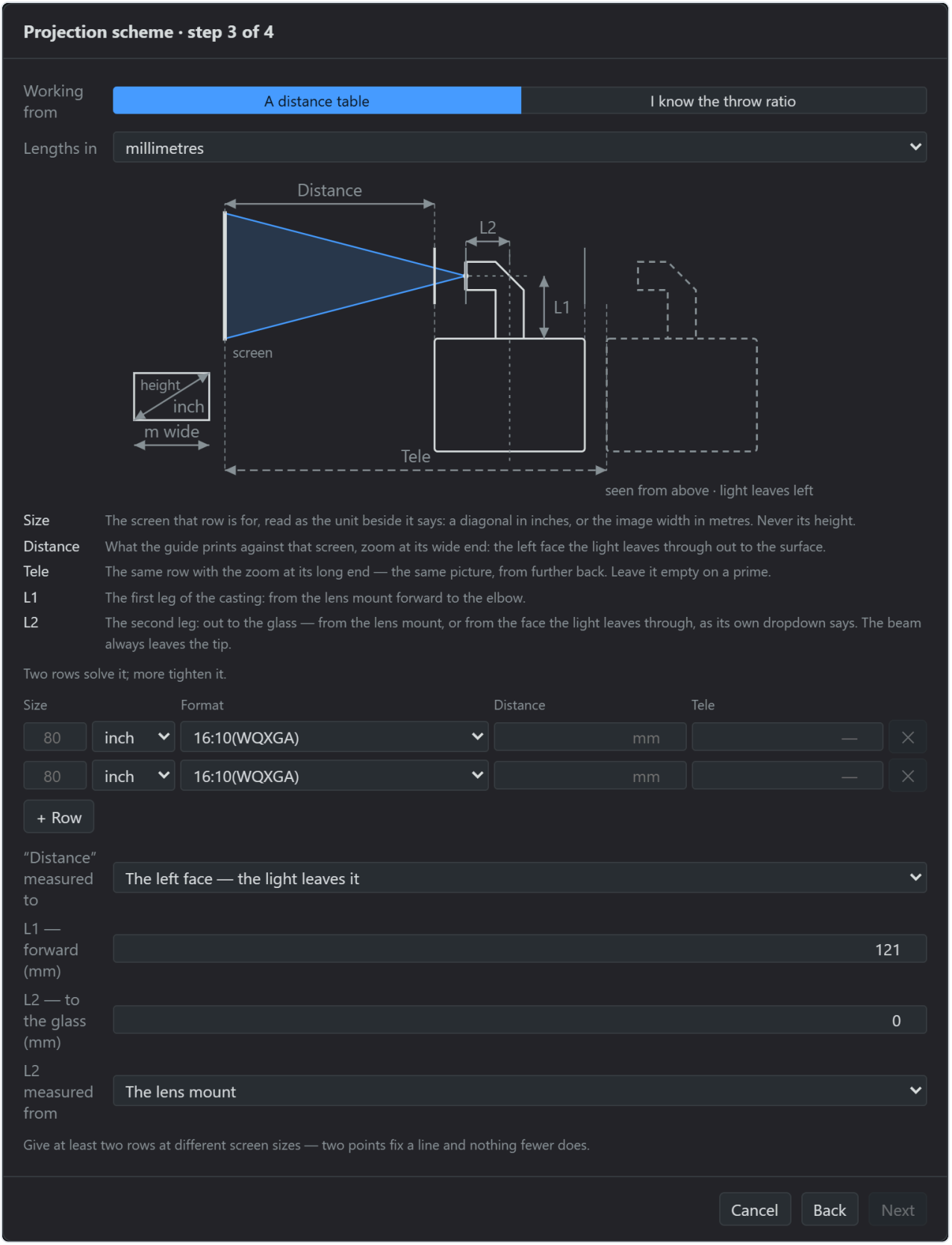
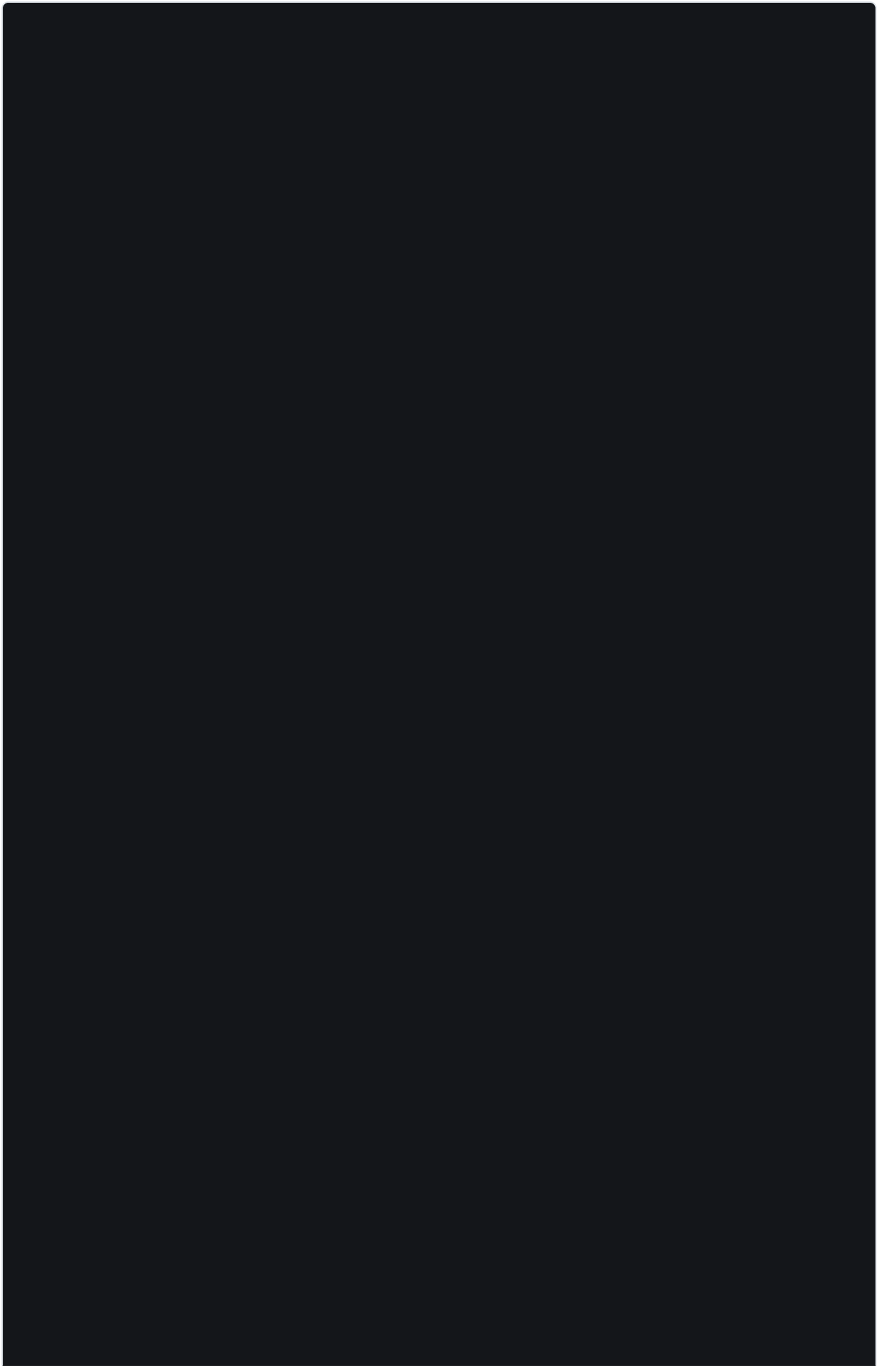


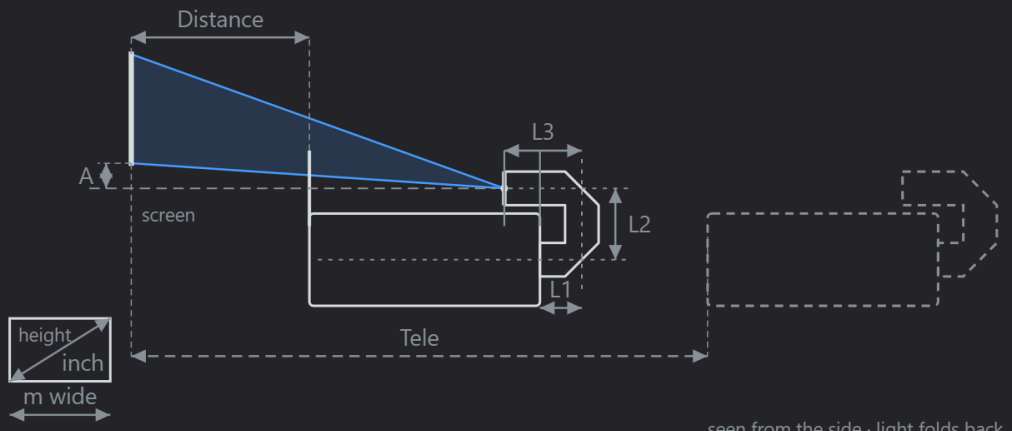
Figure 7.8b **L-shaped**, light leaving to the left. Drawn from *above*, because that is the plane its run turns in — pick up or down instead and the drawing becomes a side elevation whose screen is the ceiling or the floor. The two legs are typed underneath, "**L2**" with its own datum, and the distance runs to the near face, the lens tip or the far face — guides use all three. There are no vertical columns: this family's vertical is its arm.



Projection scheme · step 3 of 4

Working from: A distance table | I know the throw ratio

Lengths in: millimetres



seen from the side · light folds back

Size	The screen that row is for, read as the unit beside it says: a diagonal in inches, or the image width in metres. Never its height.
Distance	What the guide prints against that screen, zoom at its wide end: the back face out to the surface.
Tele	The same row with the zoom at its long end — the same picture, from further back. Leave it empty on a prime.
L1	The first leg of the casting: from the lens mount forward to the first elbow.
L2	The riser: up past the lid to the second elbow.
L3	The third leg: back over the cabinet to the glass, which sits at L1 − L3 from the front face. The beam leaves it there.
A min / max	Where the image can sit against that row's screen: the printed pair from the chosen line up to the image's bottom edge or centre, machine standing on its feet. A range here IS the vertical shift travel, stated as geometry — it writes the lens's travel, not its park — and pairs at two sizes pin the axis height with it.
Lateral	The same across the room, when the guide prints one: image centre from the body's centreline, + to the machine's own right. Solves the horizontal travel — a fold swaps the sign itself, since its beam-right is the body's left.

Two rows solve it; more tighten it.

Size		Format	Distance	Tele	A min	A max	
80	inch	16:10(WQXGA)	mm	—	—	—	✕
80	inch	16:10(WQXGA)	mm	—	—	—	✕

+ Row

"Distance" measured to: The back face

L1 — forward (mm): 121

L2 — rise (mm): 0

L3 — back (mm): 0

Image position (optional) — when the guide prints where the image can sit rather than a shift percent: one pair per screen size, in the A columns above, machine standing on its feet; one value twice for a fixed position. This solves the shift *travel* — and pairs at two sizes pin the axis height with it.

Measured from: The lens axis

To the image: bottom edge

Lateral min / max (mm):

Give at least two rows at different screen sizes — two points fix a line and nothing fewer does.

Cancel | Back | Next

Figure 7.8c **Rear fold**, folding up over the lid. The light walks the three legs and leaves over the cabinet's own back, which is why its distances are stated to the *back* face. All three legs are typed; the "**A min / max**" columns take the guide's image-position pair, which is how a sheet states shift when it never says "shift".

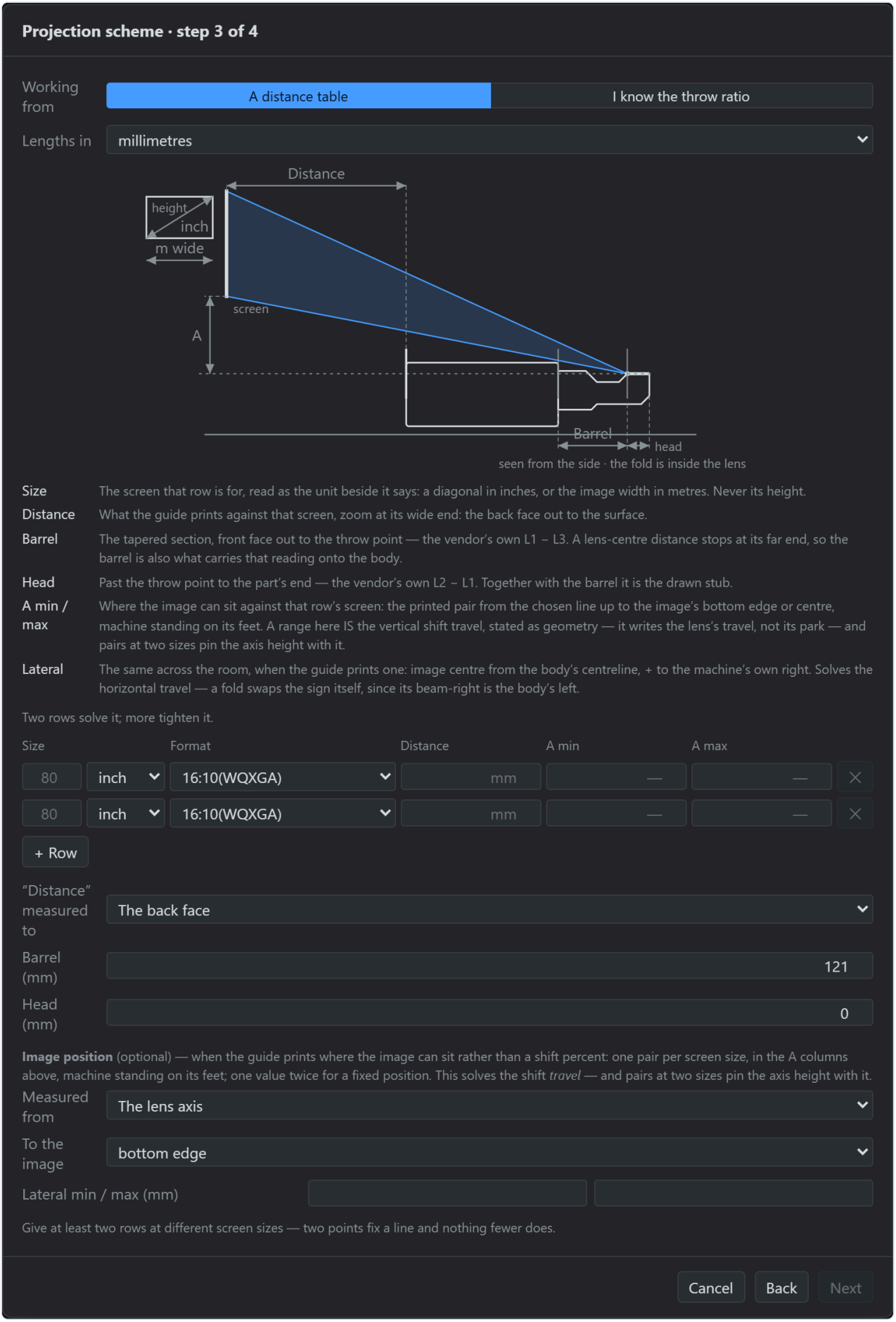


Figure 7.8d **Ultra-short-throw**, bolted onto this chassis. The fold is inside the lens, so the part is drawn as the vendor draws it — machine standing on its feet, screen to the left — and its two runs are typed by name:

"**Barrel**" to the throw point, "**Head**" past it to the end. No "**Tele**" column, because every lens of this kind is a prime, and no "**Drop**": that column belongs to the welded-in guides' inverted wall mounts.

Two of those differences are worth stating as rules, because they decide which figure on a datasheet you are looking at:

- **Typed sections are the authority.** The casting is what you typed, the beam is derived from it, and the rows contribute the throw ratio, the zoom walk and a cross-check. Where the rows put the throw point somewhere the legs do not, the disagreement is reported in millimetres rather than resolved silently in either direction. It also retires a question: the fold's legs and the bolt-on's barrel already say where the glass sits, so those two are not *also* asked for a lens-from-front figure that could disagree.
- **Drop and image position are two conventions for one thing.** A welded-in ultra-short-throw is printed for an inverted wall mount and states the vertical as a drop from its own datum line — hence the "**Drop**" column and the "**Plate → datum**" figure that places that line on the machine. Everything else is printed standing on its feet and states the vertical as where the image can sit. Feeding one to the other reads the machine upside down, which is why a family is only ever shown its own.

Step 4 reviews it. Two columns, "**Now**" and "**After**", listing exactly the fields Apply will write and nothing else: the exit point, the rotation, the throw ratio and its per-format column, the apex at the long end of the zoom, the shift park and travel, and — above them, if step 1 staged any — the body rows.

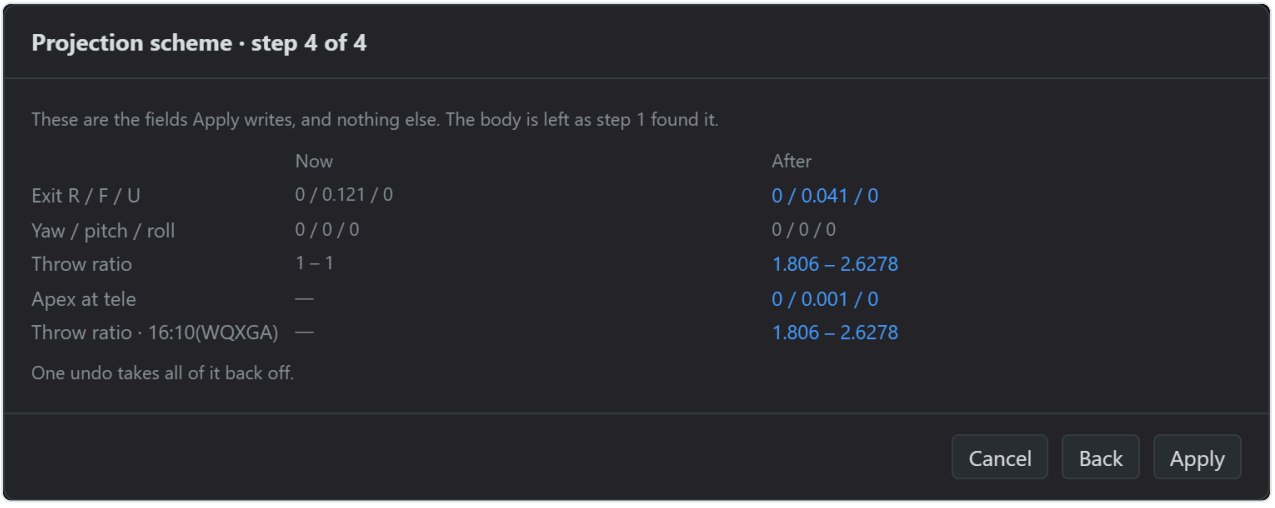


Figure 7.8e Step 4. A lens that knew nothing about its own optics — throw ratio 1, and a barrel with no beam behind it — against what two calculator rows solved. Everything the wizard writes is on this page, and one undo takes all of it back off.

WHAT THE WIZARD DOES NOT ASK FOR

Coefficients. Some vendors describe a lens as `distance = a × diagonal + b`, and it is tempting to offer those two boxes — but **no datasheet prints them**. They live inside a manufacturer's online calculator, and every machine that has them already ships in the bundled catalogue. That calculator answers "this size, this far" one size at a time, exactly as a printed guide does, so its answers go in as table rows — measured to the lens, so set "**Distance measured to**" to the lens tip — and recover the same optics. The worked example above is precisely that: two calculator answers, and the lens comes back.

CHAPTER 8

Cables and the diagram

Projectors and devices carry typed ports. Join two and a cable appears — in the 3D scene, in the schematic, in the length totals, and in the reports.

8.1 Ports and connections

Every port has a type (HDMI, SDI, Ethernet, HDBaseT, fibre, DMX, power, …) and a direction: **in**, **out**, or **bidi**. One predicate governs every connection, whether made in the panel or by dragging in the diagram — it rejects joining a port to itself, a type mismatch, a direction clash, and a port that is already occupied.

8.2 Cables in 3D

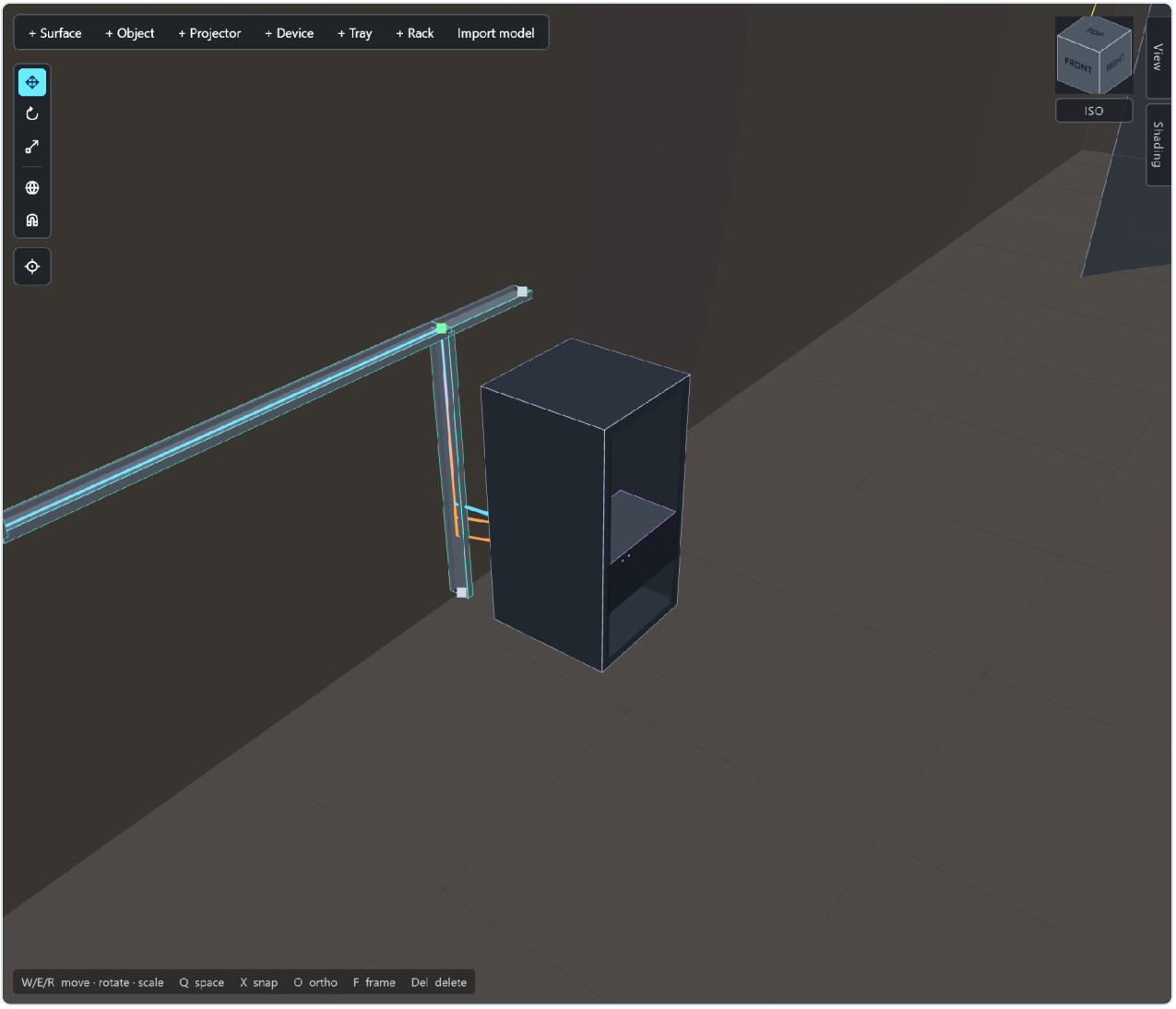


Figure 8.2a Cables are real geometry with a real route — here the four runs leaving the rack and climbing to the tray above it. Colour follows the port type — two HDMI and two Ethernet, and all four routed through the tray network. Beams are switched off in this view so the cables are not lost among them.

Click a cable to select it; click again to **insert a waypoint**, which you then drag with the normal gizmo. Waypoints are world-space on purpose: a cable route is a fact about the building. The ends follow their devices; the dressed middle stays where you put it.

Every cable reports a live length — the routed polyline plus a slack percentage. The project default for slack is set from the "⚙" in the Cables section header; an individual cable can override it.

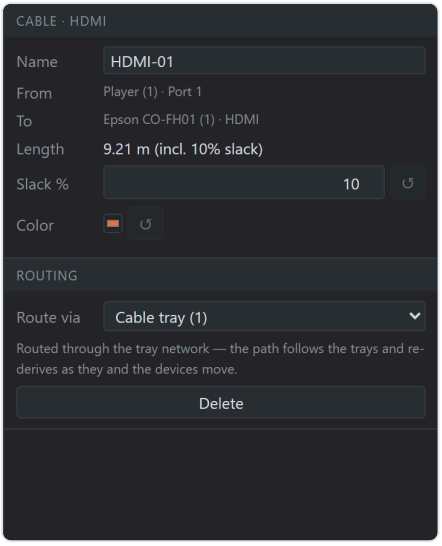


Figure 8.2b The cable inspector: its two ends, its type and colour, slack, and the "**Route via**" control that hands the route over to a tray network (Chapter 9).

> **SEVERAL CABLES AT ONCE**
>
> `Ctrl`-click adds to the selection in all three places a cable can be picked — the browser list, the run in 3D, and the edge in the diagram — and every selected cable draws selected, not just the last one. "**Slack**", "**Colour**" and "**Route via**" then write to the whole selection, dashed where they disagree (§2.4), as do "**Clear waypoints**" and Delete: one undo entry each.
>
> What stays singular is what can only be singular. The name and the waypoints belong to one run; "**From**" and "**To**" read **—**, and "**Length**" totals the selection, counting any unplaced ends beside the total rather than folding them into it. A modifier-click on a run in the viewport therefore adds to the selection instead of inserting a waypoint.

8.3 The 2D diagram

Diagram mode draws the same rig as a schematic.

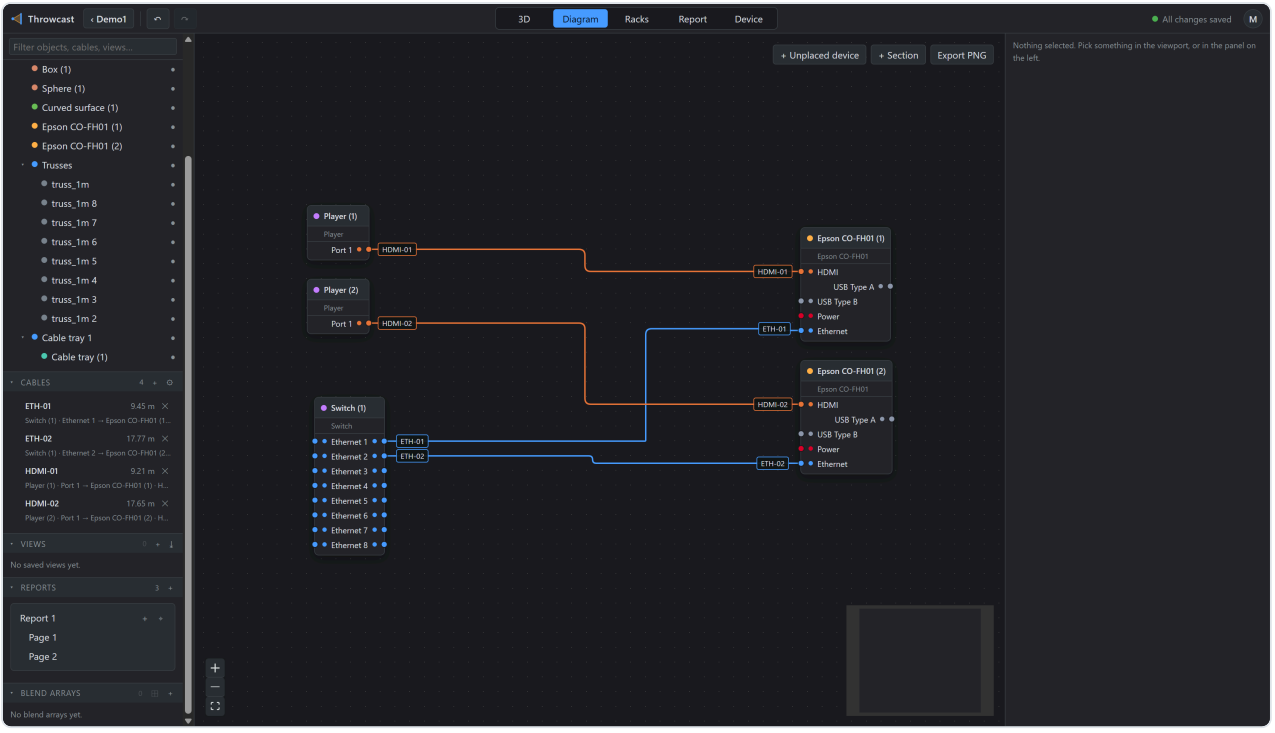


Figure 8.3 The demo rig as a schematic. Each device shows its ports; a cable's name rides both ends of its run. Five devices here — two players, a switch and the two machines — with four cables between them. A device with no place in the 3D scene would be marked "**UNPLACED**"; every device here has one.

- **Drag a port to a port** to create a cable.
- **Drag an edge's middle segment** to route it by hand; only the interior corners are stored. **Double-click a cable** to drop that hand-dressing and return to the automatic route.
- **Port labels track their side** — inputs on the left, outputs on the right. A bidirectional port shows both, and only one cable can occupy it at a time.
- **Export PNG** writes the diagram out as an image.

Unplaced devices

A device can be marked unplaced: it exists in the signal path but has no position in the venue — the front-of-house rack nobody modelled. It draws nothing in 3D, and its cables report no length (the browser shows `–`), but it is a full participant in the diagram.

Sections

"**+ Section**" draws a named frame around part of the diagram. Membership is **geometric, never a stored link**: whatever node's centre is inside the frame belongs to it, so devices join and leave simply by being dragged across the border. Dragging the frame moves every member with it, as one undo entry.

8.4 Naming

Cables are auto-named from the port type — `HDMI-01`, `ETH-01` — by scanning the names already in use rather than keeping a counter, which would drift on undo and delete. "**Renumber**"

in the cable settings only touches auto-named cables; a a cable you have named by hand is never rewritten.

CHAPTER 9

Cable trays

A tray is a network: points joined by straight stretches, each drawn as the open-top U-channel it is. One connected run is *one object* — bend it, extend it, branch it from its points — and a cable attached to it routes itself.

9.1 A tray is a network

A tray object stores points and the stretches between them; every stretch draws with the network's width and side-rail height. A corner is two stretches sharing a point, and a tee is a point where three meet — geometry inside one object, not separate objects lined up. "**+ Object ▸ Cable tray**" drops a single 3 m stretch overhead; everything after that is shaped from its points.

> **THE RULE**
>
> **One object is one connected network.** An edit that brings two trays into contact fuses them into one object; deleting the stretch that bridged a network splits it back into one object per piece, and any routed cable follows the piece nearest its ends. The object list always mirrors what is physically connected — there is no link to create and nothing to go stale after an undo. Either way it is a single undo step.

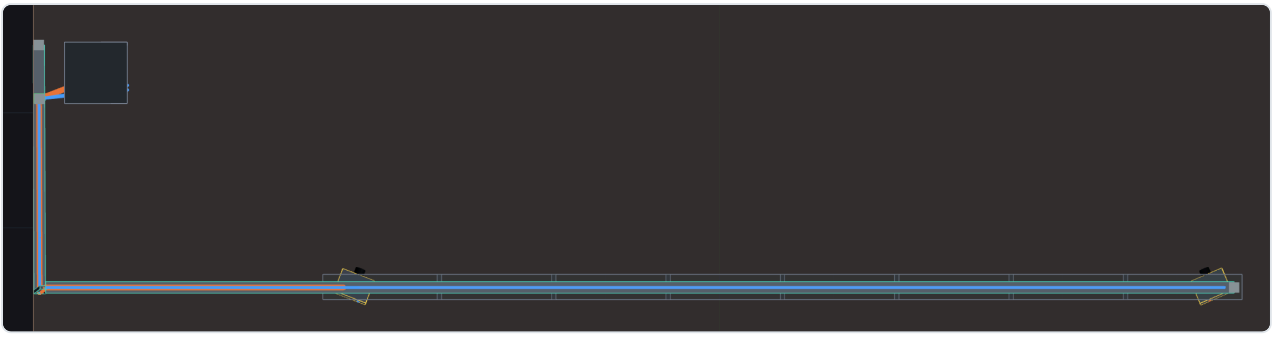


Figure 9.1 The demo project's run, seen from directly above in orthographic view (0) — the way a tray run reads best. **Grey** dots mark free ends; a plain bend carries no dot, and a junction — three or more ways meeting at one point — shows **green**. The orange and blue lines running inside the channel are cables routed through the network, dropping off where each reaches its projector.

The dots and the drag handles go away with the tray: hide a run — with its own eye, with an ancestor's, or with a report page's per-view hide (§11.4) — and its graphics go too, rather than hanging in front of the venue over a run that is no longer drawn.

9.2 Shaping a run

Select a tray and every point wears a small sphere. What the gizmo moves depends on what you have hold of, most specific first:

- **A point** — click its sphere. Dragging it re-aims every stretch touching it, and the point **snaps** onto other tray points and centrelines in reach, which is what makes a connection exact rather than eyeballed.
- **A stretch** — click the steel. The stretch moves rigidly — both its points together — and its neighbours re-aim to stay connected. `Del` removes it.
- **The whole network** — select the tray with nothing sub-selected: from the object list, or `Esc` to step back out.

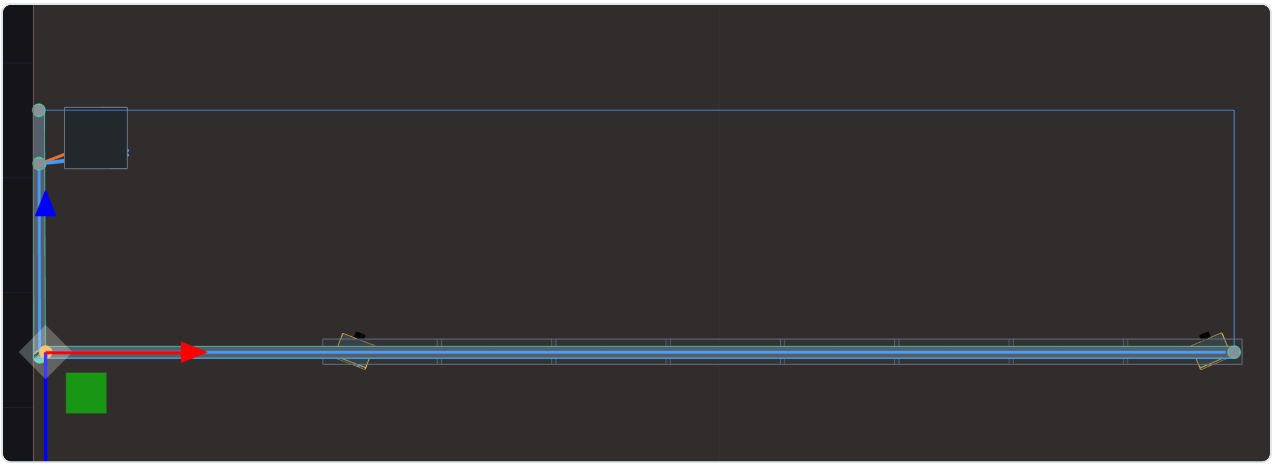


Figure 9.2 A selected point — the amber sphere — with the gizmo on it. Dragging bends the run at that point; the stretches on either side follow.

9.3 Building a run

Everything past that first 3 m stretch is built with two gestures, both on `Alt`: **Alt-click** puts a new point on the run, and **Alt-drag** grows a new stretch out of a point. There is no draw mode to enter and no tool to pick up — select the tray and hold `Alt`.

Alt-click — a new point. With the tray selected, hold `Alt` and move the pointer along the channel: the centreline lights up and a translucent ghost point slides along it under your cursor, showing exactly where the point will land. Click to place it. The gizmo steps aside while `Alt` is down so it cannot swallow the click, and the new point arrives selected with the gizmo already on it — so the very next drag bends the run there.

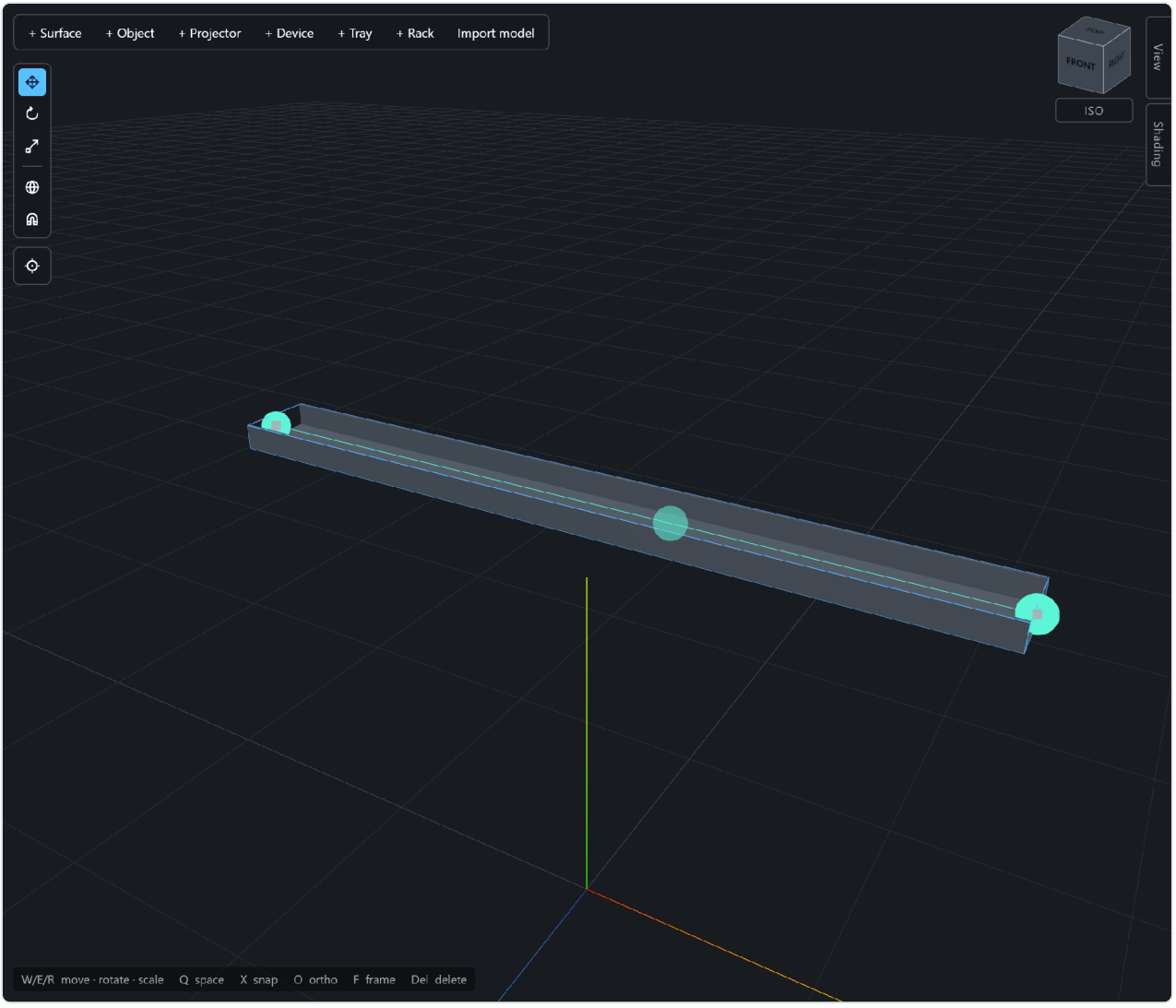


Figure 9.3a `Alt` held over a plain 3 m stretch — what "**+ Object ▸ Cable tray**" gives you. The lit centreline is the "you can place a point on this line" affordance, and the sphere sitting on it is the ghost, tracking the pointer: click and the point lands exactly there. The two ends keep their own handles.

Alt-drag — a new stretch. With a point selected, hold `Alt` and drag its gizmo. The point you started from *stays put*; a fresh point is minted under the drag and a new stretch grows between them, so you are pulling steel out of the network rather than moving it. Release to set the length. The gesture works from *any* point: from a free end it extends the run, and from a point in the middle it makes a **tee** — the mid point becomes a junction and shows green.

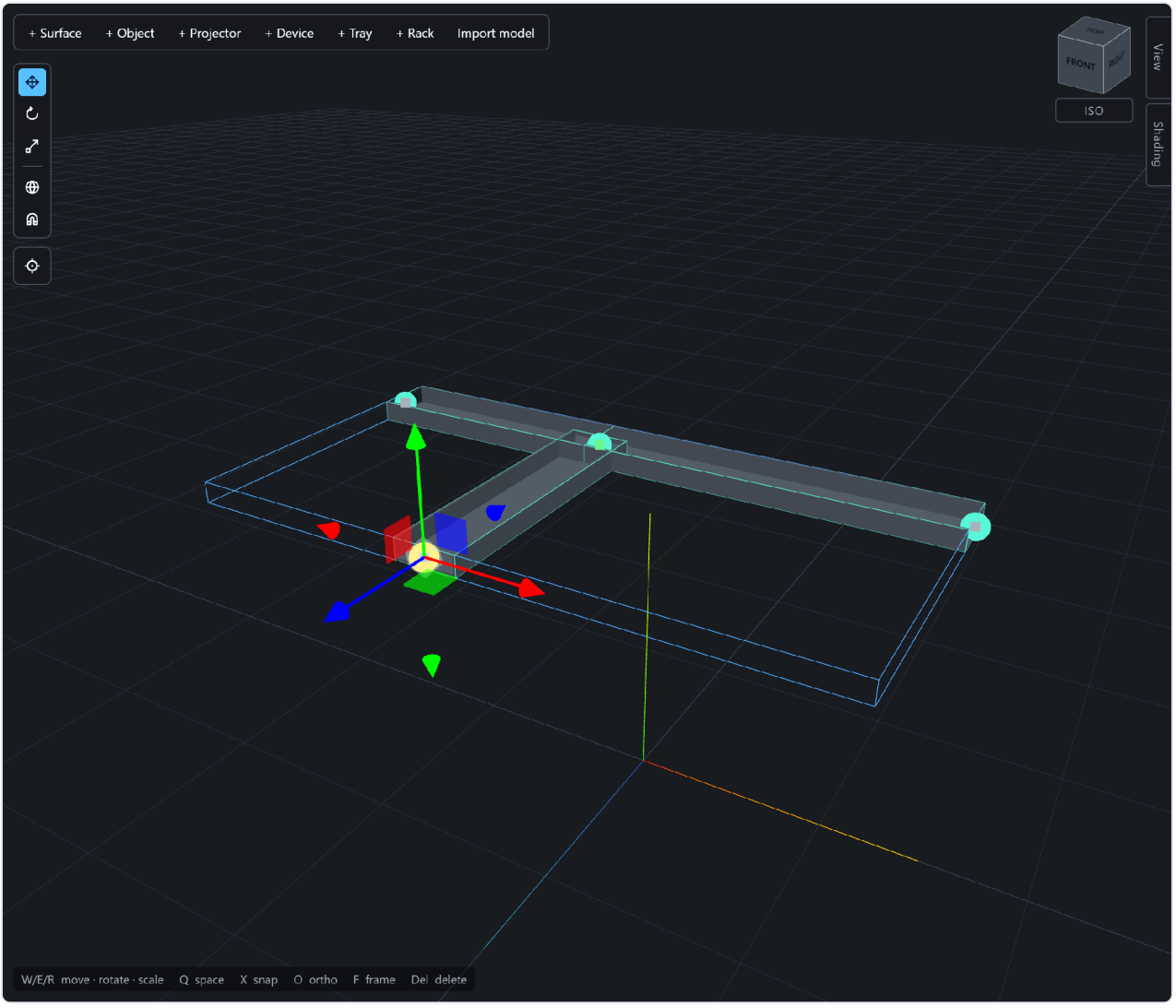


Figure 9.3b The run after an `Alt`-drag off a point in its middle: a spur at right angles, and the point it grew from is now a green junction. Pull an arrow and the branch comes out square to that axis; pull a plane handle and it stays in that plane.

Nothing about the gesture is horizontal. Drag the green axis instead and the same branch climbs — a riser to a tray at ceiling height, or a drop to a rack, is the same `Alt`-drag on a different axis.

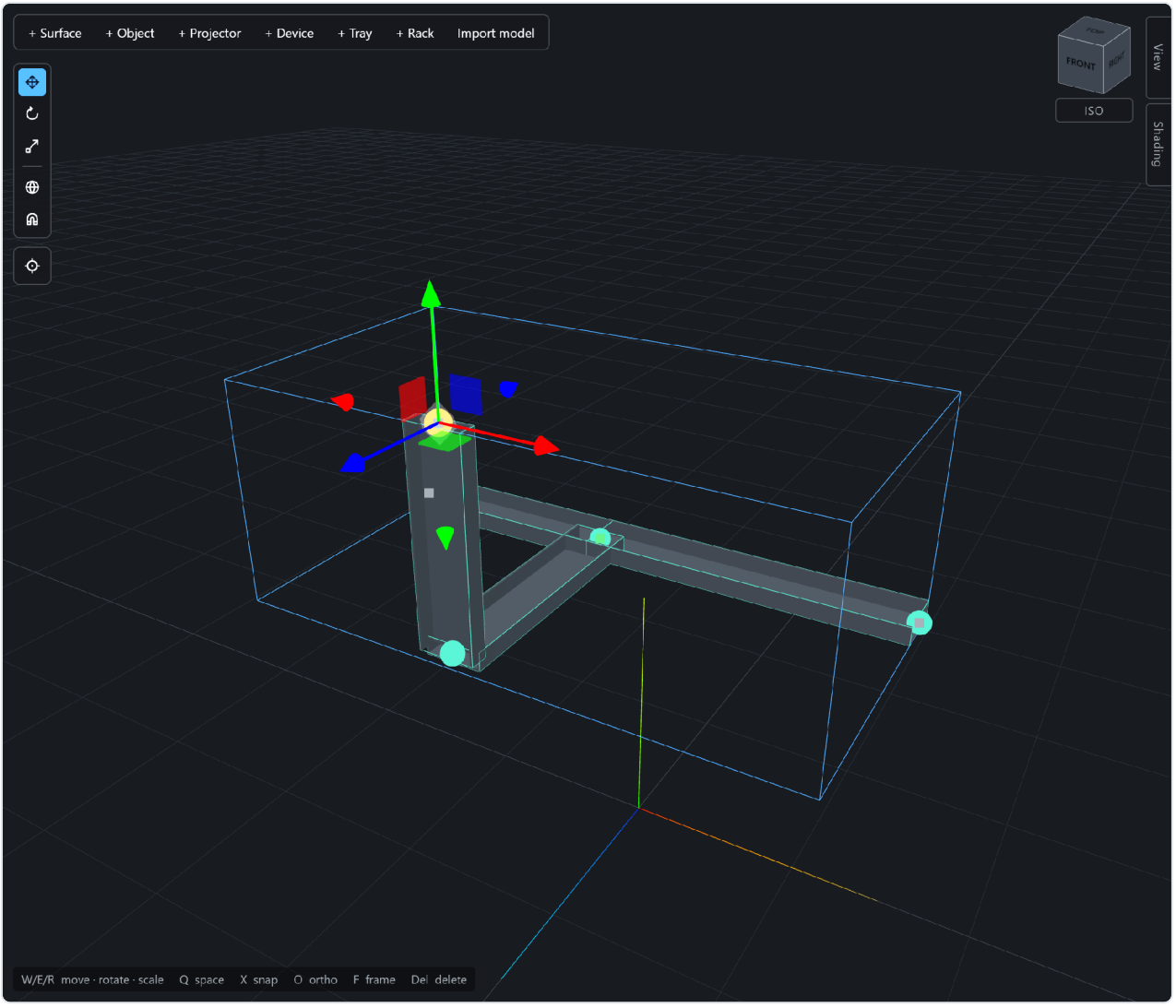


Figure 9.3c A second `Alt`-drag, this one up the green (Y) axis: the spur's end climbs into a riser. One object still — the blue box is the whole network's bounds, and a cable routed on it will climb the riser if that is the shorter way round.

> **WHICH GESTURE `ALT` MEANS**
>
> The two never compete: with a *point* held, `Alt` is the branch gesture and the centreline stays dark; with only the *tray* held, it is the insert gesture and the gizmo yields. So after placing a point — which leaves that point selected — press `Esc` once to drop back to the tray before `Alt`-clicking the next one.

Repeat the two gestures and a venue's run appears: `Alt`-click where a branch is wanted, `Alt`-drag it out, drag the end where it belongs. Two refinements worth knowing as you go:

- Points within **1 cm** fuse into one joint on release — and a point landing on the middle of a stretch makes a **tee**, splitting that stretch. This is how a branch you pull across to another run *joins* it, and why snapping (§9.2) matters: it makes the coincidence exact rather than nearly.
- Trays that merely **cross** — neither having a point on the other — stay unconnected, exactly as they would in the building.

9.4 Routing a cable through a network

Select a cable, and in the inspector set "**Routing → Route via**" to a tray. Its route then becomes derived: device → nearest entry point on the network → shortest path along it → nearest exit → device, re-derived live as trays and devices move.

Cables enter *anywhere* along a tray, not only at its ends, because that is what installers do.

> **WHY YOU PICK A TRAY, NOT "THE NEAREST ONE"**
>
> The tray you choose names a *network*, not a path. With several disjoint runs in one venue, "the closest tray anywhere" is ambiguous — and "the run I attached it to" is what attaching meant.

Two behaviours make attachment safe:

- **Manual waypoints are kept and ignored** while a cable is on a tray, and come straight back on detach. Attaching is not destructive.
- **Deleting a tray does not delete the cables through it.** The attachment is cleared and each cable falls back to its own waypoints. One undo restores the tray and every attachment to it.

WORKED EXAMPLE — THE DEMO PROJECT'S CABLE SCHEDULE

HDMI-01 · Player (1) → Epson CO-FH01 (1), via the tray	9.21 m
ETH-01 · Switch (1) → Epson CO-FH01 (1), via the tray	9.45 m
HDMI-02 · Player (2) → Epson CO-FH01 (2), via the tray	17.65 m
ETH-02 · Switch (1) → Epson CO-FH01 (2), via the tray	17.77 m

Every figure includes the project's 10 % slack. All four runs ride the same tray network, so each length is the routed path — device to tray, along the run, and down to the machine — not a straight line.

CHAPTER 10

Racks

A rack is a container: a 19-inch or 10-inch frame whose only job is to hold devices at discrete U positions and move them as one.

10.1 Adding a rack

"**+ Rack**" places one. Its inspector sets the capacity in rack units, the width standard and the front-to-back depth; the outer box is derived from those. The opening faces the rack's local +Z, and the origin is its bottom-front-left corner, so a rack at floor height stands on the floor.

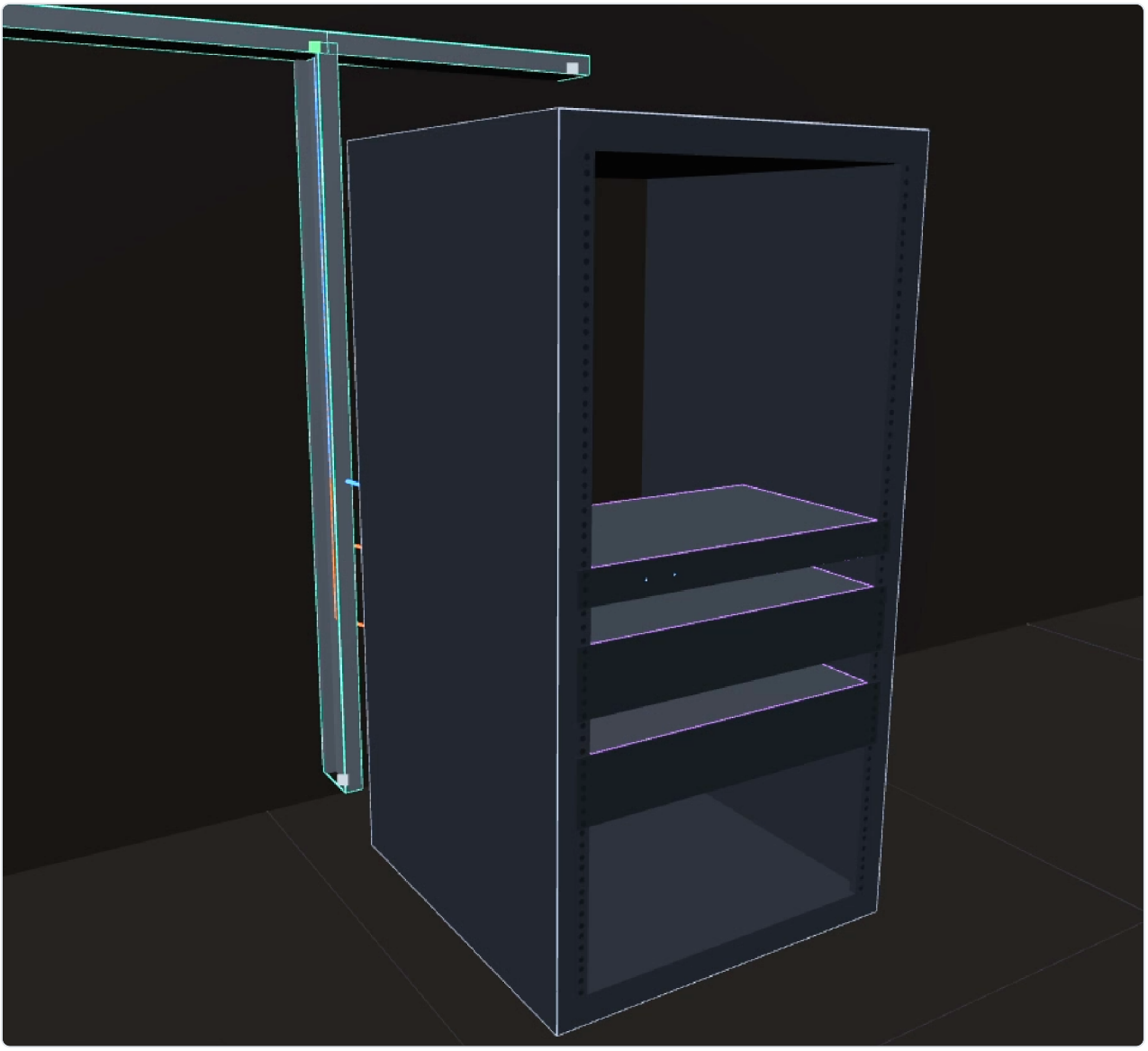


Figure 10.1 The rack in the venue: an open 22 U frame holding a switch and two players, with their cables leaving through the top. Occupancy is derived from the rack's children — nothing about it is stored on the rack itself.

10.2 Mounting devices

A device is mountable if its definition says it is rackmount, which is set in the Device tab along with its height in U and its depth. There are two ways to mount one:

- **In the object tree** — drag the device row onto the rack row. Racks take no insertion line, so a drop inside one becomes a drop *onto* it.
- **In Racks mode** — drag from the palette into a free slot.

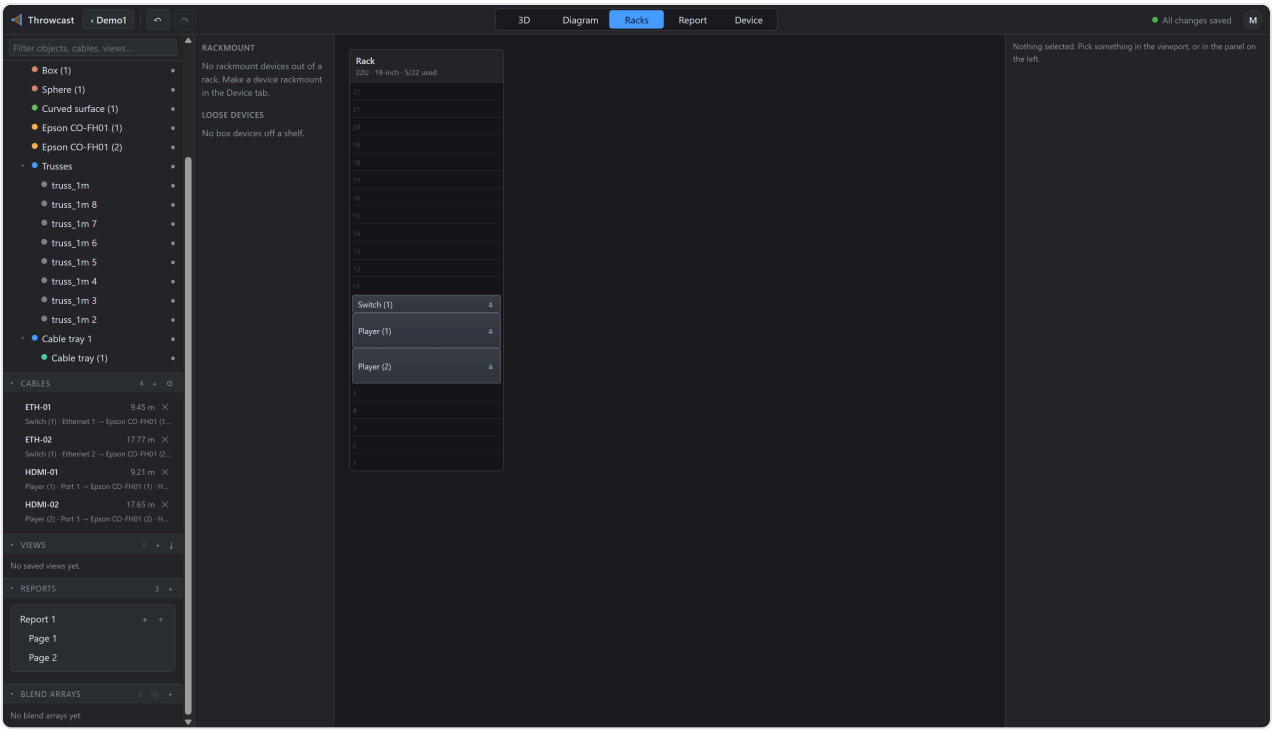


Figure 10.2 Racks mode. The rack reads as a front elevation with numbered U slots; the palette on the left offers rackmount devices for the rails and loose box devices for shelves. The header reports `22U · 19-inch · 5/22 used` — a 1 U switch and two 2 U players.

A mounted device becomes an ordinary child of the rack, carrying the slot it occupies. Its position is *derived* from that slot and rewritten whenever the mount changes — the slot is the editing handle, the position is what gets drawn. Moving the rack moves everything in it.

> **SHRINKING A RACK**
>
> A rack will not shrink below its highest occupied slot. The inspector says so — “can’t shrink below 6 U — a device is mounted up to there” — rather than silently dropping the device.

10.3 Shelves

A shelf is a rackmount device whose job is to carry others. Its block in Racks mode is a drop zone: box devices dropped on it become riders, rendered as chips, landing at the gap under the pointer. A laden shelf claims as many U as its cargo really needs, so nothing can be mounted into the air its contents occupy.

CHAPTER 11

Views and reports

A saved view is a camera you can come back to. A report is a set of composed pages that export to PDF — the deliverable a client actually receives.

11.1 Saved views

The "**+**" in the Views section bookmarks the current camera; double-clicking a saved view flies back to it. Views travel with the project. The "⤓" in the section header renders the current viewport to a 3840 px PNG.

11.2 What a report is

A report is a set of **pages**, and a page is a **sheet of paper you arrange content blocks on** — a 3D view, a paragraph, the cable schedule, a rack elevation. One of those blocks is a frozen view of the venue, which is what a whole page used to be; now it is one thing a page can hold, at whatever size you drag it to, beside the tables that explain it.

Table 11.1 The seven kinds of block.

BLOCK	WHAT IT PUTS ON THE PAPER
3D view	A frozen view of the venue — camera, display settings, hidden set and annotations (§11.4)
Text	Headings, paragraphs and lists (§11.6)
Cable schedule	Every run, its ends, its length to order (§11.7)
Equipment list	What is in the rig, how many, and where (§11.7)
Rack	A front elevation of one rack (§11.8)
Diagram	The 2D signal diagram, or one named section of it (§11.8)
Blend array	One array's projectors and the seams between them (§11.8)

The "**+**" on a report row adds an empty sheet. Pages made before sheets existed open as sheets carrying one full-bleed 3D view block: the same camera, the same crop, the same PDF. (A page that had lost its camera opens *empty* instead — there is no honest default for where a drawing was framed from, and guessing one prints a picture of somewhere nobody chose.)

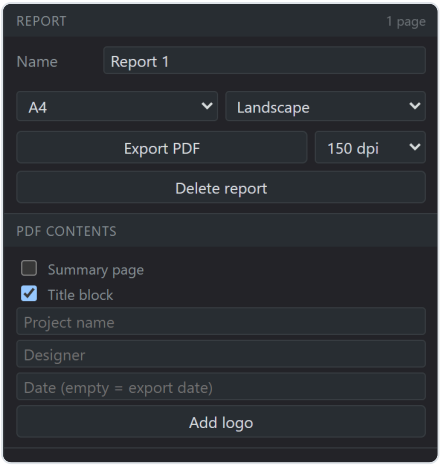


Figure 11.2 The report inspector: paper size and orientation, export resolution, and what goes into the PDF — an optional summary page with the equipment list, and a title block with project name, designer and date.

11.3 Arranging the sheet

Double-click a page in the Reports section (or press "**Report**" in the mode switcher) and the sheet covers the viewport: the paper at its real proportions, the reserved title strip along the foot, and a toolbar across the top.

Table 11.2 The sheet toolbar.

BUTTON	DOES
"**+ 3D view**"	Puts the view you are looking at onto the page, and opens it for framing
"**+ Text**"	A text block, opened for typing
"**+ Cables**" / "**+ Equipment**"	The two schedules
"**+ Diagram**"	The signal diagram, or a section of it
"**+ Blend array ▾**" / "**+ Rack ▾**"	Menus listing what the project has, rather than guessing one
"**Refresh**"	Take the cached pictures again (§11.4)
"**−**" "**+**" "**Fit**"	Zoom out, in, and fit the whole page in the window — the percentage between them is the current zoom. `Ctrl` +scroll does the same
"**Done**"	Leave report mode

A block is dragged by its middle and resized by its edge and corner grips. It snaps to the edges and centre lines of the other blocks and of the page itself, and draws the guide line that says why it stopped there — no grid, because a page is a composition rather than a spreadsheet, and what you actually want is this block's left edge flush with that one's.

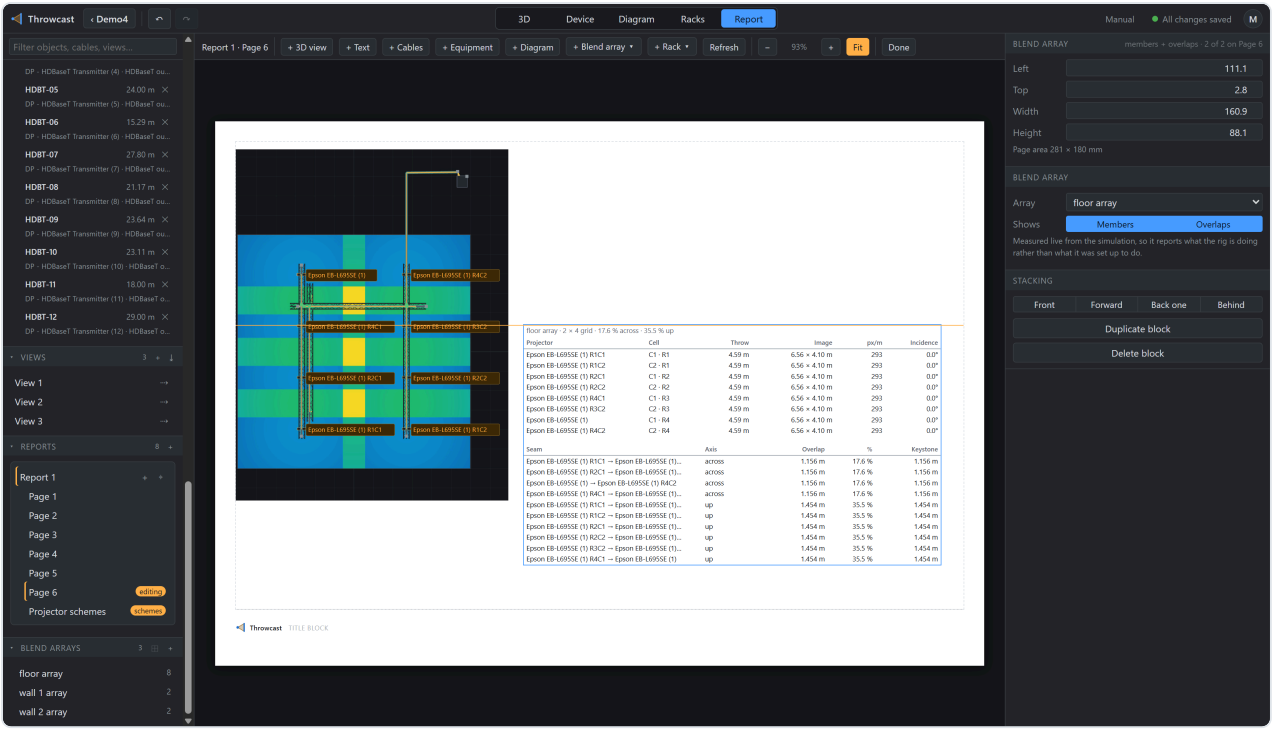


Figure 11.3 A sheet mid-drag. The blend-array table is being pulled down the page and has stopped on the orange line — the 3D view's own centre line, which is what the guide is announcing. Nothing is written yet: the inspector on the right still reads the block's stored "**Top**", and `Esc` here would leave the page exactly as it was.

Table 11.3 On the sheet.

INPUT	DOES
Click / `Ctrl`-click	Select a block / add to the selection
Drag on empty paper	Marquee-select
Double-click	Open the block: type into a text block, frame a 3D view
Arrow keys	Nudge — `Shift` for ten times the step
`Ctrl` + `D`	Duplicate the selection, offset and on top
`Del` / `Backspace`	Delete the selection
`Esc`	Close a text editor, then clear the selection

Nothing is written while a gesture is in flight: a drag or a resize is **one** undo entry and one save, and `Esc` mid-drag abandons it having written nothing. Typing into a text block is the same bargain one level up — the block is committed when the editor closes, so a paragraph is one `Ctrl` + `Z` rather than a hundred.

The inspector holds the numeric side of the same thing: "**Left**", "**Top**", "**Width**" and "**Height**" in millimetres from the printable area, a "**Stacking**" row (blocks are drawn in their own order, so this is what puts one in front of another), "**Duplicate**" and "**Delete**", and whatever that kind of block is about.

SEVERAL BLOCKS AT ONCE

Left, Top, Width and Height are the vocabulary every block has and cross kinds, so “these five, 60 mm wide” is one edit — and only the edge you typed travels, each block keeping its own other three. A schedule’s “**Group by**”, a rack block’s rack, a diagram block’s crop and a blend array’s “**Shows**” stop at the blocks of that kind. Duplicate, Delete and Stacking take the whole selection, the stack moving as one group so a selection does not shuffle against itself on the way up. The exception is a schedule’s “**Page**” (§11.7), which stays with its own block.

A PAGE IS A VIEW, NOT A SCENE

While a report page is open the Add cluster and the gizmo rail are gone, the gizmo detaches, and **the whole properties panel is read-only for anything that is not the report** — an object, a cable, a viewpoint, a blend array. A page is a saved *view* over the one live scene, so an edit made while laying out a page silently changes every other page in the report, and a lens is swapped from an inspector rather than from a gizmo. The report, the page and the blocks stay live: they are the document you came here to edit. Selection, shading, the view cluster and the annotation tools stay live too — that is what report mode is for.

11.4 The 3D view block

Double-click a view block — or press “**Edit view**” in its inspector — and the sheet lifts: the live viewport comes back, bound to that block’s camera, display state and hidden set, letterboxed to the *block’s* rectangle rather than to the whole page. That is the framing session, and it is where the annotation tools live.

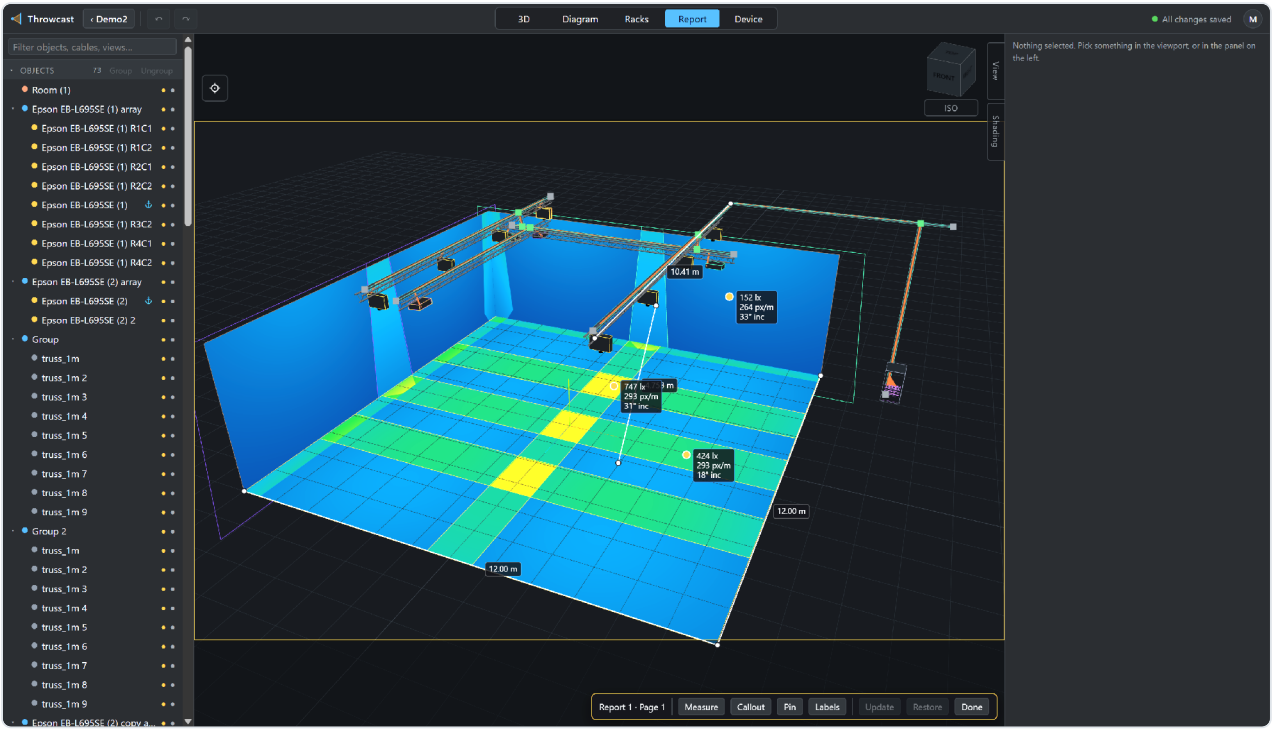


Figure 11.4 Framing a view. The orange frame is the composition boundary — what falls inside it is what prints into the block. The page bar sits bottom-right.

> **SKETCHUP SEMANTICS**
> Moving the camera or toggling display marks the block **modified**. Nothing is written until you press "**Update**". "**Restore**" discards your changes and re-applies the stored view. "**Done**" goes back to the sheet.

What a view freezes: the camera and its projection, the shading mode, the render mode (§11.5), beams, footprints, **cable runs**, **cutter ghosts**, the grid, the per-view "**Labels**" toggle on the page bar, and the hidden-object set you set with the dots in the object browser. So one page can carry the client's view of the room and the one you kept, showing the tools that shaped it.

Sheets show cached pictures

Display state and the hidden set are global to the one live scene, so two blocks looking at it differently cannot both be live. A sheet therefore shows a **cached render** of each view, taken at named moments: opening the page, leaving a framing session, resizing the block, an undo or redo that invalidated it, and "**Refresh**". Doing it on every document change would make a venue-sized scene impossible to arrange, which is what the sheet is for.

A picture older than the scene says **stale** rather than quietly lying. Arranging the sheet does not stale anything — a picture is a picture of the *scene*, and nudging a text block cannot have changed one.

> **AN EXPORT IS NEVER STALE**
> The PDF ignores the cache entirely and re-captures every block at print resolution, however long the sheet has been open. The badge is about what you are looking at, not about what you will get.

`Ctrl` + `D` on a view copies its camera, hidden set and annotations — which is the whole reason to copy one rather than insert a fresh one and frame it again — and takes the new block's picture straight away.

11.5 Drawing mode

"**View ▸ Drawing**" renders the venue as a **technical drawing**: black edges on white paper, no shading and no colour. A shaded render shows a room; a drawing shows dimensions and relationships, prints legibly in black and white, photocopies, and takes a pencil.

It is frozen into a view block like any other display setting, so one page can carry a shaded render and a line drawing of the same rig side by side, each block printing in its own mode.

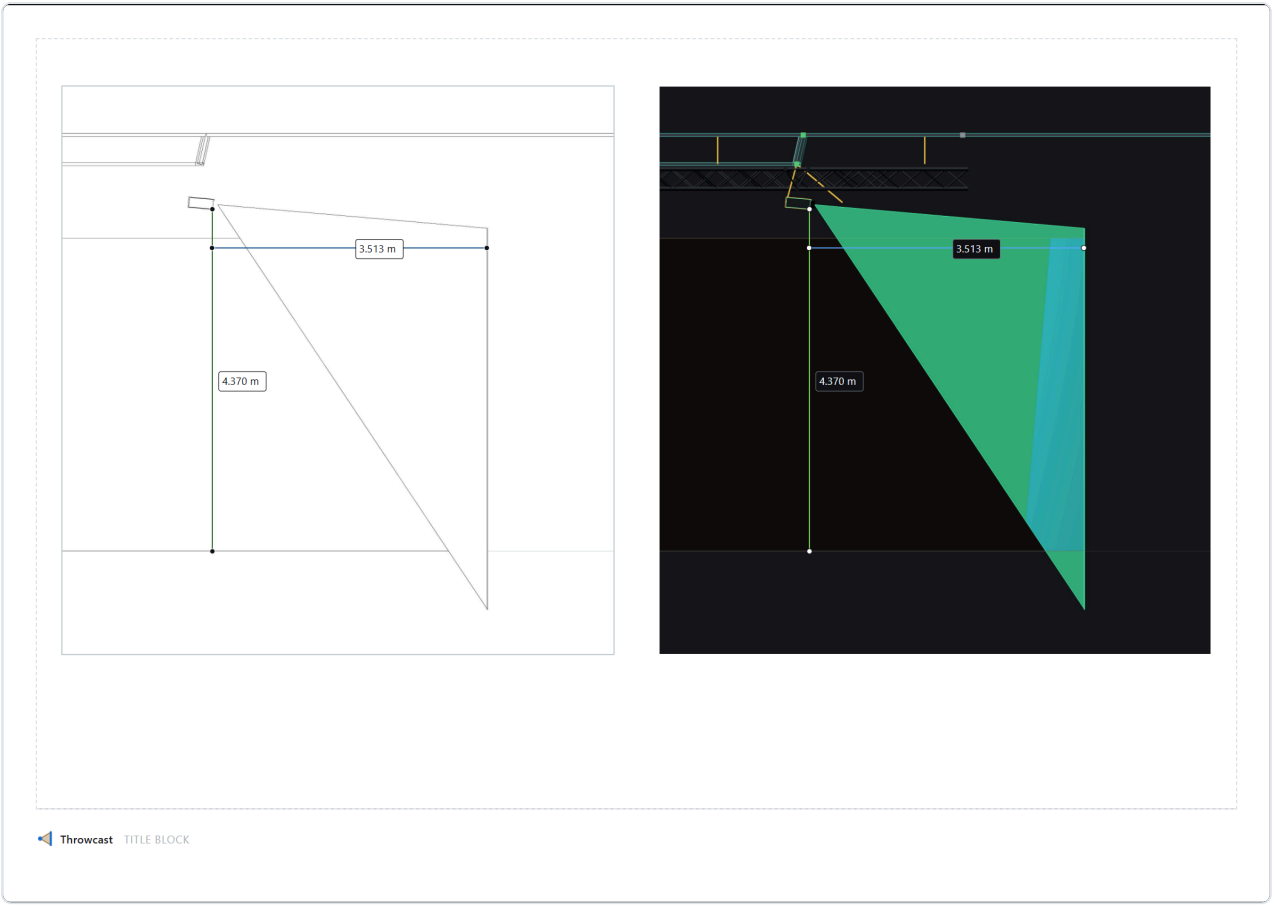


Figure 11.5 The same camera twice, one display setting apart. The second block is a `Ctrl` + `D` copy of the first with "**Drawing**" switched off, which is why the measurements are identical — they are the block's own annotations, copied with it. Note the ink: black on the paper, white on the render, and the Y-locked dimension green in both (§11.9).

> **IT IS A REPORT CONTROL**
>
> The button appears **only while a view block is being framed**, which is the one moment the setting has something to attach to. A drawing is a way of presenting rather than a way of working; in the 3D tab it would be a mode you could leave switched on with nothing on screen to say why the room had gone white. Leaving the framing session puts the viewport back to shaded, and — alone among the display toggles — it is not remembered between sessions.

Hidden lines are removed properly: the geometry is drawn white on white to write depth, and the edges are drawn over it depth-tested against that, so a beam stops where the surface hides it. The floor grid and the world axes go first and pale, as construction lines — at full ink the grid is by far the heaviest thing on the sheet and buries the rig it sits under.

Two things to know. Imported CAD arrives as shaded triangles with no edges, so Throwcast generates them; everything the app builds itself already outlines at the crease angle it was drawn with. And the in-scene name chips do not appear in a drawing — they are bitmaps in the app's dark palette. A view's own "**Labels**" are unaffected: those are vectors drawn over the picture by the annotation layer.

11.6 Text blocks

"**+ Text**" drops a block and opens it. Double-click one to type into it again; `Esc` closes the editor and commits the paragraph as one undo entry. The toolbar offers three heading levels, **bold**, and bulleted and numbered lists.

> **NO ITALIC, DELIBERATELY**
> The PDF embeds only the Regular and Bold faces of its font and cannot synthesise an oblique, so italic showed slanted on the sheet and came out upright on paper. A sheet that lies about what it will print is worse than one that offers less — bold does the job on a drawing sheet, and bold prints. `Ctrl` + `I` went with it.

11.7 Schedules: cables and equipment

Two tables, both derived from the project every time they are drawn — no cache, no stale badge, always current.

The **cable schedule** ("**+ Cables**") is what to order and where each run goes: Name, Type, From, To, Length, Slack. It sorts by type and then by the *number* in the name, so HDMI 10 lands after HDMI 9 rather than beside HDMI 1. A run with an unplaced end reads **unplaced**, never `0.00 m`: the FOH rack nobody has modelled has no length to measure, and a zero is what has somebody order no cable.

The **equipment list** ("**+ Equipment**") is what is in the rig: Qty, Kind, Model, Details, Location. It groups identical machines, and **location is part of the key** — two of the same switcher in one rack are a row of qty 2, the same switcher in two racks is two rows. The list is for building a rack as well as for buying the kit, and collapsing those would answer "how many" while losing "where". Model is the *model's* name, never the instance's: nobody orders a "Projector 07".

Either can be **grouped** — by cable type, by device kind — into bands carrying a count and a total, and **filtered** to one rack or one diagram section, section membership being geometric and derived exactly as it is in the diagram (§8.3). A filtered block captions itself with what narrowed it.

A schedule is **auto-height**: it ignores the height you drag and grows to its rows, stopping at the bottom of the page. It offers only the width grips, and its inspector "**Height**" reads *auto*.

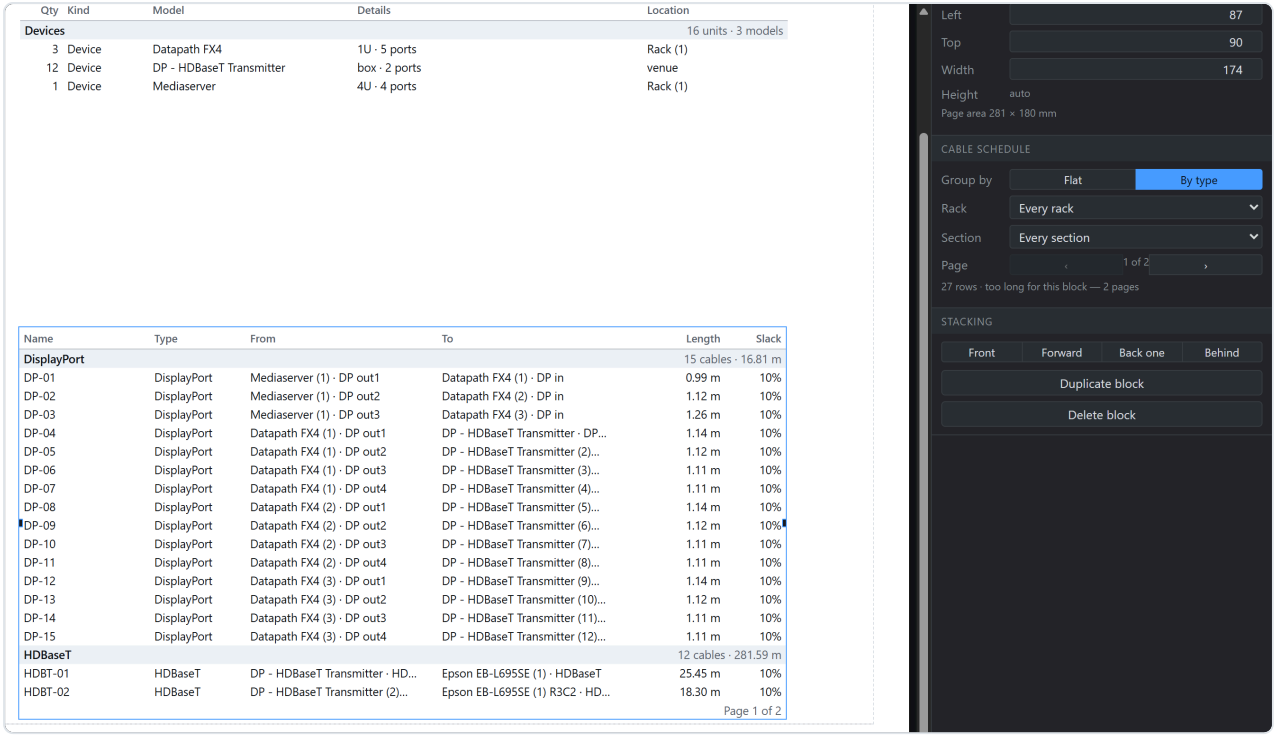


Figure 11.7 Both schedules on one sheet, zoomed in: the equipment list banded by kind above, the cable schedule banded by type below, each band carrying its count and total. This schedule sits low enough that its 27 runs no longer fit under it, so it spends its last line on *Page 1 of 2* and the inspector offers the "**Page**" control — *27 rows · too long for this block — 2 pages*. Put a second block on the next sheet, step it to page 2, and the list continues.

> **A LONG TABLE PAGINATES**
>
> A layout page is one PDF page, so a table cannot flow onto a second sheet by itself. Instead the rows are cut into pages of the same size and the inspector's "**Page**" control picks which one this block prints: put another schedule block on the next sheet, step it to page 2, and the list continues. Each page spends its last line on a dim *Page 2 of 3* — which is what makes the pages equal and what tells you there is more to place — and a page that opens inside a grouped band repeats it as *HDMI (cont.)*. The amber warning survives for the case that really is a fault: a block so near the foot of the page that nothing fits under its header at all.

What you see is what prints, in the strong sense: the sheet and the PDF are drawn from one layout pass in millimetres, so *1 of 4* on the panel and *Page 1 of 4* on the paper cannot disagree.

11.8 Rack, diagram and blend-array blocks

A **rack block** ("**+ Rack ▾**") prints a front elevation: the U column with U1 at the bottom, a block per mounted device labelled with its slot span, what is standing on a shelf, and the rack's name and *7/12 used* over it. It is drawn as vectors on screen and on paper — a rack is straight lines and short labels, and an image of one is fuzzy at exactly the size a rack is read at — at the proportions a rack really has, so a 12U cabinet comes out about square and a 42U one nearly four times taller than wide. Resizing the frame never distorts it; the elevation is fitted inside whatever aspect you drag.

A **diagram block** (“**+ Diagram**”) prints the signal diagram — the whole drawing, or one named section. Because section membership is geometric, a section block re-crops itself as devices are dragged into and out of the frame: the crop *is* the query. This one is a photograph rather than vectors, because the picture is the diagram’s own layout and a second implementation of it would be a second diagram free to disagree with the one you arranged. It is photographed from its own offscreen copy in a paper palette, so a report exports with the Diagram tab shut and never carries your pan, zoom or selection onto the page. Like a 3D view it shows a cached picture with a stale badge, and the PDF re-renders regardless.

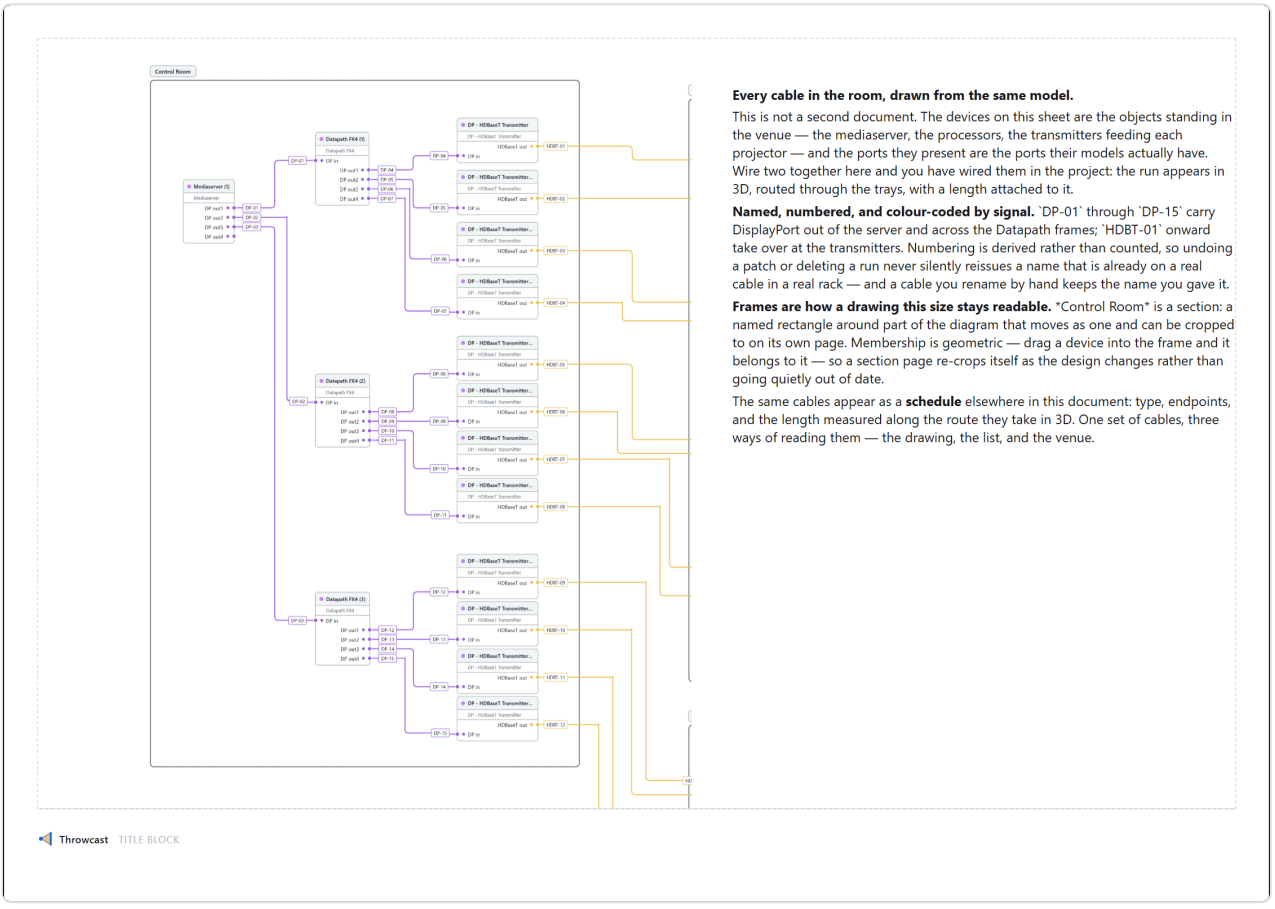


Figure 11.8 A diagram block beside a text block. This is the same drawing as the Diagram tab — the same nodes, the same routed edges, the same “**Control Room**” section frame — repainted in the paper palette rather than the app’s, which is why it reads as ink on white and not as a screenshot of a dark application.

A **blend-array block** (“**+ Blend array ▾**”) reports one array as two stacked tables. **Members** lists its projectors in lattice order — so a wall array reads left to right on the page because that is how it stands in the room — each with its cell (`C2 · R1` in a grid, `#3` round a ring), throw, image size, px/m and incidence; a machine hitting nothing says *no target* rather than reporting a throw of 0.00 m. **Overlaps** gives the seam between each adjacent pair: the metres and per cent of the planar approximation, beside the true keystone figure (§5.5). Either half can be switched off in the inspector, and the survivor takes the caption.

THESE NUMBERS ARE THE SIMULATION'S

A throw distance and an overlap are measured against whatever surface the beams actually land on, so a blend-array block reports what the rig *is* doing rather than what it was set up to do — and the difference between those two is the reason to put one on a page. Like a schedule it is live, with nothing to go stale.

11.9 Annotations

Annotations belong to the **3D view block** that carries them, so the tools are on the page bar and appear while one is being framed (§11.4). They are content rather than view state: each one commits as you place it, and undoes normally, without waiting for "**Update**". A sheet that is merely being arranged draws none of them — there is no frame to lay anything out in — and the venue's own rulers stand down for the duration, so a printed drawing carries what was laid out on it and nothing left lying around the scene.

Table 11.4 The four annotation tools.

TOOL	WHAT IT DOES
Measure	Two points, vertex-snapped. `X` / `Y` / `Z` lock to an axis; hold `Alt` to suppress snapping.
Angle	Three clicks: an arm, the corner, then the other arm. `Tab` swaps the inner angle for the outer one.
Callout	Click a point and type. `Enter` commits, `Shift` + `Enter` makes a new line.
Pin	A live lux / px-per-m readout anchored to a surface.
Labels	Projector, device and rack name callouts.

All of them anchor to any visible body — surfaces, models, projector chassis, devices and cable trays. You can dimension a projector to the tray above it, or a rack to the truss over it.

RE-AIMING A MEASUREMENT

Click one and its two ends become **grips**; drag one and that end is taken again from wherever you drop it — the same snapping to vertices and to a projector's lens, and the same `X` / `Y` / `Z` locks, now running through the *other* end. `Esc` mid-drag puts it back; a finished drag is one `Ctrl` + `Z`. It is not a translation — the end lands on what is under the cursor, which is what a measurement is: two places in the venue.

The grips are there only while it is selected. Left alone, a measurement takes no clicks on its ends at all, so a ruler lying across a venue never comes between you and the corner it is measuring to.

A dimension measured on a *sketch* (§3.11) never grows these grips: its ends belong to the drawing, and you drag them inside the sketch editor, where they snap to its corners and its grid. Open the drawing to move one.

Angles

A takes the angle tool — the ruler's sibling, and the only annotation that wants three clicks: **an arm, the corner, then the other arm**, so the middle click is the one that lands on the corner. It snaps exactly as the ruler does, reads out live from the second click, and stays armed so the next one is three more clicks.

> **THE RULE**
>
> Tab **swaps the inner angle for the outer one** — while you are placing it, and afterwards on one you have selected. Three points describe two angles that add to 360°, and which of them you meant is a fact about the question rather than about the points: a splayed wall read from inside the room and from the corridor outside it is the same three clicks and two different answers. So the answer is stored with the measurement, and an angle you never pressed Tab on reads the inner one.

The **arc** is the part that says which angle is meant, and it is also the part you click to select one — the arms are left alone deliberately, since they run along the very geometry you are measuring and a fat target there would sit between you and it. Selected, an angle offers three grips: both arms and the corner, dragged like a ruler's ends.

Angles work wherever the ruler does — in the venue, on a report page, and inside the sketch editor, where A arms it against the drawing's own corners and grid.

Measuring a projector's throw

Two refinements make this work on a machine that is actually aimed somewhere:

- Hovering a projector adds its **lens** — the point the optics throw from, and the one the Metrics *Throw* reads — as a snap candidate. Without it, "lens to wall" came out as whichever chassis corner you happened to grab.
- Pressing an axis key a **second time** swaps that axis from world to the anchor object's own frame. So Z Z runs the far end down the projector's optical axis. A third press clears the lock.

> **WHY LOCAL AXES MATTER HERE**
>
> Once a projector is aimed off-square, no world axis is parallel to its throw — so a world lock measures a *projection* of the throw rather than the throw. A unit rolled 30° off vertical at 4 m reports 4.000 m under a world lock, and 4.619 m under its own.

While an axis is locked, the point where that axis pierces the scene is itself a snap candidate, so the far end can land on the wall instead of sliding through it.

Pins recompute

A pin stores only geometry — a point, the normal of the face it sits on, and which body it belongs to. Every number it shows is recomputed, so a pin stays true as the rig changes around it. On a sheet it shows the readings taken with the block's picture, through that view's own hidden set; they are refreshed with the picture and dropped with it.

Ink follows the picture underneath

An annotation over a view block always lands on the render, which fills the block edge to edge, so there are two palettes and the block's render mode picks one. Over a **shaded** view: white ink, dark chips, and the viewport's own axis colours, so a ruler locked to Y is green in the venue, green on the sheet and green on paper. Over a **drawing** (§11.5): black ink, white chips ruled in black, and darker axis colours that survive white paper.

That is the whole answer to "why does the report look different from the viewport": it should not, and the layout, the colours and the arcs are computed once and drawn by the viewport, the sheet and the PDF alike.

11.10 Projector-scheme pages

The "⌖" on a report row adds a generated page kind: **one PDF page per projector**, carrying that machine's 2D throw diagram at its *measured* throw, plus a data block — device and lens, throw ratio, shift, throw distance, image size, px/m, average lux, incidence and keystone.

By default the scope is every visible projector at export time, so a rig grown later stays covered; you can also pick an explicit checklist. These pages have no camera and take no capture: the preview is a stack of paper-aspect sheets and the PDF is pure vectors, both drawn from the same source — so the sheet you review is the sheet that prints.

The side and top views sit **side by side**; "**Views**" in the page inspector stacks them instead, one under the other on a shared vertical line, the way a vendor sheet draws them. On a landscape page the stack is limited by the height of the paper and leaves most of its width empty, which is why two columns are the default — the same drawing, at roughly twice the size. Side by side the two views are aligned on the **optical axis**: the dashed centre line runs across both at the same height, so what you read off one view carries straight into the other.

11.11 Exporting

"**Export PDF**" writes the report: paper size and orientation, 150 or 300 dpi, an optional summary page, and a title block on every sheet.

Every block is re-drawn for the export at print quality — 3D views and diagram blocks are re-captured, and text, schedules, rack elevations and blend-array tables come out as **vector text you can select and search**, at the size and position they occupy on the sheet.

> **WHY PRINT MATCHES SCREEN**
>
> The on-screen composition frame and the PDF crop are computed by the same rule from the same view-projection matrix, and a block's frame is a fraction of the page rather than a set of millimetres that could be rounded differently twice. What the orange frame contains is what the block contains; what the sheet shows is what the paper gets.

The title strip is signed **Throwcast** — the mark, the name, a hairline, then your project — drawn in vectors so it stays sharp beside vector type. Putting a studio's own logo there is a paid-tier

feature and the tier does not exist yet, so the control that used to set one is gone; a document that already carries a logo keeps it.

> **THE OLD PER-PAGE SWITCHES**
>
> A page inspector still offers "**Notes**", which prints on a continuation page *after* the sheet — and older projects may carry the per-page metrics and overlap tables, which still print the same way though nothing now switches them on. All three date from before a page could hold anything but one view. Blocks cover them better: a text block, a schedule or a blend array sits where you put it, at the size you chose, rather than on a page the layout has no say over. They stay only because turning them off would silently shorten every report that uses them.

CHAPTER 12

Keyboard reference

The shortcuts

Table 12.1 Shortcuts, all handled in the 3D viewport.

KEY	ACTION
`Ctrl` + `Z`	Undo — works in every mode
`Ctrl` + `Shift` + `Z` / `Ctrl` + `Y`	Redo
`W` or `G`	Gizmo → move
`E`	Gizmo → rotate
`R`	Gizmo → scale
`Q`	Transform space: world ↔ local
`X`	Snapping — 0.1 m translate, 15° rotate
`O`	Orthographic ↔ perspective
`F`	Frame the selection
`Esc`	Step *out* one level: annotation → tray point or stretch → waypoint → cable → object
`Del` / `Backspace`	Delete the *most specific* thing held: annotation → tray point or stretch → waypoint → cable → object

The two ladders climb the same rungs. On a tray, Delete takes the point (and every stretch touching it) or the stretch you have hold of — and if that stretch was the only path across the network, the tray splits into one object per piece. `Esc` first if you meant the whole tray.

Annotation tools

KEY	ACTION
`X` / `Y` / `Z`	Lock the second point to a world axis; press again for the anchor object's own axis; a third time clears it
`Alt`	Hold to suppress snapping
`Esc`	Abandon the measurement, then put the tool away
`Enter`	Commit a callout (`Shift` + `Enter` for a new line)

A report sheet

While a sheet is up it owns the keyboard, exactly as the sketch editor does — the viewport's shortcuts stand down, because there is no viewport on screen to aim them at. Undo and redo are the exception: they belong to the app.

KEY	ACTION
`Ctrl` + `D`	Duplicate the selected blocks, offset and on top
Arrow keys	Nudge the selection — `Shift` for ten times the step
`Del` / `Backspace`	Delete the selected blocks
`Esc`	Close a text editor, then clear the selection
`Ctrl` +scroll	Zoom the sheet

Inside a framing session (§11.4) the viewport is back and so are its keys, including the annotation tools above.

The sketch editor

While the editor is open it owns the keyboard, so none of the shortcuts above fire — which is why the strip in the bottom-left corner changes to these:

KEY	ACTION
`V` `L` `P` `R` `S` `C`	Select · line · polyline · rectangle · square · circle
`M`	The ruler, out and away again — the pen comes back, the drawing stays open
`A`	The angle: an arm, the corner, the other arm. `Tab` swaps inner for outer
`Esc`	Step *out* one level: put the ruler away, or finish the gesture, then back to Select, then leave the editor
`Enter`	Finish an open polyline run where it is
`Tab`	Take the size readout, and move between a rectangle's two sides
a digit	Take the size readout, starting the number with what you pressed
`Shift` / `Alt` + a box drag	Add to / remove from the selection
`Alt`	Hold to draw off the grid
`Del` / `Backspace`	Delete the selected dimension, or the selected curves

Move, rotate and scale have **no keys in here**: `W`, `E` and `R` are the pen's, `R` being Rectangle. The three buttons on the left rail are how you switch between them while a drawing is open.

Why a shortcut may not fire

Check these first:

- The focus is in a text field, a number field or a dropdown.
- A modal dialog is open — every shortcut is suppressed while one is.
- You are in **Diagram** mode, where everything below undo/redo is skipped because the diagram owns `Delete` and `Backspace`.

- A **report sheet** is up, which owns the keyboard the same way — see above.
- An **annotation tool is armed**, in which case the tool owns the keyboard.
- You are on the login or projects screen, which has no viewport and no listener.

The blend-array panel is a deliberate exception: it is modeless, so shortcuts keep working while it is open.