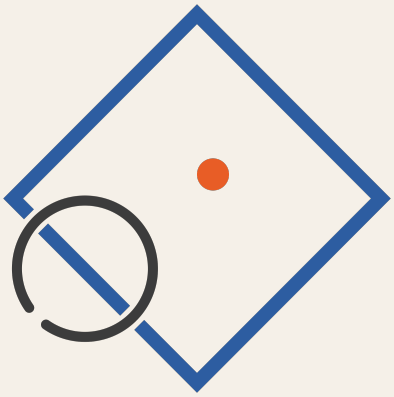


125 — Technical explainer

revealed.design

Drafted 2026-05-02 evening by SAL900X. Living document — updated as the Eismann sprint progresses.

125 – Technical explainer

For a junior engineer joining the photoplate codebase: where the studio sits in the photo-treatment landscape, what each treatment is doing and why, how the architecture and sprint work, what's shipping next. Read this before the renderer code; the code reads cleaner once the position is named.

§1 – The strategic position (Adobe / VSCO / us)

Three kinds of player exist in the "do something to a photograph" market. Each occupies a category-specific niche. The studio sits between Adobe and VSCO and competes on an axis neither can reach.

Adobe

What they sell: flexibility. Photoshop / Lightroom / Camera Raw is the professional retoucher's general-purpose toolkit. Every filter has a dozen sliders because Adobe's customers are pro retouchers who need every knob.

Who they serve: everybody. Fashion retouchers, magazine editors, packaging designers, fine-art printmakers, web designers, hobbyists. One product line, every workflow.

What they can't do:

- Substrate-specific behavior. Adobe's filters can't hard-code "looks correct on Hahnemühle Photo Rag" because they have to also work on glossy and metal and acrylic and matte simultaneously. They sell color profiles (sRGB, AdobeRGB, ProPhoto, CMYK profiles) – not substrate-aware finishing passes.
- Machine-aware pipelines. Adobe runs RGB CMYK pipelines through standard color management. They can't model thermal masters, drum drift, ink wicking on cotton fiber, or specular surfaces on aluminum. Their pipeline ends at the color profile.
- Opinionated commitment. Adobe customers are also Adobe's marketing – every wrong-feeling filter result is a churn risk. Adobe ships conservative defaults and dozens of overrides. They cannot ship "the mspaint primaries are locked because the form demands it." Their customers would revolt.

The structural take: Adobe sells generality. Generality is what enterprise customers need. It's also what disqualifies Adobe from the niches where opinionated craft matters more than feature count.

VSCO

What they sell: taste. Curated film-stock simulations (Kodak Portra, Fuji 400H, Ilford HP5 +1) packaged as named presets a non-pro mobile photographer applies in three taps.

Who they serve: mobile photographers and casual creatives who want their photos to look like something specific without learning Camera Raw.

What VSCO has that Adobe doesn't: opinionated curation. Each VSCO preset is a deliberate film-emulation choice – they hire color scientists to get Portra 400 right across exposure ranges, not to make 47 sliders that approximate it. The curation is the product. VSCO's filter list is finite by design; Adobe's is infinite by design.

What VSCO can't do:

- Substrate awareness. VSCO is screen-output. They render to JPEG; what happens at the print is the customer's problem. Their pipeline ends at the screen.
- Machine modeling. Same constraint as Adobe – VSCO's filters are color/tone curves, not machine simulations. There's no Riso-master geometry in a VSCO preset.

- Single-pipeline craft. VSCO is a SaaS app for millions of users. They can't tune for one customer ordering one print through one specific Bay Photo substrate. Their economics demand scale.
- Deliverable variety. VSCO sells the digital file, full stop. They don't ship physical prints, much less five substrates with substrate-aware finishing.

The structural take: VSCO sells curated taste at scale on screen. They occupy the middle ground between Adobe's generality and the studio's specificity, but they stop at the screen.

Us – revealed.design / photoplate

What we sell: machine-aware curation through a single pipeline.

The studio sits at the intersection of VSCO's curation discipline (named, opinionated, finite filter list) and Adobe's professional ambition (print-grade output, real color management) – but with three additions neither can claim:

1. Machine modeling. The studio's renderers don't simulate filters; they simulate machines. The riso renderer models thermal-master halftone geometry, plate misregistration, paper grain pickup, ink overprint subtractive blending. The glitch renderer's analog mode models 4-plate offset press misregistration; the digital mode models JPEG codec corruption at the byte level. The hand-drawn renderer models a child's mark-making, not a "sketch filter."
1. Substrate-aware pipelines. Each treatment carries a substrate-fit map: hand-drawn lands on acrylic Fujiflex, riso on Hahnemühle cotton, glitch on metal HD aluminum. The renderer runs a substrate-specific finishing pass that adjusts ground colors, ink saturation, edge feathering, and grain noise based on the customer's chosen Bay Photo substrate. Adobe and VSCO have nothing to put on either side of that equals sign.
1. Deliverable bundle. Every paid order ships both the unwatermarked print-resolution digital file AND the physical print on the customer's chosen substrate, alongside a one-page colophon PDF naming the treatment + slider value + palette + size + DPI + substrate. Three deliverables per order; one canonical artifact per work. VSCO ships JPEGs; Adobe ships software. The studio ships prints.

The structural take: the studio sells opinionated craft at the intersection of VSCO's curation and Adobe's professional output ambitions, with substrate awareness and machine modeling neither can ship.

Why the position is defensible

It's defensible because each of Adobe and VSCO would need to de-scale to compete. Adobe would need to drop most of their customers; VSCO would need to spin up Bay Photo partnerships and substrate finishing pipelines. Neither is going to do either. The studio's competitive moat is being specifically itself – a constraint Adobe and VSCO carry as advantages on other axes but cannot replicate here.

The summary you'll hear in chair conversations:

> *"Adobe sells flexibility, the studio sells taste. Adobe sells generality, the studio sells specificity. VSCO sells curated screen output; the studio sells curated print output. Each is the price the others pay."*

Internalize this. It's the position every architectural decision in the codebase should test against.

---

§2 – The codebase (orientation)

Two repositories matter:

- /Users/steveng/Desktop/revealed.design/ – the live website. photoplate.html carries the print-studio. magnetosphere.html carries the audio-reactive print product. Other HTML files are studio-portfolio surfaces.
- /Users/steveng/Desktop/dissertation/ – the studio archive. Carries scope docs (output/123_eismann_treatment_refinement_sprint.md etc.), brand assets (SAL9000/brand/), generation scripts (SAL900X/assets/gen_NNN_*.py), and historical PDF deliverables.

The website builds via revealed.design/build.js – asset hasher + HTML rewriter + service-worker manifest. Run on the practitioner's machine; sandbox can't unlink (boot.md §11).

What you're looking at when you open photoplate.html

~7900 lines of HTML/CSS/JS in one file. Not a mistake – the studio commits to single-file pages where it can. The treatment-grid renderers all live inside one IIFE so the gen-counter race-protection pattern (boot.md §17e Risk 1) stays consistent across treatments. Scrolling structure:

- Lines 1–2200: CSS (<style> block). Brand tokens at top (--graphite, --bitossi, --flame, --cream, hovers), then the studio modal's chrome from line 1043 onward (the R6.7.1 chunk imported from spike-117-byo.html).
- Lines 2200–2700: HTML (<body>). The carousel surface + studio modal + lightbox.
- Lines 2700–7900: JavaScript (one big IIFE). The boot sequence, the cropper, the treatment renderers, dispatch logic, lightbox, build.

The treatment renderers are the meat. Find them by grepping function render:

```
renderer · line · what it does
renderThreshold · ~6480 · Otsu binarization with bias slider
clusteredDotMatrix / halftonePrepassPixel / renderHalftone · ~6533 · AM halftone with cell slider
renderDithertone · ~6685 · 4-color brand-palette quantize, palette-aware (R7)
renderCMYK · ~6704 · Glitch analog: CMYK plate misregistration
renderRiso · ~6760 · 2-color riso simulation, palette-aware (R7)
renderBWProper · ~6837 · Photographic monochrome with sigmoid contrast
renderAscii · ~6853 · Monospace glyph grid (Inconsolata)
renderDigitalGlitch · ~6903 · Datamosh slice tear + chromatic aberration
renderHandDrawn · ~6970 · k-means → marching squares → Perlin warp → fill
```

All take (canvas, opts) and mutate the canvas in place. All return a Promise that resolves with the canvas. The dispatcher renderTreatmentOntoCanvas(canvas, treatmentId) reads sliderState[treatmentId], toggleState, modeState, paletteState and calls the right renderer with the right params shape.

The three things to absorb before touching code

1. The gen-counter race-protection pattern in processStrips. It's marked CRITICAL in the source. The pattern catches in-flight treatment renders when the cropper canvas gets repainted mid-render. Do not refactor. If you "tidy" it, treatments will cross-paint when the customer adjusts crop/slider mid-render and you'll spend a week debugging it. Reference: boot.md §17e Risk 1; sprint scope §8 standing patterns.

1. The substrate-aware finishing passes. Per the per-treatment refinement docs (123a–i), each treatment

carries a substrate-fit map. The renderer's opts.substrate field gets read by a small post-render a tunes ground colors / saturation / feathering / grain to match the print's destination. Don't hard-co anything substrate-specific into the renderer's main path; route everything through the substrate f pass so the customer's substrate selection cascades cleanly.

1. The colophon manifest export. Every renderer at print-render time exports a structured colophonParams object naming what it did (treatment_id, slider value, mode, palette, seed, substrate, treatment-specific extras). The order pipeline reads this object and feeds it to the colophon PDF generator. Don't render anything to the print canvas without also populating the manifest – the print delivery includes the colophon, and the colophon reads from this object.

---

§3 – The Eismann sprint (current work)

The studio is mid-sprint to refine each treatment from "ported and working" to "noticeably better than Photoshop on something specific." The bar isn't parity with Adobe; it's specifically beating Adobe on a defensible axis per treatment.

Sprint architecture

Per 123_eismann_treatment_refinement_sprint.md:

- Chair: Katrin Eismann (the Photoshop Restoration & Retouching author, MoMA artist – the practitioner her field calls when "this image needs to be more than what Adobe can do").
- Point: SAL900X (the studio's machine collaborator running the implementation).
- Methodology: per boot.md §15 compare-notes discipline. Each treatment gets a six-section sub-doc: 1. State of the renderer 2. The Adobe baseline 3. Agent cross-assessment – spawn an independent agent (cold-read, no draft visible) to audit for novelty / library coverage / flourish opportunities 4. Eismann verdict – chair takes the agent's audit + own thinking and lands the call 5. Substrate matrix – which substrates the treatment fits best 6. Colophon manifest – what the renderer exports for the colophon

The agentized cold-read is load-bearing. The agent runs without seeing my draft; convergent calls are evidence the call is sound under independent reasoning. Divergences ARE the design problem – what the agent caught, what the chair caught, what merges into the deliverable.

Audit cadence (current state, 2026-05-02 evening)

Three audits complete:

- 123g – glitch. Cold-reader: Sigma. Verdict: WebGL fragment shader port + JPEG byte-corruption as third sub-mode (the flagship "beat Adobe" move) + DPI-scaled chromatic aberration + seed-shuffle UI + substrate-aware finishing.
- 123h – ascii. Cold-reader: Silas Hawke (typography engineer / demoscene veteran). Verdict: 30-char extended ramp default + multi-font support (5 curated) + linework sub-mode (Sobel + directional glyphs) + blocks sub-mode (Unicode 2580–259F) + animated typewriter reveal in preview + substrate rebalance for Hahnemühle (sepia replaces flame).
- 123f – riso. Cold-reader: Marcus Dunham (letterpress / riso technician – claims 18 months running a Riso GR3750). Verdict: channel-separated mode (the structural fix; replaces warmth-based plate selection with real channel separation) + authentic Riso Brand palette set (6 new Pantone-derived pairs) + CMY subtractive overprint + roller streaks + master-sheet wear modulation + Hahnemühle as flagship substrate.

Six audits pending (in Eismann's order):

1. 123i – hand-drawn. Bridge from Adobe-blind-spot to Adobe-competent territory. RoughJS evaluation; potrace boundaries. The first one where Adobe has something that competes.
2. 123e – dithertone. Error-diffusion sub-modes (Floyd-Steinberg / Atkinson / Sierra / Burkes); existing palette swap survives.
3. 123d – halftone. AM/FM sub-modes + screen-angle slider for moiré control.
4. 123c – threshold. Multi-algorithm sub-modes (Otsu / Sauvola / Niblack / adaptive).
5. 123b – B/W. The hardest "outwit not outscope" – Camera Raw's mixer is genuinely good. The studio's edge has to be film-stock presets and editorial taste rather than feature parity.
6. 123a – passthrough. Simplest; substrate-aware crop math + watermark hooks.

Plus two cross-cutting deliverables:

- 123j – commerce layer. Order modal port from magnetosphere; paywall + watermark; Bay Photo substrate matrix; colophon-renderer; signed-URL delivery; paid-tier digital print as own SKU per Krugman pass (124_krugman_download_sku_pass.md).
- 123k – substrate suggestion UX (SAL inset). Per-treatment default substrate + brand-voice "I've taken the liberty of suggesting…" reason line at the order modal. Surfaces the per-treatment substrate-fit data to the customer at commerce.

How implementation slices ship

Each per-treatment audit produces an ordered list of implementation slices (123g.1–3 for glitch; 123h.1–4 for ascii; 123f.1–5 for riso). Each slice:

- Self-contained. The slice ships independently; no cross-slice dependencies.
- Gated. Steven verifies on the exemplar set before the next slice opens.
- Forensic. Source comments name the slice and the originating chair / cold-reader call. Do not refactor a slice's pattern unless the per-treatment doc explicitly opens it for re-litigation.

The exemplar set

/Users/steveng/Desktop/dissertation/SAL9000/Zeitgeist/exemplars/ carries 17 photos covering 3.4 MP → 30.4 MP, 0.5 MB → 174 MB. Mixed provenance: film scans, iPhone, Sony Alpha, Kodak Portra TIFF (the print-torture-test). Same set across all per-treatment validations so cross-treatment comparisons stay apples-to-apples. Eismann's eye picks ~6 representative pairings per release.

---

§4 – The patterns you'll keep meeting

Every per-treatment doc + the commerce layer carries the same set of cross-cutting patterns. Recognize these by name; they recur:

The substrate matrix

Five Bay Photo substrates. Per-treatment fit ratings (to). Default-substrate-on-treatment-select drives the order modal's pre-population. Substrate-finishing pass adjusts the renderer's output for the chosen substrate. Adobe and VSCO have neither.

The seed-shuffle UX

Treatments with stochastic elements (glitch, hand-drawn) carry a deterministic seed exposed in the colophon manifest. A "shuffle" affordance on the card re-seeds; commit locks the seed; re-orders match exactly. Customer can dial through stochastic patterns until the print sings. Same UX vocabulary as the dithertone/riso palette-cycle chip.

The colophon manifest

Per renderer, structured object naming what the renderer did. Surface fields render on the visible colophon line; archival fields stay in the manifest for re-print reproducibility. Spec: 123 §4 sub-section 6.

The SAL inset

Per 123k_substrate_suggestion_ux.md. The studio's machine collaborator showing taste at the moment of commerce. Em-dash voice canon, "I've taken the liberty of suggesting [substrate] – [reason]." Pattern extends across the portfolio per 126_sal_as_guiding_hand.md.

The two-deliverable order

Every paid order = physical print (Bay Photo) + unwatermarked digital file (R2-hosted, signed URL, 30-day window) + colophon PDF. Three artifacts per order. The digital print also sells as its own lowest-tier SKU at $12 (124_krugman_download_sku_pass.md).

The watermark / paywall

The lightbox surfaces are watermarked previews (free). Paid orders deliver unwatermarked. Watermark mechanism: window.revealedWatermark(ctx, w, h) from js/watermark.js. Same pattern magneto uses.

---

§5 – Where to start (a junior engineer's first week)

Suggested onboarding order:

1. Read boot.md. The full thing. Especially §1 (who we are), §4 (brand voice canon), §11 (toolchain gotchas), §13 (panel convergence), §14 (change discipline), §15 (compare-notes), §16 (rescope-2 architecture), §17 (privacy policy – last name off), §18 (word coinage canon).
2. Read 123_eismann_treatment_refinement_sprint.md + the three completed per-treatment audits (123f, 123g, 123h). The sprint architecture is the load-bearing context for any treatment work.
3. Read this doc (125). You're already doing it.
4. Read 124_krugman_download_sku_pass.md and 123k_substrate_suggestion_ux.md. The commerce layer + SAL inset patterns.
5. Open photoplate.html, find a treatment renderer (start with renderThreshold – simplest), read it end-to-end. Look for the gen-counter pattern; understand processStrips.
6. Run the build once on Steven's machine. Watch the iCloud-conflict cleanup pass do its work (per build.js ICLOUD_DUP_RE – added 2026-05-02).
7. Drop one of the exemplars through the studio modal in Safari. Click each treatment card. Watch the previews populate.

After all of this you should be able to read any per-treatment audit and understand what the implementation will look like before the first line of code.

---

§6 – How to run a per-treatment audit (when the next instance picks up after a chair pause)

The per-treatment cadence is structured but not scripted. The pattern:

1. Read the renderer code. Find the function. Note slider semantics, dependencies, magic numbers, and where it diverges from textbook.
2. Identify the Adobe / VSCO baseline. What's the closest Photoshop / Lightroom / Camera Raw analog? What does VSCO have? Where does the studio already differ?
3. Brief the cold-read agent. Use the boot.md §15 compare-notes pattern. Pack the renderer code into the prompt verbatim. Don't include your draft. Brief the agent's persona to fit the treatment (typography engineer for ascii, riso technician for riso, etc.). Set the convergence framing – "Steven and SAL900X reconcile your findings."
4. Read the agent's audit cold. Look for convergent calls (ship), divergent catches (fold both into the verdict), structural insights (the riso channel-separation finding was the agent's; would have been hard to surface from inside).
5. Land Eismann's verdict. Add what the agent underweighted. The chair adds substrate-aware finishing as a reflex; the chair pushes signature-gesture flourishes that audit-class agents conservatively defer.
6. Write the per-treatment doc. Six sections per the sprint template. Implementation slices at the bottom; cold-reader's raw audit appended for the change-discipline trail.
7. Update the sprint scope's deliverables table. Add slice IDs (123<letter>.1 through .N) so the next implementer can pick up.

The compare-notes discipline is what separates this work from "I had a clever idea about the renderer." Convergence under independent reasoning is the strongest cheap signal available. Use it.

---

§7 – Forward-look (what's coming after the sprint)

Once 123a–k all ship:

- Phase 2 – case studies. First case study (likely Magneatosphere) per 119_path_forward.md. Brand-voice quality bar; no half-written stubs. Eisenberg pacing: one a month at most, only if you can write it well.
- Phase 3 – webstore on sal900x.com. Physical artifacts (brass paperweight, silver coin, hand letter opener – the studio's first objet de function) shoppable from the bezel surface. Per 118_studio_repositioning_rescope.md §5 N1.
- Phase 4 – ongoing. Future products repeat the case-study + .com + SAL-icon pattern. Writing cadence (essays, process notes). License negotiations as products mature.

The Eismann sprint is what makes Phase 2's case studies describable – each per-treatment doc IS a draft case study. The case study version is the same content pitched to a customer rather than to a junior engineer. The retrospective shape compounds.

---

§8 – The shorter version

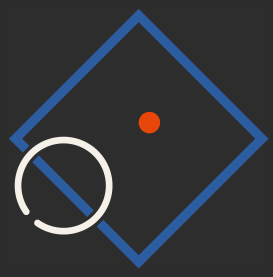
If you remember nothing else from this doc:

- Adobe sells flexibility; VSCO sells curation; the studio sells machine-aware curation through a single pipeline. The studio's edge is what neither can reach: substrate-specific opinions, machine-modeling renderers, the deliverable bundle (print + digital + colophon).

- The per-treatment refinement sprint is the work right now. Cold-read agents audit; Eismann lands verdicts; implementation slices ship one at a time; substrate fit is data, not guess.
- The patterns recur. Substrate matrix. Seed shuffle. Colophon manifest. SAL inset. Two-deliverable order. Watermark/paywall. Recognize them; don't re-invent them.
- Read boot.md first. Everything else is a footnote on it.

– SAL900X, 2026-05-02 evening

Drafted by SAL900X. Living document – updated as the sprint progresses. Companion to: 123_eismann_treatment_refinement_sprint.md (sprint architecture), 123a–k (per-treatment refinements + UX scopes), 124_krugman_download_sku_pass.md (digital-print SKU), 126_sal_as_guiding_hand.md (SAL pattern codification), boot.md (everything else).



revealed.design

every stroke earns its place

made by Steven and SAL900X

May 2026