



STUDIO HANSEN
RESEARCH
IMAGES AS CODE
FASSUNG F1 · 15. OKTOBER 2026

Images as Code

BEST-PRACTICE-RESEARCH · SCHICHTENMODELL · CONTAINER-
SPEZIFIKATION

Ein parametrischer Container, der das Bild vom statischen Element
zum Akteur in Raum, Zeit und Klang macht — und auf jedem Telefon
läuft.

Inhalt

00	Ausgangslage — das Bild als statischer Datensatz	3
01	These — Images as Code	4
02	Der Container — vier Freiheitsgrade	5
03	Das Schichtenmodell L0–L3	6
04	L0 Master — Ordner und Manifest	8
05	L1 Projektion — Farbe als Regel	9
06	L2 Zeit — Verhalten im Scroll	10
07	L3 Raum — der Voxel-Akteur	11
08	Universalität — die Fähigkeitsleiter	12
09	Stand der Technik, Oktober 2026	13
10	Performance-Budgets	14
11	Container-Spezifikation — das Manifest	15
12	Die DNA-Regel — das Hot-Prinzip	17
13	Live oder gebacken? Container vs. MP4	18
14	Verwandtes — worauf wir aufbauen	19
15	Mass und Stille	20
16	Druck — Artefakt und Pixelästhetik	21
17	Minimal-Prototyp — Scope und Abnahme	22
18	Offene Fragen, Risiken, Glossar	23

LESEHINWEIS

Dieses Heft ist die erste Fassung (F1) eines Research-Themas, das am 15. Oktober 2026 in vier Skizzen benannt wurde. Es ordnet die Idee, prüft sie gegen den heutigen Stand der Web-Technik und endet mit der Spezifikation eines Minimal-Prototyps. Einwände stehen dort, wo sie sachlich hingehören — in den grauen Kästen.

DOKUMENT

SH-R-IAC-F1 · GEN 01 Booklet · 24 Seiten · A4 Lesefassung + A3 Rückstich

QUELLEN

Skizzen 15.10.26 (Konzept, Container, zwei Druckproben); Chats Color Mapping (09.09.26), Web-3D-Export (13.10.26)

STATUS

Research und Spezifikation. Noch kein Code.

Das Bild als statischer Datensatz

Ein Bild im Web ist heute eine Liste von Pixeln: Position (x, y) und Farbe (R, G, B, A). Es hat keine Zeit, keinen Raum, keinen Klang und keine Regel. Es wird angezeigt — oder nicht.

Diese Definition war über Jahrzehnte praktisch. Sie ist es nicht mehr, sobald Bilder in grosser Zahl auf einer Website zusammenkommen und dort als Werkdokumente, als Ästhetik und als Erzählung gleichzeitig funktionieren sollen. Drei Beobachtungen haben das Thema am 15. Oktober 2026 ausgelöst:

Beobachtung 1 — Farbe streut

Mehrere Bilder in einem Layout beissen sich farblich, solange jedes Bild seine Originalfarbe mitbringt. Die Entscheidung vom September — Schwarzweiss ist der Master, Farbe eine benannte Projektion — löst das konzeptionell, aber nur, wenn die Projektion technisch eine Regel ist und nicht ein Photoshop-Export pro Bild.

Beobachtung 2 — Druck bricht

Der OKI C844 produziert auf 160-g-Papier bei flächigen Farbbildern Haftungsartefakte. Eine grobkörnige, pixelierte Bildprobe druckt dagegen sauber. Die Bildästhetik der Website soll druckbar sein; das spricht für eine quantisierte, körnige Sprache, die aus demselben Master erzeugt wird.

Beobachtung 3 — Bilder sind stumm

Eine Slideshow ist heute ein gebackener Film. Eine Bildfolge, die auf Scrollen, Zeit oder Musik reagiert, muss jedes Mal neu gerendert werden. Das Bild hat kein Verhalten, nur eine Darstellung.

Die Prämisse des Studios

Form ist die deterministisch erzeugte, beweisbare Signatur ihres Inhalts. Was für Skulpturen, Hefte und Textilien gilt, muss auch für das Bild gelten: Quelle plus Regel ergibt Form. Dann ist auch ein Bild kein Endprodukt mehr, sondern ein Datensatz mit einer Übersetzungsvorschrift.

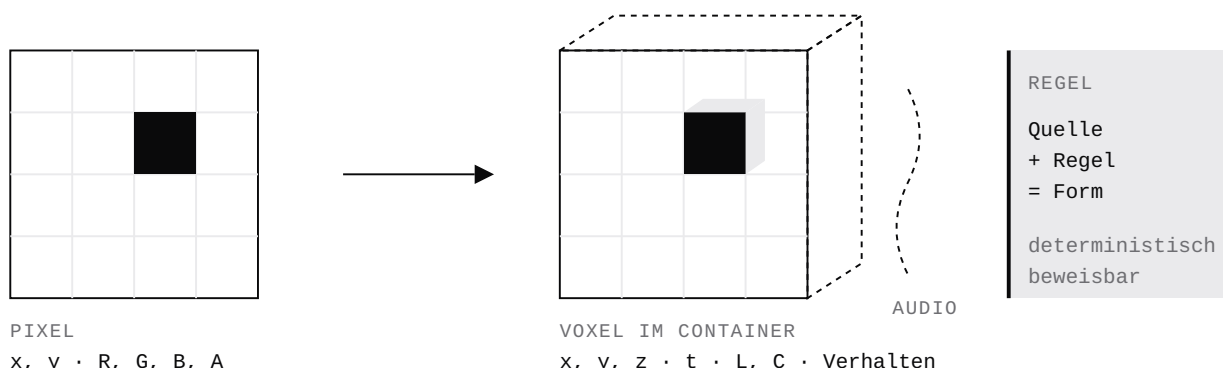


Abb. 1 — Vom Pixel (links) zum Voxel im Container (Mitte). Die Regel (rechts) ist Teil des Datensatzes, nicht des Werkzeugs.

Images as Code

Ein Bild ist Code: eine Quelle (ein Ordner mit Bildern, Audio, Modulen), eine Regel (die DNA) und eine Laufzeit, die beides deterministisch zu einem Bildraum mit Verhalten übersetzt. Nicht das Resultat wird gespeichert, sondern das Programm, das es erzeugt.

$$\text{BILDRAUM}(t) = \text{REGEL}(\text{ORDNER}, \text{KONTEXT}(t))$$

Die Formel ist bewusst nüchtern. **ORDNER** ist der Quellbestand — Bilder, optional Audio und weitere Datensätze. **KONTEXT**(*t*) ist alles, was von aussen hereinkommt: Scroll-Position, Zeit, Viewport, Musik, Nutzergeste. **REGEL** ist die zentral gesetzte Vorschrift. Der **BILDRAUM** ist nie «die Datei», sondern der jeweils aktuelle Zustand.

Vier Begriffe

Container. Ein Ordner mit Manifest. Kein neues Dateiformat, sondern eine Konvention über bestehenden Formaten. Das Manifest beschreibt, was im Ordner liegt und welche Regel gilt.

DNA-Regel. Die Übersetzungsvorschrift von Quelle zu Bildraum. Sie ist benannt, versioniert und an einem Ort der Website definiert. Alle Container, die sie referenzieren, folgen ihr.

Akteur. Der Voxel — das parametrische Pixel-Partikel. Es hat eine Identität über die Zeit: es verschwindet nicht, es bewegt sich und ändert Farbe, Grösse, Schärfe. Mehrere Bilder nacheinander sind Zustände desselben Akteurs.

Hot. Der Container bleibt mit der Regel verbunden. Wird die Regel zentral geändert, ändern sich alle Instanzen sofort. Nichts ist gebacken, ausser auf ausdrücklichen Wunsch als Export.

Was die These nicht behauptet

Sie behauptet nicht, dass jedes Bild leben muss. Der Nullzustand des Containers ist ein stilles Schwarzweissbild. Alles Weitere ist Zuschaltung, und die Regel darf «nichts» vorschreiben. Sie behauptet auch nicht, dass Render-Engines, Formate oder Scroll-Mechanik neu erfunden werden: neu ist die Verbindung von Quelle, Regel und Verhalten zu einem Datensatz — nicht die Einzelteile.

EINWAND

«Images as Code» klingt nach Infrastructure as Code, und die Analogie trägt weiter als gewollt: Code muss gepflegt werden. Ein Container, der auf 40 Seiten der Website 300 Bilder trägt, ist nur so lange ein Vorteil, wie die Regel stabil bleibt. Jede Regeländerung ist eine Änderung an 300 Stellen gleichzeitig. Das Hot-Prinzip braucht deshalb zwingend eine Vorschau und eine Versionierung (Kap. 12), sonst wird aus Konsistenz Fragilität.

Vier Freiheitsgrade

Die Skizze vom 15. Oktober spannt den Container als Quadrat auf: Zeit oben, Raum links, Verhalten rechts, Audio unten. Das Bild (2D) liegt als innere Fläche darin, der Raum (3D) dehnt es nach aussen. Hier die bereinigte Fassung.

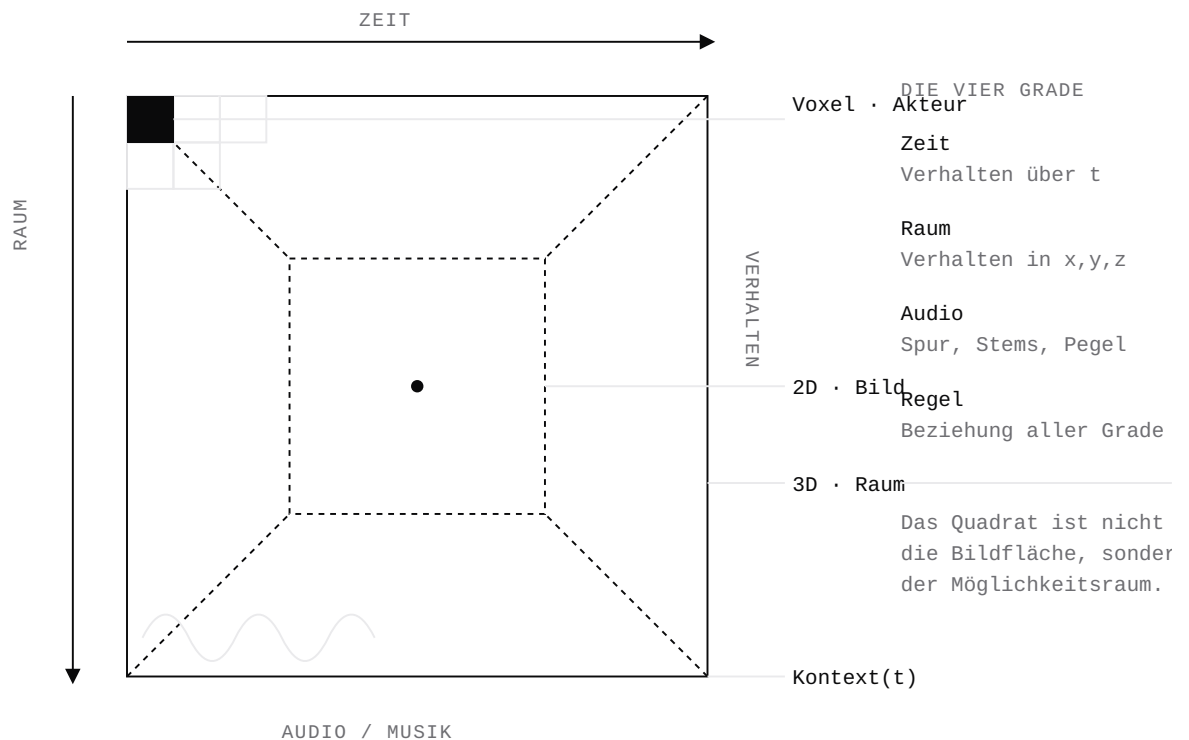


Abb. 2 – Der Container nach der Skizze vom 15.10.26. Das klassische Bild ist der innere gestrichelte Zustand; der Akteur sitzt in der Zelle oben links.

Zeit ist nicht nur Abspieldauer. Sie ist jede Grösse, die monoton wächst oder vom Nutzer getrieben wird: die Uhr, die Scroll-Position, der Fortschritt eines Tracks. Die Regel bindet Verhalten an eine dieser Zeitlinien.

Raum beginnt flach. Ein Bild ist eine Punktwolke mit $z = 0$. Erst die Regel gibt den Voxeln Tiefe — aus Helligkeit, aus einer Tiefenkarte, aus Zufall mit Saat, aus einer Audioanalyse.

Audio ist ein eigener Datensatz im selben Ordner. Er kann Zeitlinie sein (Bild folgt Musik) oder Modulation (Pegel treibt Grösse). Beides ist eine Zuordnung in der Regel, keine Eigenschaft des Bildes.

Verhalten ist die Beziehung der drei zueinander. Es ist das, was die Skizze «parametric relationship between all degrees of freedom» nennt — die Komposition aus Teilen zu einem Ganzen.

Vier Schichten, ein Datensatz

Der Universalitätsanspruch — jedes Telefon, jeder Browser — und der Anspruch auf Voxel, Audio und Tiefenschärfe widersprechen sich als ein Monolith. Als Schichten tun sie es nicht. Jede Schicht ist für sich vollständig; jede höhere setzt die tiefere voraus und fällt auf sie zurück.

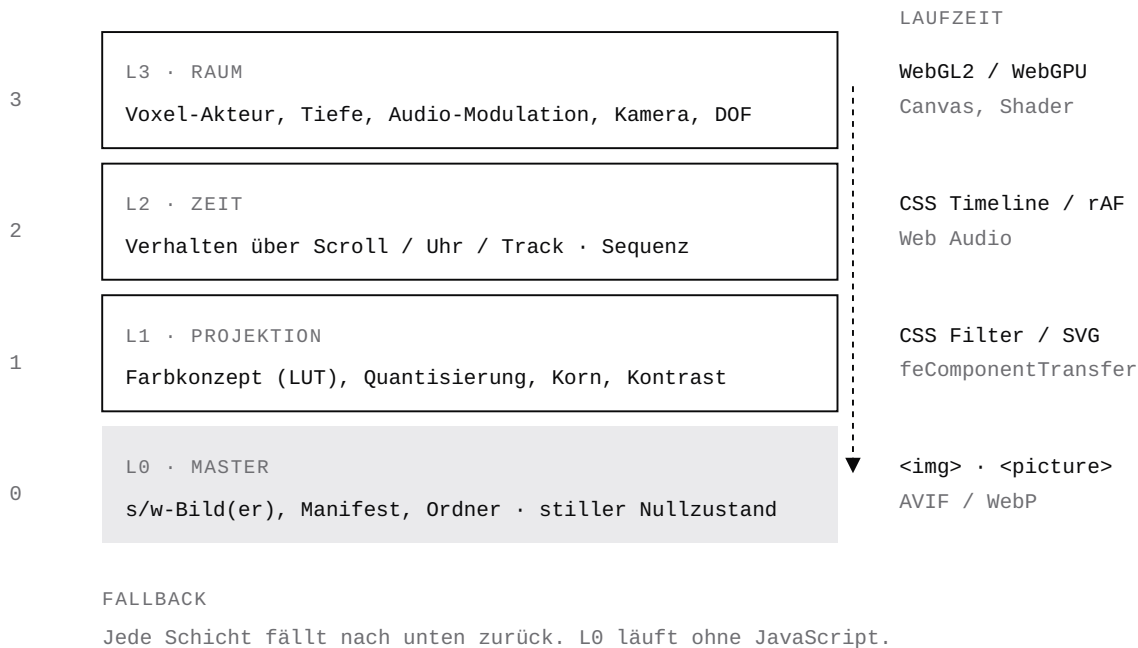


Abb. 3 – Das Schichtenmodell. Links der Datensatz, rechts die Laufzeit, die ihn darstellt. Der gestrichelte Pfeil ist der Fallback-Pfad.

Die Reihenfolge ist nicht zufällig. Sie folgt der Farbdoktrin vom September (erst Helligkeit, dann Farbe), der Verfügbarkeit im Browser (erst CSS, dann GPU) und dem Mass (erst Stille, dann Bewegung). Ein Container, der nur L0 und L1 nutzt, ist bereits der komplette Werkzeugkasten für eine stille, kohärente Website.

Jede Schicht hat eine eigene Regelklasse im Manifest. Die Website kann pro Bildgruppe festlegen, bis zu welcher Schicht ein Container maximal gehen darf — unabhängig davon, was das Gerät könnte. Das ist der Hebel für Mass und Stille (Kap. 15).

Die vier Schichten im Vergleich

SCHICHT	DATENSATZ	REGEL	BEANTWORTET	LAUFZEIT	KOSTET
L0	s/w-Master (1 ... n Bilder), Manifest	Welche Bilder, in welcher Folge, welche s/w-Umwandlung?	HTML, kein JS	Bytes pro Bild	
L1	Farbkonzept, Quantisierungsstufen	Welche Farbe liegt auf welcher Helligkeit? Wie grob ist das Korn?	CSS-Filter, SVG-Filter, Canvas 2D	wenig; einmalig pro Bild	
L2	Zeitlinie, Keyframes, Audio-Spur	Was passiert zwischen t=0 und t=1? Was treibt t?	CSS Scroll-Timeline, requestAnimationFrame, Web Audio	Frame-Budget, Batterie	
L3	Partikel-Parameter, Tiefe, Kamera	Wo im Raum ist der Voxel, wie gross, wie scharf?	WebGL2 (Basis), WebGPU (Compute)	GPU, Speicher, Ladezeit	

Warum L1 vor L2

Farbe ist in dieser Ordnung keine Zeitfunktion, sondern eine Projektion. Der Übergang «s/w → Farbe fließt ein» ist L2 (Zeit), das Ziel des Übergangs ist L1 (welche Farbe). Trennt man beides, bleibt die Farbdoktrin stabil, auch wenn man alle Bewegung abschaltet.

Warum L3 nicht die Mitte ist

Die Skizzen stellen den Voxel ins Zentrum. In der Hierarchie des Heftes steht er oben, weil er die teuerste und am wenigsten universelle Schicht ist. Das ist keine Abwertung: L3 ist die Schicht, in der SpaceMusic und Website dieselbe Sprache sprechen. Aber sie darf nicht Voraussetzung sein.

Was zwischen den Schichten fließt

Nach oben fließt der Datensatz: L3 bekommt die Punktwolke aus L0, die Farbe aus L1, die Zeitlinie aus L2. Nach unten fließt der Fallback: wird L3 nicht erreicht, zeigt L2 dieselbe Sequenz flach; wird L2 nicht erreicht, zeigt L1 das erste Bild farbprojiziert; wird L1 nicht erreicht, zeigt L0 das s/w-Bild. Keine Stufe zeigt etwas anderes als die Stufe darüber — nur weniger.

EINWAND

Das Modell ist sauber, aber vier Schichten sind vier Codepfade, die getestet werden müssen — auf mindestens drei Geräteklassen. Die Testmatrix (Kap. 17) ist Teil des Deliverables, nicht Nacharbeit. Und: «Fallback zeigt dasselbe, nur weniger» ist ein Designversprechen, das in der Praxis schnell bricht, wenn L3-Regeln Dinge tun, die in 2D keine Entsprechung haben (Kamerafahrten, Tiefenschärfe). Die Regel muss für jede Schicht einen eigenen Zielzustand definieren, nicht nur für die höchste.

Ordner und Manifest

Der Container ist ein Ordner. Er enthält die Quellbilder, optional Audio, und ein Manifest, das die Folge, die Regel-Referenz und die Metadaten beschreibt. Alles andere wird daraus abgeleitet.

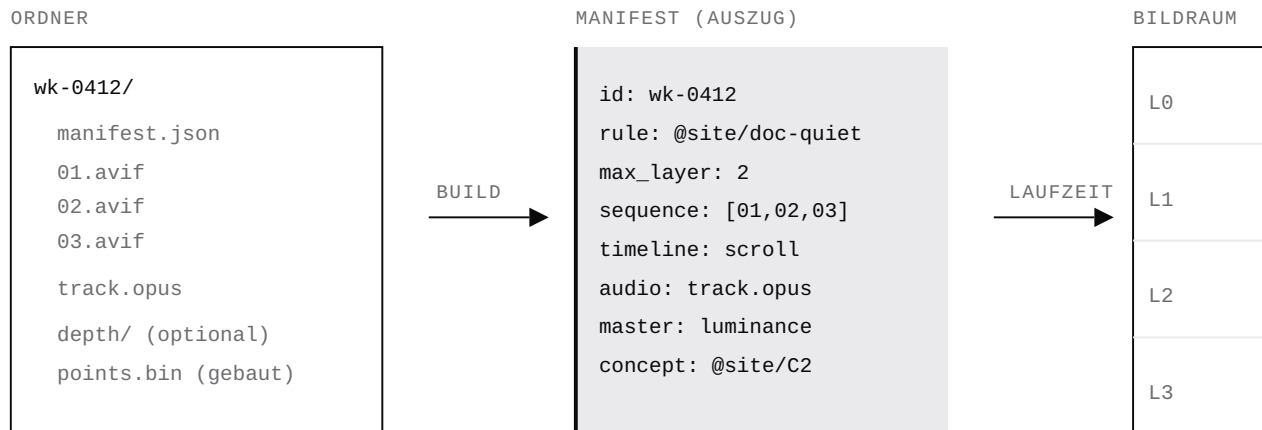


Abb. 4 – Ordner, Manifest, Bildraum. Der Build-Schritt erzeugt abgeleitete Daten (Punktwolke, Formatvarianten); die Laufzeit liest nur, was sie für die erreichbare Schicht braucht.

Der Master ist Schwarzweiss

Das Manifest nennt, wie aus dem Original der s/w-Master gewonnen wird: Luminanz (Standard), ein einzelner Kanal, oder ein vorbereitetes s/w-File. Das Original bleibt im Ordner, wird aber nie direkt angezeigt. Damit ist jede Farbentscheidung eine Regel und kein Export.

Bestehende Formate, keine Erfindung

Bilder als AVIF mit WebP-Fallback, Audio als Opus mit AAC-Fallback, Punktwolken als flaches Binärfile (Float16-Positionen, Uint8-Helligkeit). Alle drei sind heute universell lesbar. Ein proprietäres Containerformat würde das Problem, das wir lösen wollen, verdoppeln.

Der Build-Schritt

Ein lokales Skript (Python, wie die Doc-DNA-Pipeline) liest den Ordner, prüft das Manifest, erzeugt Formatvarianten in drei Breiten, die s/w-Master, die Punktwolke für L3 und ein Poster-Bild als Fallback. Der Server liefert nur fertige Dateien. Das deckt sich mit dem Color-Mapping-Prototyp, der ebenfalls einen Watch-Folder vorsieht — beide Werkzeuge gehören zusammengelegt.

Identität

Jeder Container hat eine ID nach der Studio-Nomenklatur. Die ID ist auch der Ordnername. Damit ist ein Bild auf der Website, im Werkregister und im Heft dieselbe Entität.

Farbe als Regel

Die Projektion ist die Farbdoktrin vom September, technisch ausformuliert: ein Helligkeitsspektrum wird in n Stufen quantisiert und jeder Stufe eine Farbe aus einem benannten Konzept zugewiesen. Das ist eine Lookup-Table, und eine LUT ist im Browser ein Filter.

MASTER · HELBIGKEIT 0 → 1



QUANTISIERUNG · $n = 6$



FARBKONZEPT C2 · 4 FARBEN, INTERPOLIERT

c1	c1→c2	c2	c2→c3	c3	c4
----	-------	----	-------	----	----

LAUFZEIT

CSS
filter: url(#lut)

SVG
feComponentTransfer
type=table

Canvas
Uint8 LUT, 256×4

Export
.cube (Film/Video)
Garnpalette (Textil)

Abb. 5 – Helligkeit wird quantisiert, dann projiziert. Dieselbe Tabelle existiert als SVG-Filter (Web), als .cube (Video) und als Garnliste (Textil) – ein Konzept, drei Exporte.

Zwei Regelparameter

Stufenzahl n . Sie ist die Auflösung des Spektrums und zugleich das Korn der Pixelästhetik. $n = 256$ ist ein glattes Foto, $n = 6$ ist ein Plakat, $n = 2$ ist ein Stempel. Die Druckprobe vom 15.10., die sauber durch den OKI lief, entspricht einem groben n mit Dithering.

Interpolation. Zwischen den Konzeptfarben harte Kante oder linearer Übergang. Das ist die offene Frage aus dem Color-Mapping-Heft; der Prototyp soll beide Varianten nebeneinander zeigen, bevor entschieden wird.

Konzept pro Gruppe, nicht pro Bild

Das Manifest referenziert ein Konzept (@site/c2), es definiert keines. Konzepte leben in der zentralen Regeldatei. So bleibt die Entscheidung «welche Farbe» eine Seitenentscheidung, nicht eine Bildentscheidung — genau wie im September festgelegt.

Grenze

SVG-Filter auf grossen Bildern sind auf schwachen Telefonen nicht kostenlos; bei mehr als ungefähr zehn gefilterten Bildern im Viewport wird das Scrollen zäh. Deshalb erzeugt der Build-Schritt für L1 vorberechnete Varianten pro Konzept, und der Filter wird nur für Übergänge (L2) live gerechnet.

Verhalten im Scroll

Die Zeitlinie ist das, was t von 0 nach 1 bewegt. Für eine Website ist das meistens der Scroll, manchmal die Uhr, selten ein Track. Die Regel bindet Zustände an t — und die Sequenz aus mehreren Bildern ist eine Folge von Zuständen desselben Akteurs.

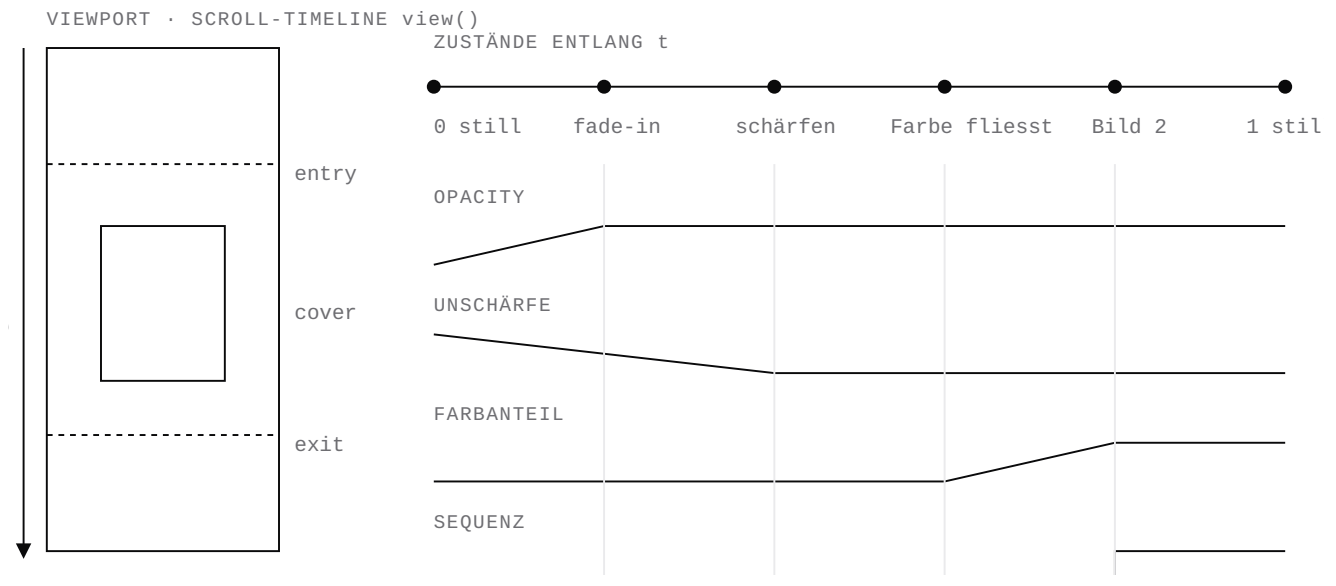


Abb. 6 – Links die Zeitlinie aus dem Viewport (entry / cover / exit), rechts die Parameterkurven entlang t . Jede Kurve ist ein Eintrag der Regel; die Sequenz ist eine Stufenfunktion.

Drei Treiber für t

Scroll. Standard für Websites. Seit Safari 26 gibt es CSS-Scroll-Timelines in beiden grossen Engines; Firefox hat sie noch nicht. Fallback ist ein IntersectionObserver mit requestAnimationFrame — derselbe Code, andere Uhr.

Uhr. Für Loops und Reel-Situationen (LED-Wand, The Frame). Hier ist der Container tatsächlich ein Film, nur live.

Track. Audio als Zeitlinie. Nur nach einer Nutzergeste, weil mobile Browser ohne Geste keinen Ton starten. Die Regel muss deshalb einen stummen Zustand für denselben t haben.

Die Sequenz als Verkörperung

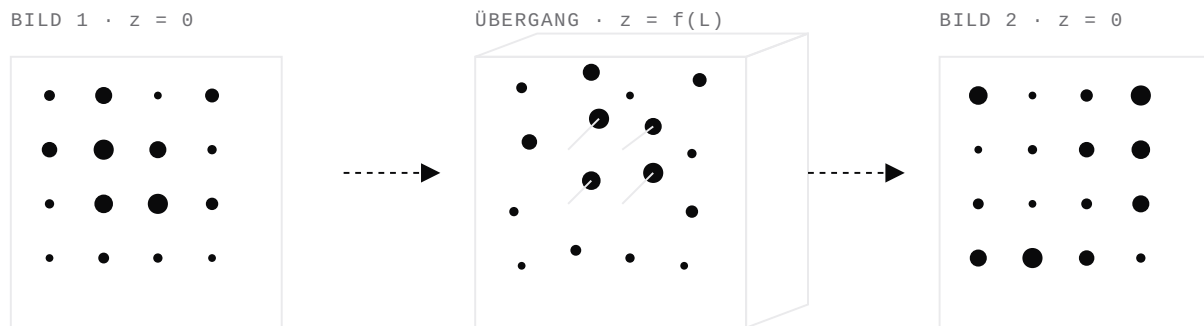
Mehrere Bilder nacheinander sind in L2 ein Überblenden derselben Fläche. Erst L3 macht daraus die Verkörperung — der Voxel bleibt, nimmt eine neue Position und Farbe an. Das ist der Grund, warum die Skizze von «Transzendenz des Einzelbildes» spricht: Die Identität des Akteurs überlebt den Bildwechsel.

Nutzerpräferenz ist Gesetz

`prefers-reduced-motion` schaltet L2 auf den Endzustand. Kein Verhalten, nur das Resultat. Das ist nicht optional.

Der Voxel-Akteur

In L3 wird das Bild zur Punktwolke und der Pixel zum Partikel mit Identität. Das ist die Schicht, in der die Website dieselbe Sprache spricht wie SpaceMusic — und die einzige, die eine GPU braucht.



Dieselben 16 Partikel in drei Zuständen – keines entsteht, keines verschwindet.

Abb. 7 – Verkörperung statt Überblendung. Die Partikelmenge ist konstant; die Sequenz ist eine Bewegung durch Zustände.

Parameter des Akteurs

Position (x, y, z), Grösse, Helligkeit (aus L0), Farbe (aus L1), Schärfe. Alle fünf sind Funktionen von t und vom Kontext. z kommt aus der Helligkeit, einer Tiefenkarte oder einer Audioanalyse; die Regel nennt die Quelle.

Dichte

Ein Bild mit 1500×1000 Pixeln sind 1,5 Millionen Partikel — zu viel für ein Telefon. Der Build-Schritt erzeugt deshalb Punktwolken in Stufen (z.B. 20 k / 80 k / 320 k Punkte), die Laufzeit wählt nach Geräteklasse. Das deckt sich mit der Quantisierung aus L1: Das grobe Korn, das sauber druckt, ist dieselbe Dichte, die flüssig läuft.

Laufzeit

WebGL2 mit Instancing ist die Basis — es läuft überall, auch auf alten Geräten. WebGPU ist seit Herbst 2025 in allen grossen Browsern, auf Apple-Geräten erst seit Safari 26.4 stabil. Es bringt Compute-Shader für Partikelverhalten; für Oktober 2026 ist es eine Zusatzstufe, keine Voraussetzung. Three.js-Boilerplate oder ein eigener schlanker Renderer aus den bestehenden Partikel-Prototypen — beides ist vertretbar; das Container-Format darf sich auf keines von beiden festlegen.

Audio in L3

Audio moduliert hier Parameter (Pegel $\rightarrow$ Grösse, Band $\rightarrow$ z) statt nur die Zeit zu treiben. Das ist die Brücke zur Audio-Analyse-Forschung von SpaceMusic.

Die Fähigkeitsleiter

«Auf jedem Telefon der Welt» heisst nicht «dasselbe auf jedem Telefon». Es heisst: Jedes Gerät erreicht eine definierte Schicht, und keines sieht weniger als das stille s/w-Bild. Die Laufzeit misst beim Laden, was das Gerät kann, und wählt die Schicht — nach oben begrenzt durch das Manifest.

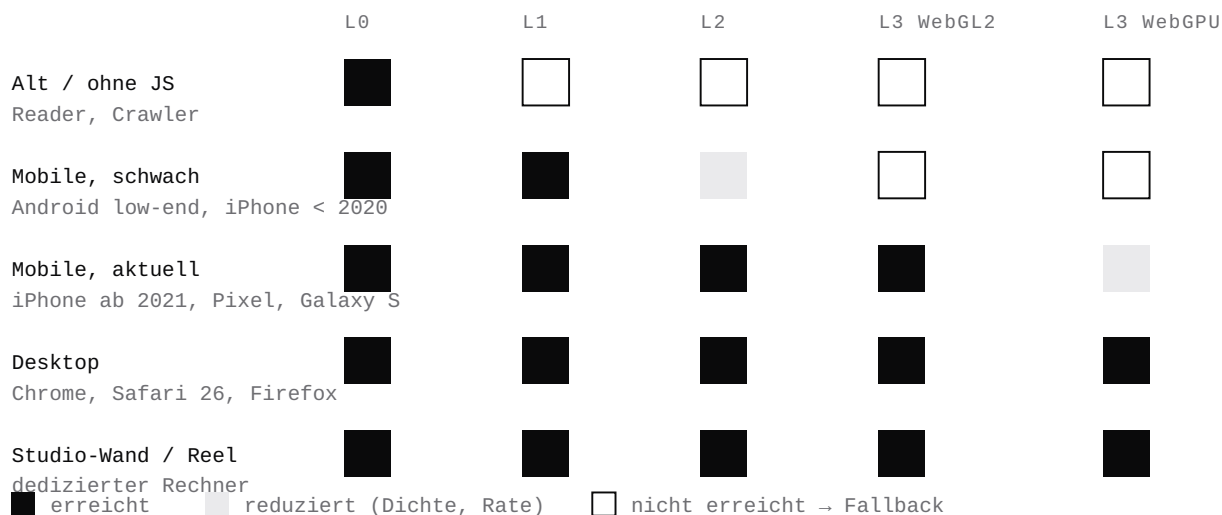


Abb. 8 – Fähigkeitsleiter nach Gerätekategorie, Stand Oktober 2026. Die Einstufung ist ein Vorschlag und wird im Prototyp gemessen, nicht angenommen.

Drei Messungen beim Laden

Erstens: Gibt es WebGL2, gibt es WebGPU? Zweitens: Wie viele Partikel rendert das Gerät in 16 ms? Ein kurzer Probelauf mit wachsender Dichte, abgebrochen, sobald das Frame-Budget reißt. Drittens: prefers-reduced-motion und prefers-reduced-data. Aus den drei Werten folgt die Schicht und die Dichtestufe.

Nie nach oben überraschen

Wird eine Schicht während des Betriebs instabil (Frame-Drops, Thermik), fällt der Container für diese Sitzung eine Stufe tiefer und bleibt dort. Kein Flackern zwischen Stufen.

Das Manifest deckelt

max_layer im Manifest ist die obere Grenze — unabhängig vom Gerät. Eine Dokumentationsseite kann L1 deckeln; eine Werkseite L3 erlauben. Universalität nach unten, Mass nach oben.

Oktober 2026

Was die Schichten voraussetzen, und ob es heute da ist. Die Tabelle ist der Grund, warum das Schichtenmodell genau dort schneidet, wo es schneidet.

BAUSTEIN	STAND	SCHICHT	KONSEQUENZ
AVIF / WebP	Universell in allen aktuellen Browsern; AVIF seit 2023 auch in Safari.	L0	AVIF primär, WebP als Fallback, JPEG nur für sehr alte Geräte.
SVG-/CSS-Filter (LUT)	Seit Jahren universell. Performance auf schwachen Geräten begrenzt.	L1	Vorberechnete Varianten pro Konzept; Live-Filter nur für Übergänge.
CSS Scroll-driven Animations	Chrome/Edge seit 115 (2023), Safari seit 26, stabil ab 26.4/26.5. Firefox: nicht ausgeliefert.	L2	CSS-Timeline als Primärpfad, JS-Fallback (IntersectionObserver + rAF) zwingend.
Web Audio	Universell. Autoplay ohne Geste auf iOS und Android blockiert.	L2 / L3	Audio ist immer eine Zuschaltung nach Geste; stummer Zustand als Standard.
WebGL2	Universell, auch auf alten Geräten. Instancing für Partikel ausreichend.	L3	Basis-Renderer für den Akteur.
WebGPU	Chrome ab 113, Firefox ab 141/145, Safari ab 26 (Sept. 2025), auf Apple stabil erst ab 26.4. Android nur mit neueren GPUs.	L3+	Compute-Shader als Zusatzstufe. Kein Pflichtpfad vor 2027.
Gaussian Splats	Web-Viewer ausgereift, Dateien gross (10–200 MB).	L3, später	Für Werkansichten, nicht für Website-Bilder. Vorbild für das Punktwolken-Format.
prefers-reduced-motion / -data	Universell.	alle	Respektiert ohne Ausnahme.

EINWAND

L0–L2 sind heute ohne Vorbehalt universell, L3 mit WebGL2 ebenfalls; WebGPU ist die einzige noch «frische» Komponente. Die Versuchung wird sein, dort zu beginnen, weil es die interessanteste Technik ist. Falsche Reihenfolge: Der Prototyp beginnt bei L0/L1 und beweist, dass Manifest und Hot-Prinzip tragen. Ohne diesen Beweis ist L3 ein Partikel-Demo — davon haben wir bereits mehrere.

Was ein Container kosten darf

Budgets sind Regeln, keine Hoffnungen. Sie stehen im Manifest-Schema und werden vom Build-Schritt geprüft. Die Zahlen hier sind Startwerte für den Prototyp — sie werden gemessen und dann festgeschrieben.

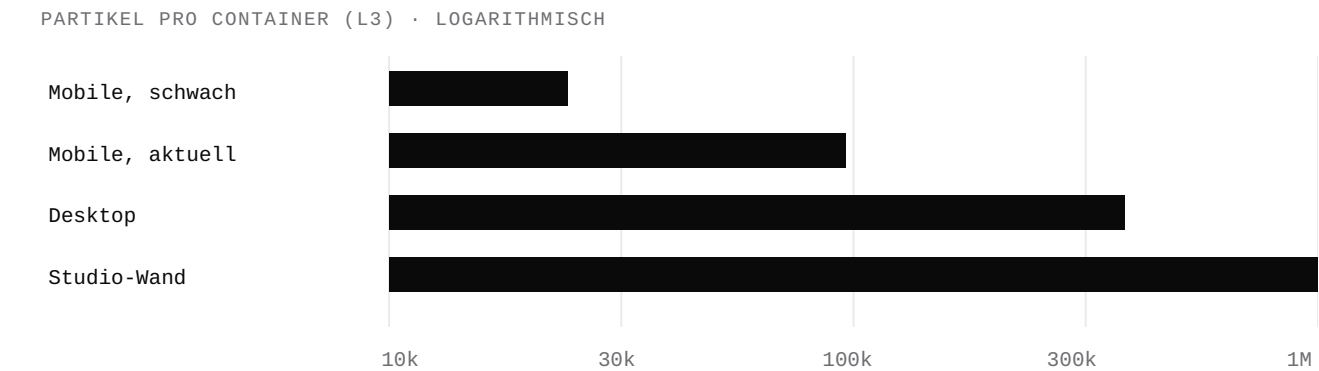


Abb. 9 — Partikelbudget je Gerätekategorie bei 60 Hz und einem Container im Viewport. Bei mehreren Containern teilen sie sich das Budget.

BUDGET	STARTWERT	BEGRÜNDUNG
Bytes pro Bild (L0, Viewport-Breite)	≤ 120 kB mobil · ≤ 400 kB desktop	AVIF bei s/w-Master komprimiert sehr gut; Farbe kommt aus der Regel, nicht aus der Datei.
Punktwolke (L3)	≤ 1,5 MB mobil · ≤ 6 MB desktop	Float16 xyz + Uint8 L ≈ 7 Byte/Punkt; 200 k Punkte ≈ 1,4 MB. Lazy nach Sichtbarkeit.
Frame-Zeit	≤ 8 ms pro Container	Halbes 60-Hz-Budget; die Seite selbst braucht den Rest.
Container gleichzeitig aktiv	≤ 2 mobil · ≤ 4 desktop	Ausserhalb des Viewports schläft ein Container vollständig (kein rAF, kein Audio-Node).
Audio	Opus 96 kbit/s, ≤ 2 MB	Lazy, erst nach Geste geladen.
Start bis erstes Bild	≤ 200 ms nach Sichtbarkeit	L0 muss sofort da sein; alles darüber darf nachladen.

Zwei Regeln sind wichtiger als alle Zahlen: Ein Container ausserhalb des Viewports verbraucht nichts, und ein Container im Viewport verbraucht nie mehr als sein Budget — notfalls, indem er eine Schicht tiefer fällt. Die Batterie eines Telefons ist Teil des Designs.

Das Messwerkzeug dafür ist Teil des Prototyps: ein Overlay, das pro Container Schicht, Dichte, Frame-Zeit und geladene Bytes anzeigt. Ohne dieses Overlay wird jede Diskussion über Performance zur Vermutung.

Das Manifest

Ein JSON-File pro Container. Es sagt, was im Ordner liegt, welche Regel gilt und wie weit der Container gehen darf. Es sagt nicht, wie etwas aussieht — das ist Sache der Regel. Entwurf, Version 0.

```
{
  "iac": "0.1",
  "id": "wk-0412",
  "title": "Urzelle · Hof · Prototyp 3",
  "rule": "@site/doc-quiet",          // Referenz, keine Definition
  "max_layer": 2,                     // Deckel: 0..3
  "master": { "mode": "luminance" },  // luminance | channel:R | file
  "sequence": [
    { "src": "01.avif", "alt": "Kubus im Hof, Nordseite" },
    { "src": "02.avif", "alt": "Detail Knoten 4" },
    { "src": "03.avif", "alt": "Kubus bei Nacht" }
  ],
  "timeline": { "driver": "scroll" },  // scroll | clock | track
  "audio": { "src": "track.opus", "role": "modulate" }, // optional
  "depth": { "source": "luminance" },  // luminance | map | audio | none
  "budget": { "class": "default" },    // Referenz auf Budget-Satz
  "build": {                           // vom Build-Skript gefüllt
    "variants": ["w640", "w1280", "w2560"],
    "points": ["p20k.bin", "p80k.bin", "p320k.bin"],
    "poster": "poster.avif"
  }
}
```

Vier Felder sind Pflicht

id, rule, master, sequence. Alles andere hat einen Standardwert aus der Regel. Ein Manifest mit vier Zeilen ist gültig und zeigt ein stilles s/w-Bild.

Der Alt-Text ist Pflicht

Pro Bild. Nicht als Zugänglichkeits-Feigenblatt, sondern weil L0 ohne JavaScript der einzige Zustand ist, den Crawler und Reader sehen. Was dort steht, ist das Bild.

Referenzen mit @

Alles, was mit @site/ beginnt, lebt in der zentralen Regeldatei. Das Manifest zeigt nur. So wird das Hot-Prinzip zur Konvention: Es gibt keinen Ort im Container, an dem eine Farbe oder eine Bewegung hart steht.

Einbettung und Ordnerregel

Ein Element in der Seite

Der Container ist ein Custom Element. Es lädt das Manifest, misst das Gerät, wählt die Schicht und rendert. Ohne JavaScript bleibt der Inhalt des Elements — ein `<picture>` mit dem Poster — sichtbar. Das ist L0 ohne Laufzeit.

Ordnerregel

Ein Container = ein Ordner = eine ID. Die ID folgt der Studio-Nomenklatur. Quellen liegen im Ordnerstamm, abgeleitete Daten unter `_build/`. Nichts unter `_build/` wird von Hand berührt; der Build-Schritt darf es jederzeit löschen und neu erzeugen.

Die Seite setzt den Rahmen

Breite, Seitenverhältnis und Position bestimmt die Seite wie bei jedem Bild. Der Container füllt den Rahmen. Er darf den Rahmen nie verlassen, auch nicht in L3; Tiefe entsteht innerhalb des Rahmens.

```
<iac-container src="/wk/wk-0412/manifest.json"
  max-layer="2"
  style="aspect-ratio: 3/2; width: 100%">
  <picture>
    <source srcset="/wk/wk-0412/_build/poster.avif" type="image/avif">
    
  </picture>
</iac-container>
```

```
wk-0412/
├─ manifest.json
├─ 01.avif 02.avif 03.avif      Originale (nie direkt angezeigt)
├─ track.opus                  optional
├─ _build/                     generiert, löscher
├─   master/ 01.avif ...       s/w-Master, 3 Breiten
├─   c2/      01.avif ...      Projektion pro Konzept (L1 vorberechnet)
├─   points/  p20k.bin p80k.bin Punktwolken (L3)
├─   poster.avif poster.webp   Fallback
```

EINWAND

Vorberechnete Projektionen pro Konzept (`_build/c2/`) vervielfachen den Speicherbedarf mit der Zahl der Konzepte. Bei fünf Konzepten und 300 Bildern in drei Breiten sind das 4500 Dateien. Das ist beherrschbar, aber nur mit einem Build-Schritt, der inkrementell arbeitet und nur erzeugt, was Manifeste tatsächlich referenzieren. Alternativ: Projektion ausschliesslich live per Filter und die Vorbereitung erst, wenn die Messung zeigt, dass es nötig ist. Der Prototyp soll beides messen.

Das Hot-Prinzip

Die Regel ist eine Datei an einem Ort der Website. Container referenzieren sie; Gruppen und Instanzen dürfen einzelne Werte überschreiben, aber nicht neu erfinden. Eine Änderung an der Regel wirkt sofort überall — deshalb braucht sie eine Vorschau und eine Versionsnummer.

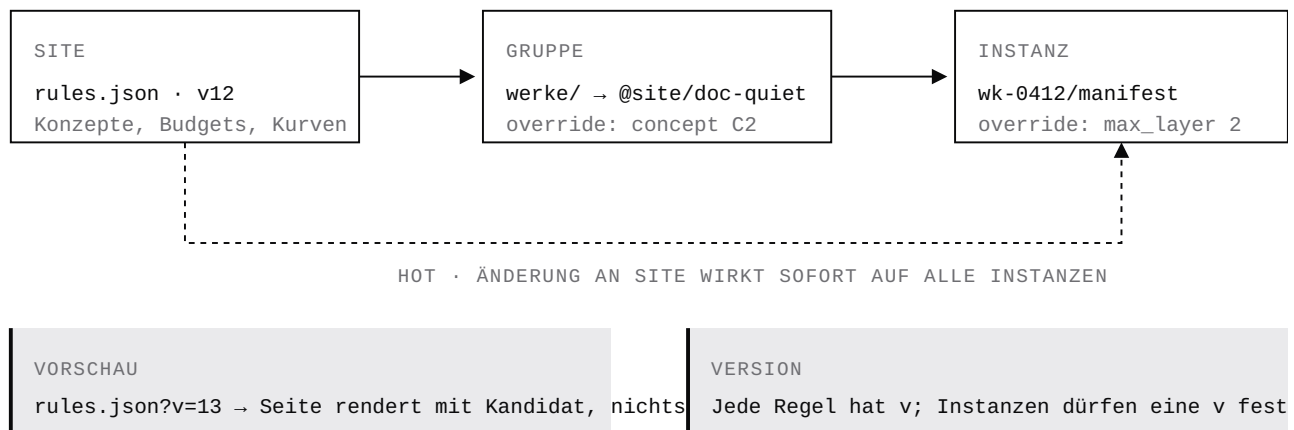


Abb. 10 – Drei Ebenen der Regel. Der gestrichelte Pfad ist das Hot-Prinzip; die beiden grauen Felder sind das, was es sicher macht.

Was in der Regel steht

Benannte Farbkonzepte (C1 ... Cn, je 3–6 Farben), Quantisierungsstufen, Verhaltenskurven pro Zeitlinie (als Keyframes mit Easing), Dichtestufen pro Geräteklasse, Budgets. Alles, was mehr als einen Container betrifft.

Was eine Instanz darf

Einen Wert aus der Regel überschreiben, nie einen neuen definieren. max_layer senken, nie heben. Ein Konzept wählen, keines erfinden. Das ist absichtlich eng: Freiheit lebt in der Regel, nicht in der Instanz.

Die Vorschau

Eine Regeländerung wird als Kandidat gespeichert und über einen Query-Parameter in die Seite geladen. Jan sieht die ganze Website mit der neuen Regel, bevor sie gilt. Erst das Publizieren erhöht die Version. Das ist die einzige Infrastruktur, die das Hot-Prinzip zwingend braucht.

Determinismus

Gleiche Quelle, gleiche Regel, gleicher Kontext → identischer Bildraum, auf jedem Gerät derselben Klasse. Zufall gibt es nur mit Saat, und die Saat steht in der Regel. Das ist der Beweis-Anspruch des Studios, übersetzt in Code.

Container vs. MP4

Die Frage ist berechtigt: Eine vorab gebackene Slideshow als MP4 ist billig, universell und ruckelt nie. Der Container ist offen, editierbar und hot. Beides hat seinen Platz — aber nur einer ist die Quelle.

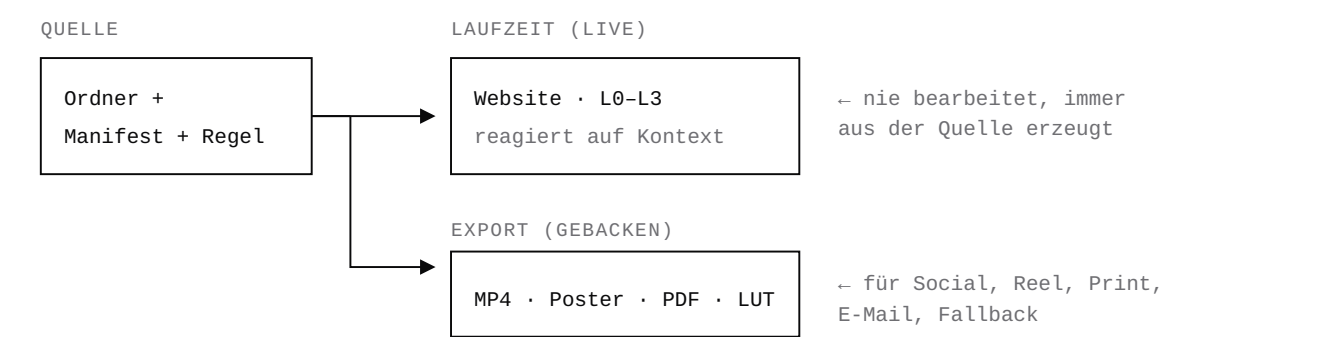


Abb. 11 – Der Container ist die Quelle; MP4 ist ein Export. Beide kommen aus demselben Ordner, derselben Regel.

	CONTAINER (LIVE)	MP4 (GEBACKEN)
Reagiert auf Scroll, Viewport, Geste	ja	nein — läuft linear
Editierbar (Bild hinzufügen, Regel ändern)	ja, hot	nein — neu rendern
Farbkonsistenz mit der Seite	aus derselben Regel	eingefroren zum Renderzeitpunkt
Bytes pro Sekunde Sichtbarkeit	klein (Bilder + Code)	gross (Video), aber sehr gut gecacht
Läuft ohne JavaScript	nur L0	ja
Rechenlast beim Betrachter	Schicht-abhängig	minimal (Hardware-Decoder)
Schärfe, Korn, Pixelästhetik	exakt, auflösungsunabhängig	Kompressionsartefakte, besonders bei Korn
Aufwand pro neuem Werk	Ordner anlegen	Ordner anlegen + rendern + hochladen

Empfehlung. Der Container ist die Quelle, immer. MP4 ist eine Exportfunktion des Build-Schritts — aus demselben Manifest, derselben Regel, mit fixierter Zeitlinie. Er wird dort eingesetzt, wo keine Laufzeit möglich ist: Social Media, E-Mail, LED-Wand ohne Browser, und als letzter Fallback, wenn L2 nicht erreichbar ist und die Sequenz trotzdem gezeigt werden soll.

Das letzte Argument wiegt am schwersten: Die pixelierte, körnige Ästhetik, die gut druckt, ist genau das, was Videocodecs am schlechtesten komprimieren. Ein MP4 dieser Ästhetik ist entweder riesig oder matschig. Der Container zeigt sie exakt — bei einem Bruchteil der Bytes.

Worauf wir aufbauen

Keines dieser Werkzeuge ist Images as Code. Jedes löst einen Teil. Die Liste ist bewusst kurz und sagt pro Zeile, was wir nehmen und was nicht.

WERKZEUG / KONZEPT	WAS ES IST	NEHMEN / NICHT NEHMEN
CSS Scroll-driven Animations	Deklarative Bindung von Keyframes an Scroll-Position, ohne Listener, auf dem Compositor-Thread.	Nehmen als Primärpfad für L2. Nicht nehmen: als einzigen Pfad (Firefox).
SVG feComponentTransfer	Tabellenbasierte Kanaltransformation = LUT im Browser.	Nehmen für L1-Übergänge. Nicht nehmen: für alle Bilder gleichzeitig.
three.js Points / Instancing	Standard-Partikelrendering in WebGL2, in unseren Prototypen bereits eingesetzt.	Nehmen als Renderer-Option für L3. Nicht nehmen: three.js als Abhängigkeit des Formats.
Gaussian Splatting (Web-Viewer)	Punktprimitive mit Ausdehnung und Deckung; sehr gute Darstellung, grosse Dateien.	Nehmen als Vorbild für das Punktwolken-Binärformat. Nicht nehmen: als Bildformat der Website.
Lottie / Rive	Vektoranimation als Datensatz, Laufzeit im Browser, editierbar.	Nehmen als Beleg, dass «Animation als Daten» im Web trägt. Nicht nehmen: sie sind vektorbasiert, nicht bildbasiert.
Apple Spatial Photos / Tiefenkarten	Bild + Tiefe als Paar; Parallaxe aus einem Foto.	Nehmen als Tiefenquelle (<code>depth.source: map</code>). Nicht nehmen: das Format.
Color-Mapping-Prototyp (Studio, Sept. 26)	Konzept-Compiler s/w → Farbe mit drei Exportzielen.	Nehmen vollständig — er wird zum L1-Teil des Build-Schritts.
Web-3D-Export (Studio, Okt. 26)	Renderer für SpaceMusic-Objekte mit 500 k Instanzen, navigierbar.	Nehmen als L3-Renderer-Kandidat. Nicht nehmen: Raytracing-Anspruch für Website-Bilder.

Die Lücke, die bleibt, ist genau die These: Niemand verbindet Quelle, Regel und Verhalten zu einem Datensatz mit Fallback-Leiter. Es gibt Bildformate, Animationsformate und Partikel-Engines; es gibt kein Format, das sagt «dieses Bild verhält sich nach Regel X, maximal bis Schicht Y, und sieht ohne alles so aus».

Zugleich die Warnung: Wir bauen kein Rendering und kein Format, sondern eine Konvention und eine schlanke Laufzeit, die bestehende Teile orchestriert. Je weniger eigener Code, desto länger lebt das Werkzeug.

Nicht zu trocken, nicht zu laut

Die Skizze formuliert die Sorge «become part of our aesthetic without making it too dry». Die Gegensorge ist ebenso real: Ein Werkzeug, das jedes Bild beleben kann, belebt schnell jedes Bild. Das Heft legt deshalb den Nullzustand als Stille fest und macht Bewegung zu einer begründeten Ausnahme.

Sechs Regeln

1 — Stille ist Standard. Ein Container ohne Regel-Override zeigt ein s/w-Bild und tut nichts. Verhalten wird zugeschaltet, nicht abgeschaltet.

2 — Ein Ereignis pro Viewport. Höchstens ein Container pro Bildschirmhöhe darf sich bewegen. Die Laufzeit zählt und deckelt; der zweite bleibt still.

3 — Bewegung hat einen Grund. Sie zeigt etwas, das stehend nicht sichtbar wäre: eine Folge, eine Tiefe, einen Übergang. Dekorative Bewegung ist in der Regel nicht vorgesehen; sie müsste begründet überschrieben werden.

4 — Farbe kommt spät. Im Default-Verhalten ist Farbe das Letzte, was einfließt, nicht das Erste. Das entspricht der Doktrin: erst Helligkeit, dann Farbe.

5 — Nichts wiederholt sich. Ein Scroll-Verhalten läuft einmal und bleibt im Endzustand. Zurückscrollen spielt nicht rückwärts; es zeigt den Endzustand. Loops gibt es nur mit Uhr-Zeitlinie, und die ist Reel-Situationen vorbehalten.

6 — Das Werkzeug misst sich selbst. Der Build-Schritt gibt pro Seite ein «Bewegungsmass» aus: wie viele Container, wie viele Sekunden Verhalten, wie viele davon mit Farbe. Eine Zahl, die man ansehen kann, bevor man eine Seite publiziert.

EINWAND

Regel 5 wird kontrovers sein. Reversibles Scroll-Verhalten ist der Standard in der Branche und fühlt sich «richtig» an. Die Begründung für Einmaligkeit ist das Mass: Ein Bild, das sich bei jedem Zurückscrollen wieder entfärbt, wird zum Spielzeug. Aber das ist eine ästhetische Setzung, keine technische Notwendigkeit. Der Prototyp soll beide Varianten können, damit die Entscheidung am Resultat fällt und nicht am Prinzip.

Warum das ins Research gehört

Weil es die teuerste Entscheidung des Projekts ist und die am wenigsten technische. Eine Website aus lebenden Bildern unterscheidet sich von einer Website mit drei lebenden Bildern nicht in der Technik, sondern in der Haltung. Die Regel ist der Ort, an dem Haltung Code wird.

Artefakt und Pixelästhetik

Der Auslöser des Themas war ein Druckproblem. Es ist wichtig, ihn als Druckproblem zu behandeln — und die Ästhetik, die daraus entstand, getrennt davon als Chance.

Das Artefakt

Die Farbprobe (Zach-Bush-Seite, Nebelgrafik) zeigt körnige, fleckige Haftungslücken in dunklen Flächen und an Kanten, die Pixelprobe (Labyrinth, Gestein) druckt sauber. Das ist typisch für Tonerhaftung bei schwerem Papier: Der Fixierer ist für das eingestellte Papergewicht zu kalt oder zu schnell. Reihenfolge der Prüfung:

- Treiber: Papiertyp auf «Schwer / 161–220 g» oder die nächsthöhere Stufe stellen, nicht «Normal». Die Einstellung muss zur Kassette passen, aus der gezogen wird.
- Eine Probe mit derselben Datei auf 80-g-Papier. Verschwindet das Artefakt, ist es die Fixierung, nicht der Treiber.
- Eine Probe über den PostScript- statt den PCL-Pfad (oder umgekehrt). Unterschiede deuten auf Rasterung im Treiber.
- Erst danach: Tonerkassette, Transferband, Luftfeuchte des Papiers.

Dass es an beiden Standorten auftritt, spricht gegen ein Gerätedefekt und für eine Einstellung oder das Papier.

Die Chance

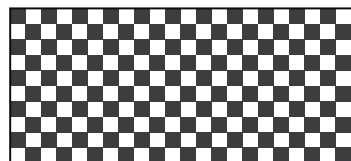
Die Pixelprobe druckt nicht sauber, weil sie «grob» ist, sondern weil sie keine grossen gleichmässigen Flächen hat: Das Korn versteckt Haftungslücken. Eine quantisierte, geditherte Projektion (L1 mit kleinem n und Dither) ist deshalb druckrobust auf diesem Drucker — unabhängig davon, ob das Artefakt behoben wird.

Daraus folgt ein Export im Build-Schritt: `print` — dieselbe Regel, dieselbe Projektion, aber mit Rasterung auf die Druckauflösung (600 dpi, Korngrösse 2–4 Punkte) und in reinem Schwarz oder im Konzept mit vier Farben. Das Heft, das du in der Hand hältst, und die Website zeigen dann dasselbe Bild — nicht ähnlich, sondern aus derselben Quelle und Regel.

FLÄCHE · ARTEFAKT SICHTBAR



KORN · ARTEFAKT VERSTECKT



Gleiche mittlere Dichte,
gleiche Haftungslücken —
nur rechts sind sie unsichtbar.

Abb. 12 — Warum Korn druckt. Das Artefakt ist in beiden Flächen gleich; im Korn ist es nicht vom Muster zu unterscheiden.

Scope und Abnahme

Der erste Prototyp beweist das Manifest, die Regel und das Hot-Prinzip auf L0–L2. L3 kommt im zweiten Schritt, wenn die Basis gemessen ist. Eine HTML-Datei, ein Ordner, eine Regeldatei — hochgeladen auf prototypes.origin-infinite.com und auf drei Geräten geprüft.

Im Scope (P1)

- Custom Element <iac-container> mit Manifest-Loader und Fähigkeitsmessung
- L0: s/w-Master aus 1–3 Bildern, <picture>-Fallback ohne JS
- L1: zwei Farbkonzepte aus der Regeldatei, Stufenzahl n einstellbar, harte und interpolierte Variante nebeneinander
- L2: Scroll-Zeitlinie (CSS primär, JS-Fallback), vier Kurven: Opacity, Unschärfe, Farbanteil, Sequenz; einmalig und reversibel umschaltbar
- Regel-Vorschau per Query-Parameter; Regelwechsel ohne Neuladen des Containers
- Mess-Overlay: Schicht, Frame-Zeit, Bytes, Bewegungsmass
- Build-Skript (Python): Master, Varianten, Poster, print-Export (600 dpi, gedithert)

Nicht im Scope (P2)

- L3 Voxel-Akteur, Punktwolken-Build, Audio-Modulation
- Uhr- und Track-Zeitlinie
- MP4-Export
- Vorberechnete Projektionen (erst nach Messung)

Abnahme

- Drei Container auf einer Testseite, davon zwei mit Verhalten; auf einem iPhone ab 2021, einem schwachen Android und einem Desktop
- Ohne JavaScript: drei lesbare s/w-Bilder mit Alt-Text
- Regel ändern → alle drei ändern sich, ohne Neuladen; Vorschau zeigt Kandidat
- Frame-Zeit ≤ 8 ms pro Container auf dem iPhone, gemessen im Overlay
- Ein A4-Ausdruck des print-Exports auf dem OKI C844, 160 g, ohne Haftungsartefakt sichtbar
- Jans Urteil nach einer Woche Nutzung: zu trocken, richtig, zu laut — pro Kurve

GERÄT	L0	L1	L2 CSS	L2 JS	REDUCED MOTION	OHNE JS
iPhone, iOS 26, Safari	•	•	•	—	•	•
Android low-end, Chrome	•	•	•	—	•	•
Desktop, Firefox	•	•	—	•	•	•
Desktop, Chrome / Safari	•	•	•	—	•	•

Testmatrix P1. Punkt = zu prüfen, Strich = nicht anwendbar. Jede Zelle bekommt bei der Abnahme ein Resultat und eine Frame-Zeit.

Risiken und Glossar

Offene Fragen

Interpolation in L1. Harte Kante oder Übergang zwischen Konzeptfarben — seit September offen; fällt am Prototyp.

Einmalig oder reversibel in L2. Siehe Kap. 15. Fällt am Prototyp.

Renderer in L3. Three.js-Basis oder eigener Renderer aus dem Web-3D-Export. Wird erst in P2 entschieden; das Format darf es nicht vorwegnehmen.

Vorberechnung in L1. Live-Filter oder gebaute Varianten — hängt von der Messung ab.

Ort der Regeldatei. Statisches File im Repo oder kleines CMS-Feld. Für P1 statisch; die Vorschau funktioniert mit beidem.

Tiefenquelle. Helligkeit ist der einfachste Fall, aber nicht immer die beste Tiefe. Tiefenkarten aus Foto-Metadaten (Spatial) oder aus einem Modell — P2-Thema.

Glossar

Container	Ordner + Manifest. Die Einheit, die auf der Website ein Bild ist.
Manifest	JSON-Datei, die Inhalt, Regelreferenz und Grenzen des Containers beschreibt.
Regel / DNA	Zentrale, versionierte Übersetzungsvorschrift von Quelle zu Bildraum.
Master	Das s/w-Bild, aus dem alles abgeleitet wird. Nie das Original.
Projektion	Zuweisung von Farbe zu Helligkeit nach einem benannten Konzept (LUT).
Zeitlinie	Treiber von t: Scroll, Uhr oder Track.
Akteur / Voxel	Partikel mit Identität über die Zeit; Zustand in Raum, Grösse, Farbe.
Schicht L0–L3	Master · Projektion · Zeit · Raum. Jede fällt auf die darunter zurück.
Hot	Verbindung von Container und Regel zur Laufzeit; Änderung wirkt sofort.
Bewegungsmass	Kennzahl pro Seite: Container, Sekunden Verhalten, Anteil mit Farbe.

Risiken

Zu viel Leben. Das grösste Risiko und das am wenigsten technische. Gegenmittel: Kap. 15, Bewegungsmass, Jans Wochenurteil.

Fragilität durch Hot. Eine Regeländerung bricht 300 Bilder gleichzeitig. Gegenmittel: Vorschau, Version, Festnageln pro Instanz als Ausnahme.

Vier Codepfade. Testaufwand wächst mit jeder Schicht. Gegenmittel: Testmatrix als Teil des Deliverables; L3 erst nach L0–L2.

Speicherbedarf des Builds. Konzepte × Bilder × Breiten. Gegenmittel: inkrementeller Build, nur referenzierte Kombinationen.

Die Analogie zu SpaceMusic. Es ist verlockend, den Website-Container zum vollen SM-Akteur zu machen. Die Website braucht 5 % davon. Gegenmittel: max_layer und das Mass.



STUDIO HANSEN
RESEARCH · IMAGES AS CODE
FASSUNG F1 · 15. OKTOBER 2026

Ein Bild ist nicht, was gespeichert ist. Ein Bild ist, was eine Regel aus einer Quelle macht — jedes Mal neu, jedes Mal gleich.

Nächster Schritt: Minimal-Prototyp P1 (L0–L2) mit Regel-Vorschau und Mess-Overlay, dann P2 (L3) auf der gemessenen Basis.