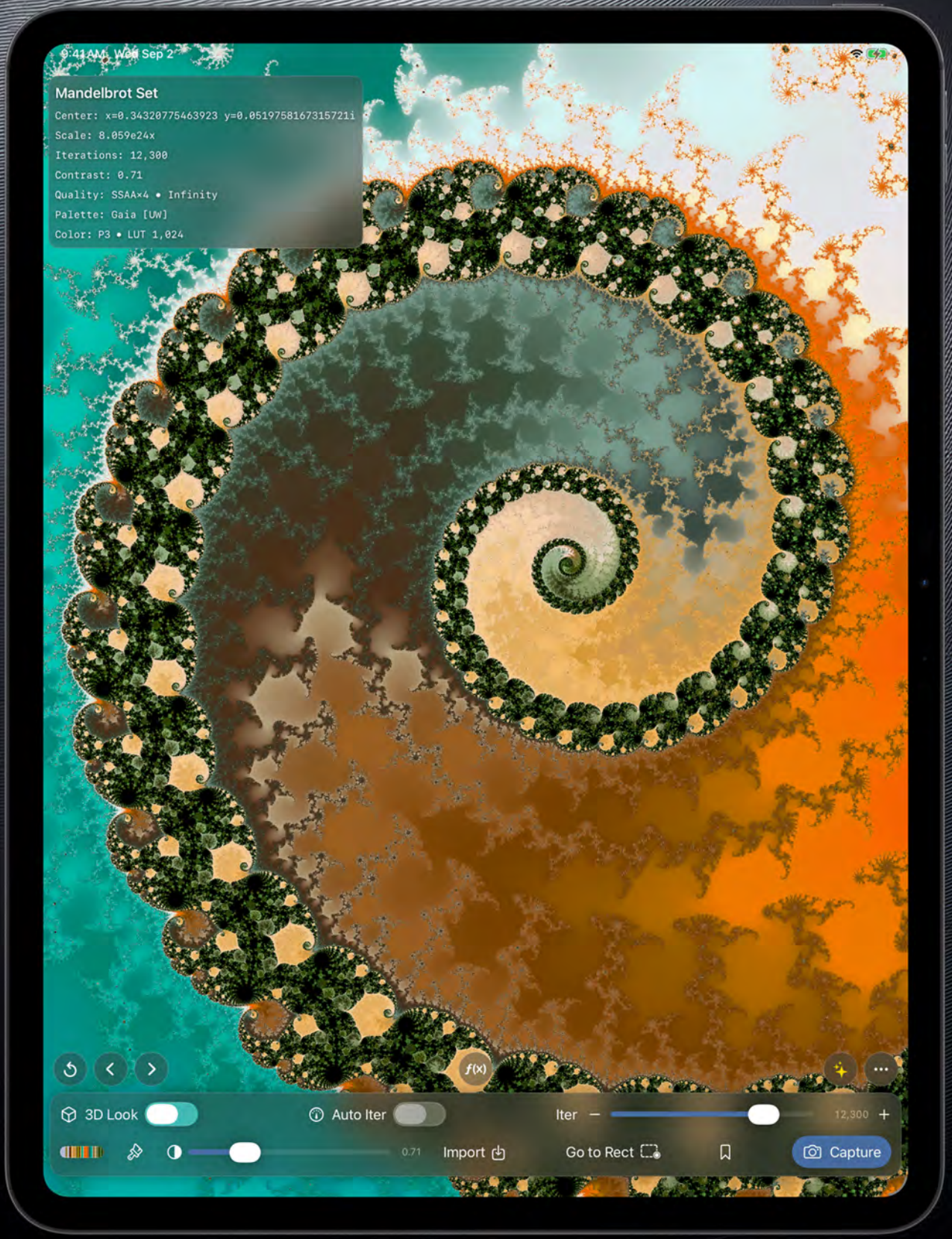

Mandelbrot Metal 3

Technical White Paper

Mandelbrot Metal 3 Technical White Paper

Covers Version 3.0 Build 570

Preface

I launched Mandelbrot Metal in October 2025 as both a visual exploration of the Mandelbrot set and a computer science project in numerical methods and high-performance GPU rendering. Since then, I have developed new algorithms, precision techniques, and rendering code to improve rendering speed and extend the zoom range.

The result is what I call the Infinity Engine: a rendering architecture that extends the app's zoom range while preserving responsive navigation and visual quality. It underpins Mandelbrot Metal 3 and its Free + Pro model.

Executive Summary

Mandelbrot Metal v3 is a GPU-first fractal renderer for iPhone and iPad that evaluates the quadratic map $z_{n+1} = {z_n}^2 + c$. Version 3 extends its fused Metal pipeline with the Infinity Engine. The CPU builds a double-double reference orbit and uploads its components as four-Float expansions. The GPU uses that orbit to evaluate each pixel through a hybrid Float/quad-single (QS) perturbation recurrence. Within the supported zoom range, Infinity renders tiles from the center outward, progressively replacing a preview. Infinity retains the completed frame for navigation. The established CPU renderer handles fallback when Infinity cannot be used.

This revision documents Version 3.0 build 570, including the September 8 launch and review-prompt changes, with source verified on September 9, 2026. Section 15.4 reports physical-device benchmarks from build 548; those results do not measure build 570. The current release combines adaptive rendering, compensated navigation, progressive presentation, complete-image capture, responsive controls, palettes, 3D Look, bookmarks, exploration history, 16-language localization, and StoreKit 2 access management.

Version 3.0 offers Free and Pro access, with permanent Legacy Pro access for verified 2.x owners. It raises the v2.2 10^{16} scale boundary to a validated Mandelbrot ceiling of 10^{25}. The build 548 benchmarks show the largest gains over v2.2.3 in deep scenes. Gestures, Coordinate Jump, bookmarks, history, persistence, HUD formatting, and renderer input share this enforced limit. Within the range, the renderer selects among GPU FP32/DS/legacy dd2, Metal Float/QS perturbation, and the CPU DD safety path based on representability and command validity.

This is adaptive finite-precision rendering. Experimental support for coordinates and reference orbits beyond the release ceiling is outside the public v3.0 path. Extending the supported range requires separate checks of image correctness, resource use, and physical-device behavior.

Version 3 release scope

Release	Engineering boundary	Status represented here
3.0	Infinity v1, Metal QS perturbation, fallback circuit, export requalification, Free + Pro, Legacy Pro migration	Implemented in build 570; historical benchmarks and remaining release checks are distinguished in Section 15
3.1	Reference-orbit acceleration, wider device qualification, and deep-zoom diagnostics	Roadmap scope; extends the v3.0 engine
3.2	Deterministic creation system	Roadmap scope only
3.3	Breadth and accessibility expansion	Roadmap scope only
3.4	Reliability and platform qualification	Roadmap scope only

Table 1 — Version 3 release plan and engineering boundaries.

1. System Architecture

1.1 Dataflow

The v3 renderer combines orbit evaluation and color generation in one compute kernel. Swift owns application state, drawable geometry, adaptive-quality policy, precision-path selection, reference-orbit construction, export orchestration, and fallback decisions. A Metal compute kernel maps pixels to the complex plane, evaluates orbit dynamics, computes smooth escape metrics, samples the active palette, applies optional lighting, and writes one BGRA8Unorm pixel per output thread.

Stage	Representation	Responsibility
Application model	Compensated center and exact strings	Viewport, Julia parameter, palette identity, history, bookmarks, entitlement
Frame policy	Swift + Metal objects	Iteration target, SSAA tier, precision label, Infinity eligibility, resources
Reference orbit	CPU DD → float4 limbs	High-precision Mandelbrot anchor orbit uploaded to a shared MTLBuffer
Per-pixel compute	FP32, DS, legacy dd2, or QS	Mapping, iteration, perturbation, smoothing, palette, lighting
Presentation/export	BGRA8Unorm texture	Initialized progressive presentation or complete tiled PNG export

Table 2 — Fused renderer dataflow and responsibility split.

The kernel is intentionally fused. It has no persistent escape-metric texture or palette-only recoloring pass. A change to palette, contrast, iteration, viewport, precision, or lighting therefore schedules a complete compute render. This costs orbit work on color-only edits but avoids a second large intermediate buffer and keeps the rendering pipeline compact.

1.2 Render-path selection

1. Start with the live or export viewport expressed as the stored center, drawable dimensions, and per-pixel steps.
2. Use the established GPU FP32/DS/legacy dd2 path at ordinary scale. For Mandelbrot, enable perturbation automatically and construct or refresh the DD reference orbit once the deep preparation threshold is crossed.

3. At the legacy CPU threshold, prefer Infinity only when the reference buffer exists, the split pixel step remains representable, and the runtime failure circuit permits Infinity. Each pixel first tries a reference-relative Float perturbation and restarts in QS when a recovery test detects a problem.
4. Within the supported zoom range, render Infinity tiles from the center outward in bounded Metal command buffers. Initialize the complete destination with a valid preview, reveal completed regions progressively, and retain the finished frame for subsequent gestures. Keep a valid completed frame even when it exceeds the 250 ms diagnostic target.
5. If an Infinity command fails or yields an unusable duration, explicitly activate the complete v2.2 CPU deep renderer. For high-resolution capture, recompute the output mapping and repeat eligibility using the export pixel step.

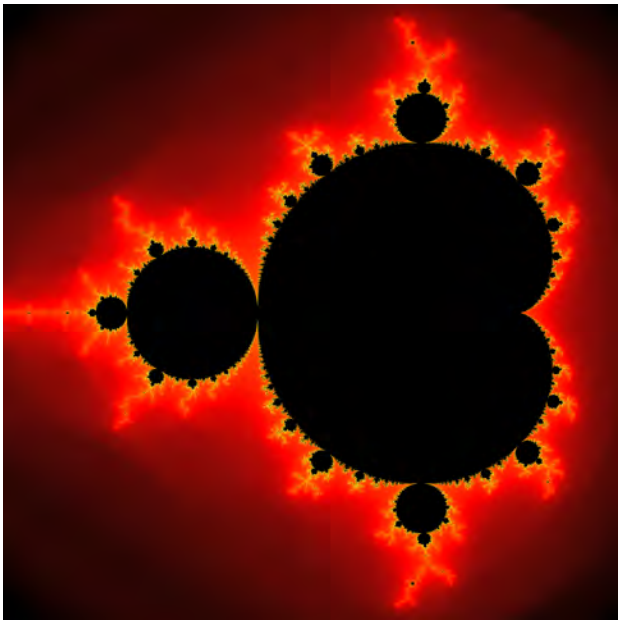

Figure 1 — Full Mandelbrot set render; the black interior and colored escape field are products of distinct classification paths.

2. Mathematical Model

2.1 Complex dynamics

For a complex parameter $c = c_r + ic_i$, where $c_r = \mathrm{Re}(c)$ and $c_i = \mathrm{Im}(c)$, define the quadratic map $f_c(z) = z^2 + c$. Mandelbrot mode fixes $z_0 = 0$ and assigns a different c to every sample. Julia mode fixes c and assigns a different z_0 to every sample. The two modes therefore evaluate the same recurrence but exchange the role of the sampled coordinate. The index n counts recurrence steps from zero; i is the imaginary unit, whose square is −1.

$$z_{n+1} = {z_n}^2 + c$$

$$M = \{c \in \mathbb{C}\text{:the critical orbit} z_0 = 0 \text{ remains bounded}\}$$

$$K(c) = \{z_0 \in \mathbb{C}\text{:the orbit remains bounded}\}, J(c) = \partial K(c)$$

Here, $\mathbb{C}$ denotes the complex numbers, M the Mandelbrot set, $K(c)$ the filled Julia set for fixed c, and $J(c)$ its boundary. The symbols $\in$ and ∂ mean “belongs to” and “boundary,” respectively; vertical bars denote complex magnitude.

Writing $z_n = x_n + iy_n$ and $c = a + ib$ gives the real-valued recurrence used by every direct kernel:

$$x_{n+1} = {x_n}^2 - {y_n}^2 + a, y_{n+1} = 2x_n y_n + b$$

The real and imaginary components of the orbit value are x_n and y_n; a and b are the corresponding components of the parameter c.

The Mandelbrot set is connected, compact, and contained within the disk $|c| \leq 2$. Its boundary is not exactly globally self-similar, although it contains infinitely many embedded and near-copies. The renderer visualizes escape behavior near that boundary; it does not alter the underlying set when color, contrast, sampling, or lighting changes.

Learn more about fractals and the Mandelbrot and Julia sets, as well as complex numbers.

2.2 Escape proof and finite budgets

For Mandelbrot mode, the mathematical escape condition is $r = |z| > 2$. The kernel evaluates the squared magnitude $r^2 = |z|^2 = x^2 + y^2$. Because r is nonnegative, $r > 2$ is equivalent to $r^2 > 2^2 = 4$. This avoids computing $r = \sqrt{x^2 + y^2}$ on every iteration: the square root would not change the escape decision and would add unnecessary work and rounding. Comparing r^2 with 2 would not test radius 2; it would test radius $\sqrt{2}$ and could mark samples as escaped too early. For Mandelbrot mode, $|c| \leq 2$ for all candidates not already known to escape on the first step. Once $|z_n| > 2$, the reverse triangle inequality gives $|z_{n+1}| \geq |z_n|^2 - |c| > |z_n|$, so divergence follows.

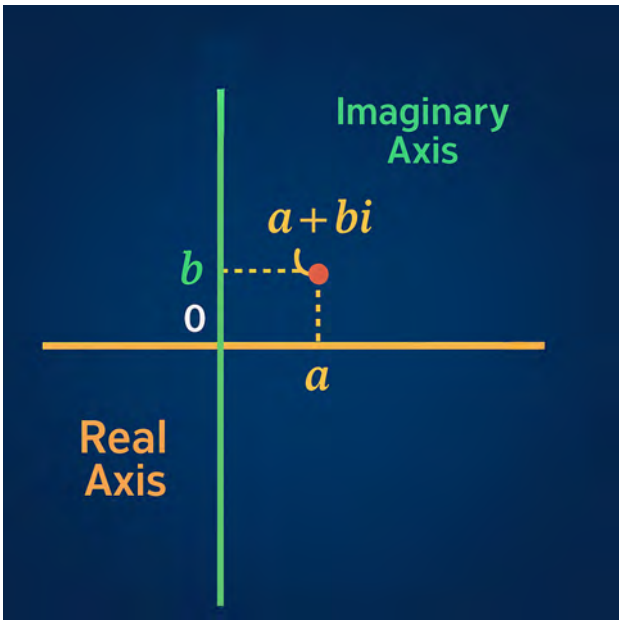

Figure 2 — Conventional complex-plane axes. The display mapping retains the same coordinates but increases the device y downward.

For a general Julia parameter, a sufficient radius R satisfies $R^2 - R - |c| > 0$. The implementation uses $R = 2$ for Julia rendering as well; that is sufficient for the intended Mandelbrot-derived domain $|c| \leq 2$ but does not prove the result for arbitrary Julia constants outside that domain.

The main-cardioid and period-2-bulb formulas identify analytic Mandelbrot interior regions in exact arithmetic. Their implementation still uses finite-precision operations, so a numerical test near a boundary is not a general proof of exact membership. Julia mode disables these shortcuts. For samples not classified by these analytic shortcuts, a black result means that no escape was detected within the iteration budget, using the selected arithmetic path and coordinate representation. It does not establish exact membership in the set.

3. Coordinate Mapping

3.1 Drawable coordinates

Let the drawable contain $W \times H$ pixels, let $C = x_0 + iy_0$ be the complex coordinate at its center, and let S be the effective drawable pixels per complex-plane unit. An integer pixel coordinate (x, y) maps to the complex plane as shown below. Here, x and y are zero-based pixel indices; x_0 and y_0 are the center's real and imaginary coordinates. Uppercase C denotes the viewport center, while $c(x, y)$ denotes a mapped sample.

$$c(x,y) = [x_0 + \frac{x - \frac{W-1}{2}}{S}] + i[y_0 + \frac{y - \frac{H-1}{2}}{S}]$$

The Metal kernel uses a positive y step, so increasing device y—downward on screen—increases the imaginary coordinate. This is a vertical reflection of the conventional diagram, where $\mathrm{Im}(c)$ increases upward. Gestures, tap-to-Julia conversion, Coordinate Jump, history, bookmarks, and export must all use the same convention, or the restored composition will drift or reflect. See Figure 2.

3.2 Center preservation and scale safety

Live origin and step are computed in Double on the CPU and also split into Float high/low pairs for Metal. A dynamic baseline is rebased when the scale drifts by a factor of 256 in either direction, reducing coordinate-delta size. The shared ViewportScalePolicy clamps model values to $[10^{-9}, 10^{25}]$ navigation

points per complex-plane unit and sanitizes NaN and infinity before gesture, Coordinate Jump, bookmark, persistence, or history restoration.

The 10^{25} ceiling is the validated Version 3.0 limit; it is not inferred from the number of limbs. It prevents the visible scale from advancing beyond the qualified navigation and renderer path. Actual detail within the range is still bounded by the active coordinate expansion, pixel-step representability, reference-orbit fidelity, iteration budget, and bailout/recovery decisions. A future 10^{100} extension requires a new coordinate and reference-orbit backend, with separate device validation.

The navigation ceiling is expressed in UIKit points per complex-plane unit. The Metal drawable uses physical pixels; its effective pixel scale is the navigation scale multiplied by the screen scale. The benchmark table explicitly reports drawable pixels per unit. Convert between navigation points and drawable pixels before comparing these scales.

3.3 Export mapping

An offscreen export preserves the live center but recomputes the step and origin to match the target aspect ratio. If (s_x, s_y) is the live step and $W_e \times H_e$ is the export size, the renderer multiplies the live step by $\max\left(\frac{W_{\text{live}}}{W_e}, \frac{H_{\text{live}}}{H_e}\right)$. It then places the origin half an image extent from the preserved center. This avoids letterboxing and makes export geometry independent of the live texture dimensions. The live dimensions W_{live} and H_{live} and export dimensions W_{e} and H_{e} are all in pixels. The step components s_x and s_y are complex-plane units per pixel; the maximum ratio selects the factor needed to fill the export frame.

4. Escape-Time Coloring

4.1 Continuous escape coordinate

For an escaped sample, z is the terminal complex orbit value, |z| its magnitude, n the zero-based loop index, and $N_{\max}$ the configured iteration budget. The kernel computes a base-2 logarithmic correction before normalization:

$$\nu = \log_2\big(\log_2(|z|)\big), \mu = n + 1 - \text{clamp}(\nu, 0,1)$$

$$t_0 = \text{clamp}\left(\frac{\mu}{N_{\max}}, 0,1\right), t_1 = \text{clamp}\left(\frac{t_0 - 0.015}{0.97}, 0,1\right)$$

$$t = t_1^{\frac{1}{\text{contrast}}}$$

The correction ν (nu) produces the fractional iteration coordinate μ (mu). The values t_0, t_1, and t are successive coordinates in the range 0–1: normalized escape value, remapped value, and final palette position. The function clamp(v, 0, 1) limits v to that interval. The positive contrast setting controls the final exponent, and $\log_2$ denotes a base-2 logarithm.

The unclamped potential-based form is often written as $n + 1 - \log_2\big(\log_2(|z|)\big)$. Mandelbrot Metal's clamped correction, normalization window, and contrast exponent are implementation choices. They control visual continuity and slider response; they do not define the set's invariants.

4.2 Palette sampling

The selected palette determines which of four shader branches maps t to color: procedural HSV, Fire, Ocean, or a one-row texture lookup. Core Graphics rasterizes named and imported palettes into BGRA8 lookup-table (LUT) textures, typically 256 or 1024 texels wide. The Metal sampler is linear and clamps to the edge. "Exact LUT" changes how texels are generated from source stops; it does not change the shader sampler to nearest-neighbor filtering.

Under supersampling anti-aliasing (SSAA), every subpixel completes its own orbit and palette lookup before RGB values are averaged. Since the output and LUT are BGRA8Unorm rather than sRGB-decoded or floating-point render targets, the numerical average is over stored component values. The pipeline therefore does not claim scene-linear light, HDR output, or bit-identical cross-device color.

4.3 Lighting

3D Look treats the smooth escape metric as a height field. Four additional finite-difference samples, capped at 256 iterations, estimate a screen-space normal. Ambient, diffuse, and specular terms then modulate the already averaged color. Lighting is evaluated once per output pixel, not once per SSAA subpixel, and invalid results are sanitized before clamping to BGRA8.

The lighting creates a visual relief effect; it does not model a physical surface. The light direction, height scale, shininess, and specular strength are members of the render state; none changes orbit classification.

5. Swift–Metal Contract

5.1 Stable uniform ABI

MandelbrotUniforms exists in both Swift and Metal and must match field order, size, and padding exactly. The original field offsets remain stable. Build 570 uses the former eight-byte padding beside c0 for tilePixelOffset and appends Infinity mapping fields plus an experimental exponent-bearing suffix. The public path leaves the experimental range disabled. Setting infinityMode to zero permits the legacy path without reinterpreting the established prefix.

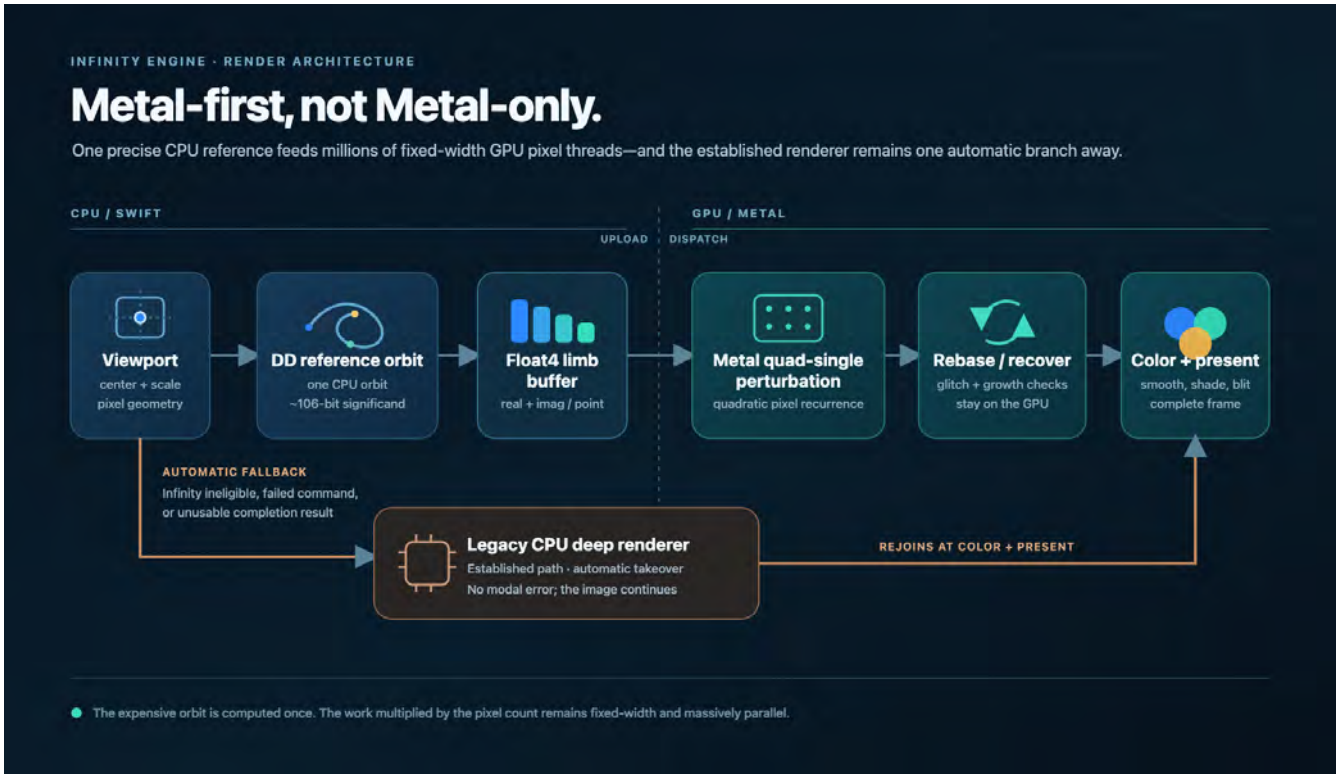

Figure 3 — The Infinity Engine's render architecture.

Field group	Representative fields	Purpose
Base mapping	origin, step, size, pixelStep	FP32 viewport and drawable/debug state
Quality/color	maxIt, subpixelSamples, palette, deepMode, contrast	Work limits and fused color policy
Split mapping	originHi/Lo, stepHi/Lo	Compensated coordinate mapping
Reference orbit	perturbation, refCount, c0	Legacy and Infinity orbit binding
Lighting	mode, strength, height, direction, shininess, specular	Finite-difference 3D Look
Fractal family	fractalMode, juliaC	Mandelbrot/Julia dispatch
Infinity suffix	infinityMode, referenceOriginHi/Lo	QS perturbation and anchor-relative mapping
Tile placement	tilePixelOffset	Center-out placement with stable prior offsets
Experimental suffix	mantissas, exponents, reference count	Isolated beyond the public 10^{25} range

Table 3 — Shared Swift–Metal uniform groups and responsibilities.

InfinityReferenceOrbitPoint is a separate buffer element with float4 real and float4 imag fields. Each four-limb expansion represents one real or imaginary CPU DD component. Keeping the limbs separate during upload prevents an immediate collapse back to a single Float.

5.2 Binding and fallback

The compute encoder binds uniforms at buffer index 0, reference-orbit data at index 1, reference descriptors at index 2, and diagnostics at index 3. The public rendering path binds valid dummy descriptor/diagnostic resources where needed. The BGRA8 output and palette textures remain at texture indices 0 and 1. If the selected LUT is unavailable, a 1×1 dummy makes the shader use HSV without changing the saved palette identity.

If perturbation is requested but the reference buffer is missing, Swift attempts to rebuild the orbit. Debug builds throttle warnings until the pending state persists. Until a valid reference is available, the renderer uses a path that can render the frame without it.

6. Precision Representations

6.1 The representation ladder

The label “deep precision” covers several materially different arithmetic systems. The path name alone does not describe the precision of every operation: bailout, smoothing, or resource transfer may reduce values to Float.

Path	Storage	Primary role	Important limit
GPU Float	1 × Float	Normal interactive orbit/color	Approximately 24-bit significand
GPU DS	2 × Float (hi + lo)	Compensated mapping and iteration	Selected tests reduce magnitude to Float
Legacy GPU dd2	Nested Float expansions	Historical deep label	Not equivalent to two IEEE Doubles
CPU DD	2 × Double (hi + lo)	Reference orbit and legacy deep tiles	About 106 significant bits in normal finite operation
Infinity QS	4 × Float limbs	Per-pixel perturbation and recovery in Metal	Step and classification diagnostics still include Float reductions

Table 4 — Precision representations, roles, and implementation limits.

CPU DD uses TwoSum, QuickTwoSum where its magnitude precondition holds, FMA residual products, and renormalization. A DD value $x = x_{\text{hi}} + x_{\text{lo}}$ carries about 106 significant bits—roughly 31–32 decimal digits—when operations remain finite and well-scaled. Infinity does not upload the two Doubles directly. It converts each DD real or imaginary component into four Float limbs suitable for Metal. The high component x_{hi} holds the leading value; x_{lo} retains the smaller residual.

6.2 Error-free transforms

$$\text{TwoSum}(a, b) \to (s, e) \text{ such that } a + b = s + e \text{ exactly, when the rounded sum is finite.}$$

$$\text{TwoProd}(a, b) \to (p, e), p = \text{round}(a \cdot b), e = \text{fma}(a, b, -p)$$

Here, a and b are floating-point operands, s is their rounded sum, p is their rounded product, and e is the retained rounding error. In this arithmetic context, round means rounding to the working floating-point format. The function fma multiplies its first two arguments and adds the third with one final rounding.

Addition combines high-part and low-part residuals before renormalization. Multiplication retains the leading product and FMA residual, then accumulates cross terms. Error-free-transform identities assume the relevant operations remain in their supported finite range; underflow, overflow, and an unmet QuickTwoSum magnitude precondition require separate treatment. The implementation guards non-finite results to prevent overflow from producing a NaN residual.

6.3 Quad-single (QS) expansion arithmetic

Quad-single (QS) is a software-emulated extended-precision representation built from four 32-bit Float limbs. It stores a value as the unevaluated sum $q = q_0 + q_1 + q_2 + q_3$, with the limbs ordered by decreasing magnitude so low-order information survives beyond ordinary FP32. Metal helpers implement two-sum, quick-two-sum, two-product, three-sum, five-term renormalization, expansion addition, subtraction, and multiplication. Complex multiplication is assembled from the real operations. The objective is to preserve enough low-order information for anchor-relative perturbations while keeping every pixel's recurrence on the GPU.

QS is not IEEE binary128 (often called quad precision) and does not provide a fixed hardware precision guarantee. Its effective accuracy depends on limb non-overlap, renormalization, cancellation, operation ordering, and the point at which the code converts the expansion to Float for an escape or glitch diagnostic. A single decimal-digit count would therefore misrepresent the accuracy of the complete rendering path.

7. Infinity Engine

7.1 Reference orbit

For Mandelbrot mode, choose a reference parameter c_0 near the viewport center and compute

$$Z_0 = 0, \qquad Z_{n+1} = {Z_n}^2 + c_0$$

Swift builds this orbit in CPU DD. After each step, each real and imaginary component is converted to a four-Float expansion and appended to an MTLBuffer. Thus buffer index 0 stores the first iterated value, after the initial zero. Construction stops at $N_{\max}$ or when the anchor orbit escapes beyond $r^2 = 4$. The buffer tracks both the actual entry count and the maximum iteration request used to build it. Uppercase Z_n denotes the reference orbit value after n steps; lowercase z_n denotes the actual sample's orbit value. The budget $N_{\max}$ limits the number of reference steps.

Rebuild the orbit when the center moves beyond a drift threshold, when the scale changes by more than about 15 percent, or when the requested iteration budget exceeds the existing reference. The

referenceOrigin fields for the live view store viewport origin minus c_0 as split Float pairs, keeping the per-pixel offset small.

7.2 Full quadratic perturbation recurrence

For pixel parameter $c = c_0 + \delta c$, write its orbit as $z_n = Z_n + \delta z_n$. Substitution into the quadratic map gives the recurrence below. The fixed offset δc is the pixel parameter minus the reference parameter; δz_n is the actual orbit value minus the corresponding reference value at step n. Both Mandelbrot orbits start at zero, so their initial orbit offset is zero.

$$Z_{n+1} + \delta z_{n+1} = (Z_n + \delta z_n)^2 + c_0 + \delta c$$

$$\delta z_{n+1} = 2Z_n\delta z_n + (\delta z_n)^2 + \delta c$$

This relation retains the full quadratic perturbation term, and its evaluation is still finite precision. Each Infinity pixel first evaluates a reference-relative Float form. Non-finite state, $|\delta z|^2 > \max(10^{-30}, 0.0625|Z|^2)$, or severe cancellation $|Z + \delta z|^2 < 10^{-6}|Z|^2$ when $|Z|^2 > 10^{-20}$ causes the fast path to return a negative control result, which restarts the pixel in QS. The QS path computes δc from the split reference-relative origin plus split step times the sample coordinate, reconstructs $z = Z + \delta z$, and applies the ordinary bailout.

```
deltaZ = 0
for n in 0 ..< maxIterations:
    previousReference = (n == 0) ? 0 : reference[n-1]
    nextDelta = 2 * previousReference * deltaZ + deltaZ * deltaZ + deltaC
    actual = reference[n] + nextDelta; validateOrEscalate(actual, nextDelta)
    if validEscape(actual): return escaped(n, actual); else deltaZ =
rebaseOrContinue()
```

A negative fast-tier result is a precision-routing signal, not an iteration count. The caller restarts the pixel in QS. If the QS reference is exhausted or invalid, direct QS recovery reconstructs c from the uploaded anchor and split offsets rather than falling back to an absolute Float coordinate.

The Float fast path also rejects ambiguous escapes before committing a color and restarts those pixels in QS. This ordering matters near cancellation-sensitive boundaries: the engine does not accept a plausible Float escape before checking recovery diagnostics.

Learn more about the Infinity Engine and how it uses perturbation theory.

8. Rebasing and Recovery

8.1 Why rebasing is necessary

Perturbation is most effective while δz remains small relative to the selected reference orbit. Version 3 uses two distinct diagnostic sets. The inexpensive Float tier escalates to QS at the conservative squared-magnitude ratio of 0.0625 and at the 10^{-6} cancellation thresholds. Once in the full QS recurrence, the existing rebase policy uses Float reductions of the expanded state with the wider 0.25 squared-magnitude ratio and 10^{-7} glitch thresholds shown below.

$$\text{QS largeDelta} \Leftrightarrow |\delta z|^2 > \max(10^{-30}, 0.25|Z|^2)$$

$$\text{QS glitch} \Leftrightarrow |Z|^2 > 10^{-20} \text{and} |Z + \delta z|^2 < 10^{-7}|Z|^2$$

These tests omit the step subscript: Z is the current reference value and δz the orbit offset. Thus $|Z + \delta z|^2$ is the reconstructed orbit's squared magnitude. The labels largeDelta and glitch are Boolean test results; $\Leftrightarrow$ means "if and only if." The numerical thresholds are recovery-policy settings.

If either QS condition holds, the actual orbit value becomes the new perturbation state and the reference index resets to zero. This rebase does not rebuild the CPU orbit. During direct manipulation, reference-orbit rebuilding is deferred entirely; the render of the committed viewport rebuilds or extends the DD orbit after the gesture ends.

8.2 Direct QS recovery

If a reference is exhausted or invalid, the kernel reconstructs c from the uploaded reference anchor, the split reference-relative viewport origin, and the split pixel step. It then evaluates $z_{n+1} = {z_n}^2 + c$ directly in QS. This path is intentionally rare: it is more expensive than perturbation, but it avoids falling back to an absolute DS coordinate that may already have lost the distinguishing low bits.

Escape decisions in both perturbation and direct recovery reduce the final QS z to Float for r^2. This limits the precision of the escape test even though the recurrence uses QS. The recurrence and coordinate accumulation preserve substantially more low-order information than FP32, while the terminal diagnostic remains inexpensive and consistent with the fused color pipeline.

8.3 Step representability gate

Infinity is permitted only when both axes' split steps are finite and the combined high/low magnitude on each axis is at least Float.leastNormalMagnitude. This rejects zeros, infinities, NaNs, and subnormal underflow before dispatch. A rejected step does not clamp to an invented coordinate; it selects the legacy deep path.

For Julia mode, Infinity is disabled. Julia continues to use the established GPU and CPU policies, and its fixed parameter remains stored in the model as Double but passed to the current Metal kernel as float2. Deep Julia fidelity is therefore limited by that parameter quantization even when the viewport mapping uses split arithmetic.

9. Performance Circuit and Fallback

9.1 Eligibility

A live frame can use the Infinity Engine only when all the following are true:

- the compile-time rollout gate is enabled, and the performance circuit is closed (Infinity is permitted);
- the active family is Mandelbrot;
- perturbation is enabled, refCount is positive, and a reference buffer is bound;
- the split pixel steps pass the representability test; and
- the live scale has crossed the existing CPU deep threshold with its entry hysteresis.

Infinity replaces the legacy CPU deep renderer where eligible. Normal-scale GPU rendering is not routed through QS. If Infinity becomes eligible, it invalidates any in-flight CPU generation. If it becomes ineligible, the established v2.2 CPU hysteresis again determines entry and exit.

9.2 Diagnostic gate and runtime circuit

Each completed Infinity command is timed using GPU timestamps when available, or positive wall-clock completion latency otherwise. The 0.250-second threshold is used for diagnostics and release qualification. A valid completed slow frame is retained; rendering the same image again on CPU would only add latency. Runtime fallback is reserved for a failed Metal command or a timing result that is non-finite or non-positive.

$$\text{diagnosticMiss} = \text{failed} \vee \neg\text{finite}(t_{\text{GPU}}) \vee t_{\text{GPU}} \leq 0 \vee t_{\text{GPU}} > 0.250\text{s}$$

$$\text{runtimeFallback} = \text{failed} \vee \neg\text{finite}(t_{\text{GPU}}) \vee t_{\text{GPU}} \leq 0$$

Here, t_{GPU} is the recorded duration in seconds, including the wall-clock fallback described above. The Boolean failed indicates command failure, finite excludes NaN and infinity, ∨ means "or," and ¬ means "not." Each result is true when any listed condition holds.

The 250 ms threshold remains an intentionally strict qualification signal, not a live discard rule. When runtime fallback is required, the renderer disables Infinity for the session, activates the v2.2 CPU renderer, preserves the rendering overlay, and reports completion only after the full CPU buffer has been presented.

9.3 Bounded Metal scheduling and observability

The public Infinity renderer divides the drawable into center-out two-dimensional regions, each at most 128 rows high. Depending on drawable width, each row band is divided into one to three columns. Adjacent regions are batched to keep roughly the same pixel budget as the earlier full-width 128-row commands. A private full-frame texture is initialized with a valid retained preview, then completed regions replace that preview. Fresh drawables present the progressively refined texture without holding one drawable for the whole computation. The renderer retains the finished texture by transferring ownership.

Generation checks protect both computation and presentation. A change to viewport or rendering state invalidates obsolete work; repeated identical updates do not restart a frame. Completed center-out regions are visible while the remaining image is still a preview. Full completion is reported only after the current generation is presented successfully. InfinityEngineMetrics includes compute and progressive-presentation GPU time, diagnostic budget status, and runtime acceptance. The build 548 timing corpus predates this presentation scheduler.

10. Sampling and Iteration Policy

10.1 Deterministic SSAA

Supersampling Anti-Aliasing (SSAA) runs inside each output thread. For $N = m^2$ samples with $m \in \{1,2,3,4,5\}$, subpixel (u, v) uses the centered offset

$$\delta x = \frac{u + \frac{1}{2}}{m} - \frac{1}{2}, \delta y = \frac{v + \frac{1}{2}}{m} - \frac{1}{2}$$

Here, m is the number of samples along each pixel axis and N is the total number of samples per output pixel. The integer grid indices u and v each run from 0 through m − 1. The offsets δx and δy are measured in pixel widths from the output pixel's center; they are not complex-plane distances.

Each subpixel performs a full orbit test and palette lookup. The kernel accumulates RGB values into a float3 register and divides by N once. The grid is deterministic and axis-aligned; it is neither stochastic nor temporally accumulated. During Infinity deep rendering, the current kernel fixes the requested sample count at one to control rendering cost while the new recurrence is qualified.

10.2 Interactive and idle quality

The renderer separates navigation latency from settled quality. Pan, pinch, and double-tap transform the retained complete frame immediately. Consecutive gestures compose against that retained source instead of repeatedly resampling or resetting the image. Newly exposed areas receive a valid preview fill rather than repeated edge pixels. After a gesture commits, the new view refines in center-out regions, with progress measured against the complete requested frame. The preview remains temporary until the requested render completes.

Reference-orbit rebuilding is deferred during direct manipulation. Once the gesture commits, the renderer renders the committed viewport with its selected iteration budget, lighting, and qualified Infinity or CPU path. The rendering overlay remains until the current frame completes presentation. Auto Iterations is frozen or disabled on the constrained deep path; manual iteration and contrast state remain authoritative for the settled request.

10.3 Manual iteration control

The manual slider is a normalized control $p \in [0,1]$ backed by an exact inverse mapping. It uses a two-segment power curve with $\gamma = 2.8$: half of the physical travel is reserved for 1–300 iterations, and the other half covers 300 through the selected cap. This gives precise low-cost control without making 75,000 iterations unreachable.

$$\text{For } 0 \le p \le 0.5: u = \frac{p}{0.5}, N_{\text{raw}} = N_{\min} + (300 - N_{\min})u^{2.8}$$

$$\text{For } 0.5 < p \le 1: u = \frac{p - 0.5}{0.5}, N_{\text{raw}} = 300 + \left(N_{\text{cap}} - 300\right)u^{2.8}$$

Here, p is the slider position over its full travel, while u is the position within the selected half. The minimum $N_{\min}$ is 1 iteration, N_{cap} is the selected ceiling, and N_{raw} is the count before rounding. The exponent γ controls the curve's shape. This local u is distinct from the SSAA grid index in Section 10.1.

At or below 300, the system rounds the value to the nearest integer. Above 300, it rounds the value to the nearest multiple of 10. The inverse raises the normalized segment value to $\frac{1}{2.8}$, so bookmark loads, cap changes, Auto-to-manual transitions, and step buttons place the thumb consistently.

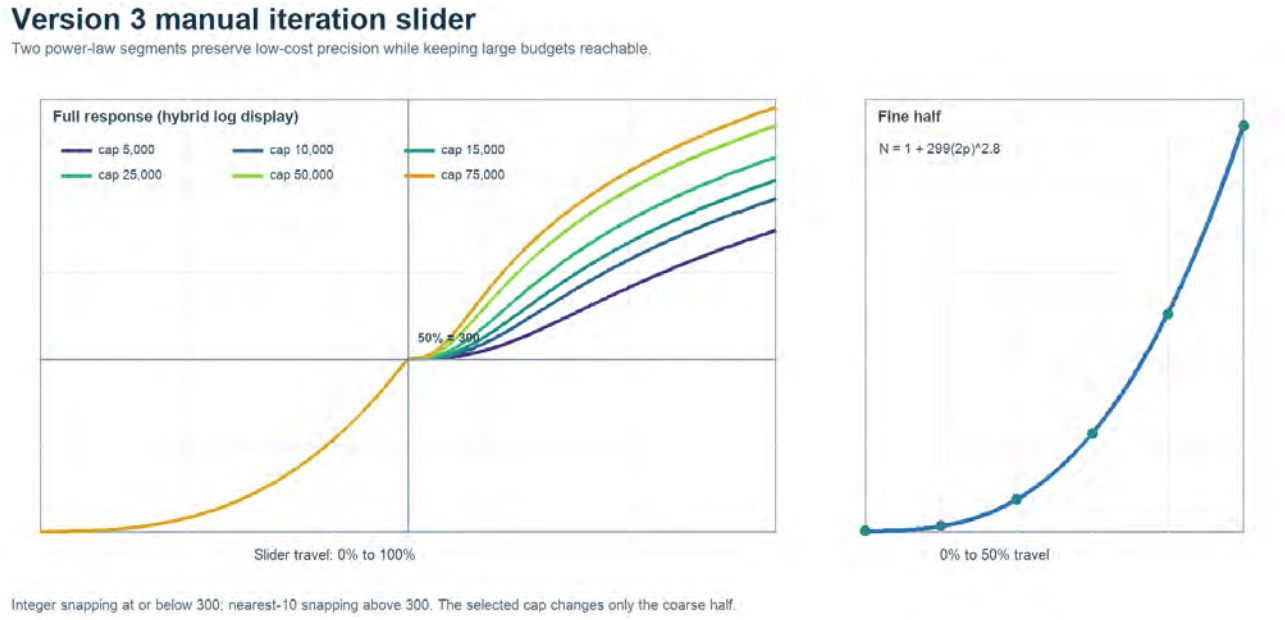

Figure 4 —Iteration-slider response. The fine half is identical for every cap; the selected cap controls the coarse half.

10.4 Selectable maximum-iteration cap

The Settings menu persists one of six upper bounds. The cap is a safety and usability control, not an automatic quality guarantee. Reducing it immediately clamps any current iteration value above the new cap; increasing it expands the slider's coarse segment.

The 50,000-iteration default preserves broad headroom without making the low end hard to adjust.

Preset	Purpose in the control model	Coarse segment
5,000	Fast ceiling for routine exploration	300–5,000
10,000	Moderate manual ceiling	300–10,000
15,000	Higher-detail working ceiling	300–15,000
25,000	Deep-scene tuning	300–25,000
50,000 (default)	General full-range default	300–50,000
75,000	Maximum user-selectable budget	300–75,000

Table 5 — User-selectable iteration caps in Version 3.0.

10.5 Auto Iterations

Auto Iterations uses a UI proposal and a renderer target. The UI proposal handles committed scale changes and supplies a display fallback until the renderer reports an effective count. For the UI calculation, S is the navigation scale and S_{baseline} is the stored model baseline; both are in points per complex-plane unit:

$$\rho = \max\left(1, \frac{S}{S_{\text{baseline}}}\right) \text{ and } \ell = \log_{10}(\rho).$$

The dimensionless zoom ratio ρ (rho) is floored at 1. The variable l (lowercase ell) counts tenfold increases above the baseline, using $\log_{10}$, the base-10 logarithm. The functions max and min select the larger and smaller argument, respectively.

The proposed count N_{UI} starts from N_{base}, the current iteration value stored in the application model, and is limited by the selected cap N_{cap}:

$$N_{\text{UI}} = \min\left(N_{\text{cap}}, N_{\text{base}} + \text{round}(250 + 900\ell + 350\ell^2)\right).$$

The constants 250, 900, and 350 set the initial boost, linear growth, and quadratic growth; round means nearest integer. With the base count, baseline, and cap held fixed, this proposal does not decrease as scale increases. The renderer's frame target uses a separate calculation: here S is the drawable scale and S_{ref} is the renderer's current baseline, both in pixels per complex-plane unit. That baseline may be rebased as described in Section 3.2.

$$o = \log_2\left(\frac{S}{S_{\text{ref}}}\right) \text{ and } q = \max(0, o - 9),$$

$$N_{\text{raw}} = \max(80, [240 + 26o + 4q^2] \cdot M_{\text{SSAA}} \cdot M_{\text{live}} \cdot M_{\text{LUT}} \cdot 1.60).$$

Here, o counts zoom doublings relative to the renderer baseline, and q counts only the doublings beyond nine (a scale ratio of 512). This q is unrelated to the QS expansion in Section 6.3. The target N_{raw} is the renderer's estimate before integer rounding and smoothing. The constants 240, 26, and 4 shape its growth; 80 sets the minimum and 1.60 is the overall gain.

$$M_{\text{SSAA}} = \frac{1}{1+0.08(f-1)} \text{ for the active sample-count field f;}$$

$$M_{\text{live}} = 0.90 \text{ during interaction;}$$

$$\text{and } M_{\text{LUT}} = \min\left(2.2, \sqrt{\frac{w}{256}}\right) \text{ for LUT width w.}$$

The three M factors adjust that estimate. M_{SSAA} reduces the budget as sampling work increases: f is the total active samples per pixel (1, 4, 9, 16, or 25), not the per-axis grid size m. M_{live} is 0.90 during interaction and 1 otherwise. M_{LUT} adjusts for palette lookup width w, measured in texels: it is 1 at 256 texels and 2 at 1,024. The displayed LUT formula covers widths of at least 256; the implementation floors the multiplier at 1 for smaller widths and caps it at 2.2.

The renderer rounds the estimate and blends it with a stored exponential average: a higher target receives 30% weight and a lower target 70%, allowing faster decreases. The stabilizer then blends its rounded proposal with the same stored average using a 25% weight. The 8% scale threshold updates a stored scale anchor; however, it does not, itsef, fix the iteration count.

On the normal GPU path with Auto enabled, the renderer writes the stabilized count into the frame's iteration budget $N_{\max}$. The display uses the renderer's reported effective count when available. The cap N_{cap} explicitly limits the UI proposal; the renderer formula above applies the 80-iteration floor without that cap. Its output can rise or fall as scale, sampling, interaction, palette width, or baseline changes.

Auto Iterations freezes or disables when the constrained deep path takes control. This prevents an expensive deep render from chasing a moving iteration target and keeps the HUD, bookmark state, and final command consistent. Manual mode always treats the displayed iteration value as authoritative.

11. Palette, Color, and Output

11.1 Palette model

The palette library currently contains 156 built-in palette definitions, including 55 Ultra-Wide source ramps with 768 samples. The number of source stops is independent of the active 256- or 1024-texel lookup texture. The host resamples the source to that width. Imported gradients carry editable stops, a name, a schema version, and a color-space label, then rasterize again on the receiving device.

A palette change invalidates a ready CPU frame, increments the palette version, and requests a fused redraw. Because the same kernel computes orbit and color, color changes can affect performance measurements and must be included in reproducibility metadata.

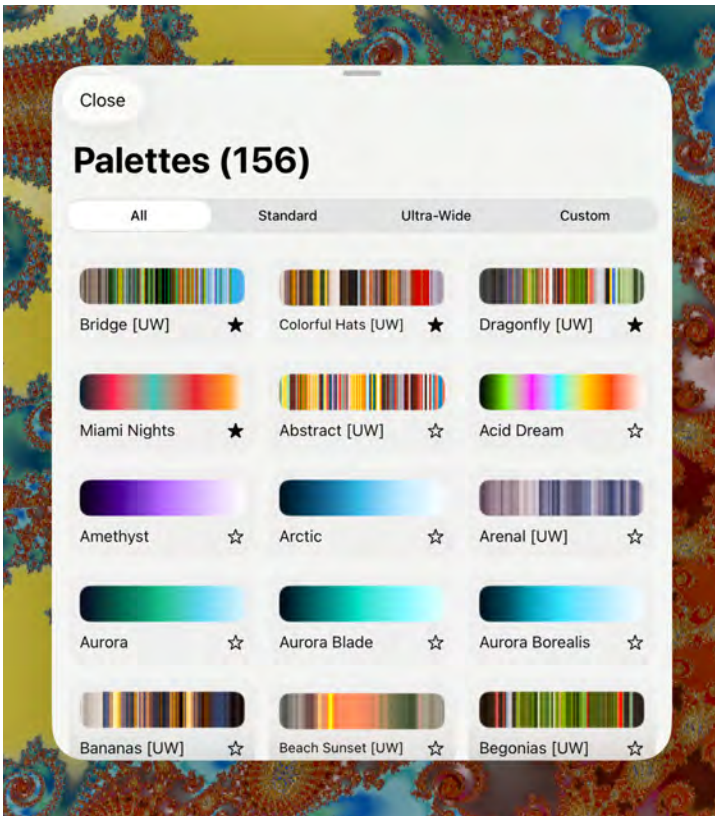

Figure 5 — The standard palette library in Mandelbrot Metal, shown here on an iPad Pro. Pro users can create their own custom palettes.

11.2 Catalog organization and discovery

PaletteOption represents each catalog entry with a display name, an optional procedural built-in index, and an ordered array of normalized color stops. The picker (see Figure 5) filters the combined catalog into All, Standard, Ultra-Wide, and Imported views. Favorites are persisted by name and sorted ahead of non-favorites without duplicating palette definitions.

Ultra-Wide palette names carry the [UW] suffix. Their 768-sample source ramps follow the same resampling path as other named palettes; the suffix identifies source density, not a separate shader mode.

11.3 LUT construction and shader sampling

For a named or imported palette, Swift/Core Graphics creates a one-pixel-high BGRA8 image in the selected working color space. The Hi-res LUT control selects 1×1,024 or 1×256 texels. The texture is

uploaded to the LUT slot, and the fused kernel samples it at $(t, 0.5)$ using linear filtering and clamp-to-edge addressing.

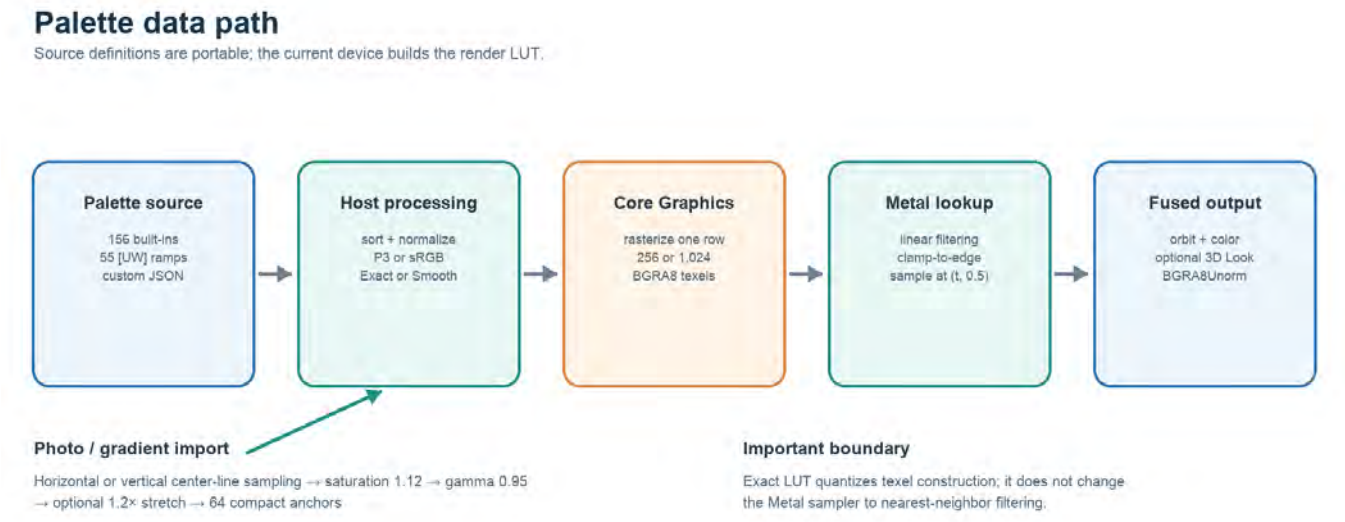

Figure 6 — Palette source, host rasterization, Metal lookup, and fused presentation path.

11.4 Smooth and Exact LUT modes

Smooth mode draws a continuous Core Graphics gradient across the destination LUT. Exact LUT mode instead assigns each destination texel the nearest source stop, preserving deliberate bands and sampled structure. The Metal sampler remains linear in both modes, so 'Exact' describes texture construction, not a nearest-neighbor shader sampler. At 256 texels the texture is compact; at 1,024 texels it better preserves dense source variation at modest memory cost.

11.5 Gradient Import processing

Gradient Import downsamples the source image to an RGBA8 working raster and samples the center row or center column. Version 3 defaults to 1,024 samples, Display-P3, saturation multiplier 1.12, gamma 0.95, and an enabled 1.2× value stretch. Compact mode reduces the result to 64 evenly spaced anchors by nearest-source selection; Exact mode retains every sampled stop. The editor can preview, name, save, rename, delete, favorite, import, and export custom palettes.

The gamma control applies to palette import; it does not make the renderer operate in scene-linear light. The target color-space label controls Core Graphics conversion and JSON metadata; the active Metal LUT and output remain untagged BGRA8Unorm resources.

11.6 Photo2Palette command-line parity

The bundled photo2palette.swift utility is versioned for Mandelbrot Metal 3.0 and uses the same defaults as the in-app importer. It accepts PNG, JPEG, HEIC, and other ImageIO formats; auto, horizontal, or vertical sampling; counts of 2–16,384 for source steps and compact anchors; Exact or compact output; Display-P3 or sRGB; saturation, gamma, stretch, and stretch-factor controls; and JSON or developer Swift output. Palette JSON remains schema version 1 and is compatible with 2.x and 3.x.

The photo2palette utility is available on GitHub.

11.7 Persistence, sharing, and redraw cost

Custom palette stop arrays and their names persist locally; imported names are normalized, marked with a [C] suffix, and adjusted to avoid duplicates. A palette share file contains type, schemaVersion, name, colorSpace, and normalized t/r/g/b stops. Import validates the schema, color space, nonempty name, at least two stops, and every component's [0, 1] range before registration.

A palette edit reruns the fused rendering pipeline described in Section 1.1. Consequently, benchmark and reproducibility records must include palette identity, LUT width, color-space choice, Exact/Smooth mode, contrast, and 3D Look.

Layer	Representation	Version 3 responsibility
Catalog	PaletteOption + stop arrays	Built-ins, [UW] ramps, imported palettes, favorites, filters
Portable file	Palette schema v1 JSON	Name, color space, normalized stops; object or array
Host LUT	1×256 or 1×1,024 BGRA8	Smooth gradient or nearest-stop texel construction
Metal sampler	linear + clamp-to-edge	Lookup at normalized escape coordinate t
Final output	BGRA8Unorm	SSAA average, optional 3D Look, display/capture

Table 6 — Palette representations and their distinct contracts.

11.8 Color-management boundary

The active Metal targets are BGRA8Unorm. Display-P3 or sRGB choices participate in Core Graphics LUT construction and UIKit image creation, but the Metal texture itself is not an sRGB-tagged or floating-point buffer. The implementation therefore does not establish scene-linear SSAA averaging, HDR precision, or a validated ICC round trip.

11.9 Visual structure at depth

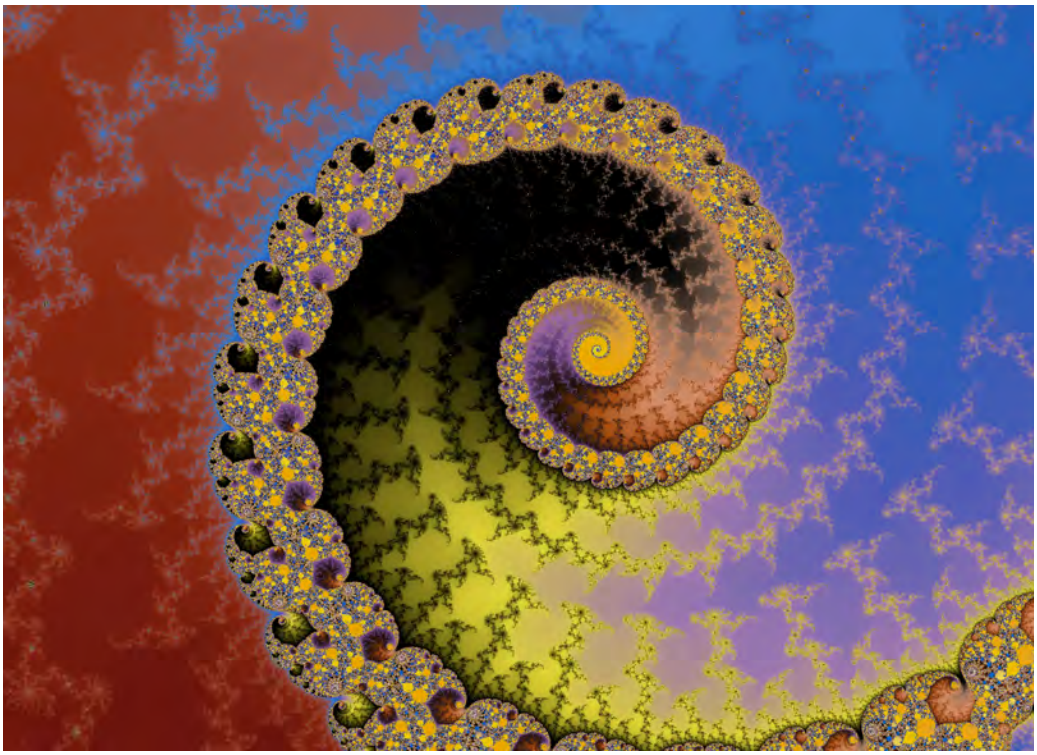

Figure 7 — A deep boundary render. Precision determines which sample coordinates remain distinct; palette and lighting determine how they appear.

At increasing scale, adjacent pixels correspond to smaller δc values. Once those values collapse in the chosen representation, additional zoom changes magnification but cannot reveal new coordinate distinctions. Infinity Engine postpones that collapse while retaining GPU throughput; it cannot remove the finite-iteration and finite-representation boundaries described throughout this paper.

12. Capture and Export

12.1 Canvas and offscreen paths

All captures, including Canvas Size, render a dedicated export image. The user-facing capture path calls the tiled exporter with 256-pixel tiles and waits for the complete image, avoiding races with partially refined live textures. Canvas Size uses the drawable dimensions; 4K, 6K, and 8K preserve the canvas aspect with nominal longest edges of 3,840, 5,760, and 7,680 pixels. Custom capture accepts explicit dimensions. Each export recomputes mapping and precision eligibility for its output.

Offscreen output is BGRA8 and is serialized as PNG through the platform image APIs. Export can differ from the live view because it uses a different pixel step, output size, tile policy, and potentially a different precision path. Export re-renders the same composition at the requested size; it does not copy the live pixels.

12.2 Export-specific Infinity qualification

Version 3 adds an explicit export gate. A high-resolution request first computes its effective pixels-per-unit and split step. Infinity is enabled for that capture only if the rollout gate is enabled and the circuit permits Infinity, Mandelbrot mode is active, a current reference anchor/buffer exists, the export scale crosses the deep threshold, and both export step components pass the normal-magnitude test.

Export eligibility is independent of the live frame's route. A live Infinity Engine frame does not imply that a much larger export has a representable step, and a live non-Infinity frame does not imply that a higher-resolution output should be denied Infinity. Snapshot and tiled capture therefore set their own infinityMode rather than inheriting the live flag.

12.3 Tiled export

The tiled exporter derives every tile origin from the full export mapping, so offset errors do not accumulate from tile to tile. It snapshots the selected center, scale, iteration budget, and render state, renders each region, and assembles the output before invoking successful completion. Deep Mandelbrot export retains the qualified CPU DD fallback when Infinity cannot be used. Tile textures limit per-command resource use, but the assembled image still has a memory cost proportional to output area.

12.4 Completion, validation, and sharing

Capture completion accepts the fully assembled image returned by the exporter, serializes it as PNG, writes a temporary file atomically, and opens the system share sheet. The current user-facing path does not call the retained imageLooksAllBlack helper, so this path does not validate image brightness. A black image can also be a legitimate finite-iteration interior view. Failed rendering or PNG/file creation must not be counted as successful artwork sharing.

Free users can export the canvas at the screen resolution. Pro enables the 4K, 6K, 8K, and custom-dimension rerender paths. The entitlement changes output options, not viewport mathematics or fractal correctness.

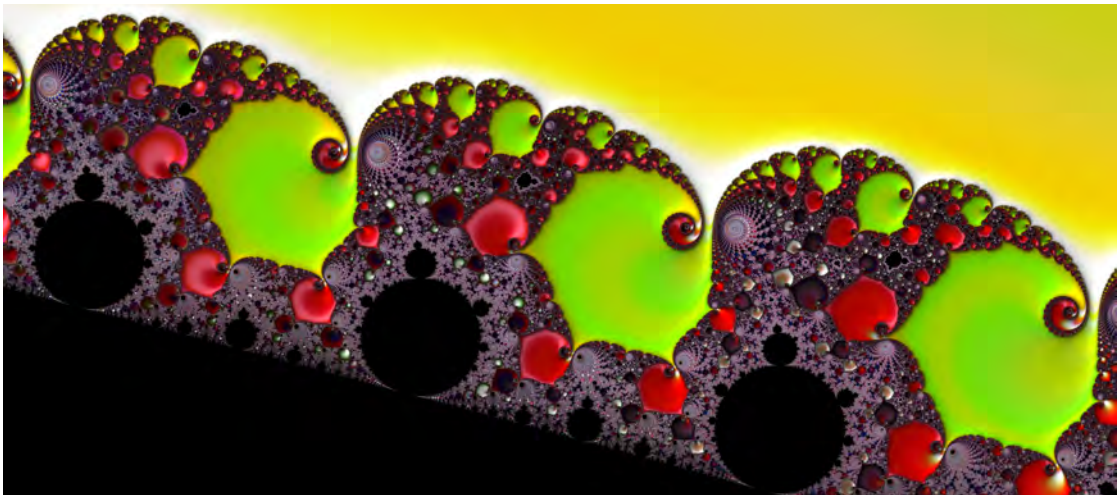

Figure 8 — Coordinate Jump render used as a composition example; export recomputes every output pixel from the saved state.

13. User Interface, State, Bookmarks, and Sharing

13.1 Exploration state

ExplorationState is an Equatable value snapshot containing center, scale, baseline scale, max iterations, Auto state, fractal mode, Julia c, the Mandelbrot return snapshot, palette index and name, contrast, 3D Look, and loaded-bookmark identity. History therefore restores rendering and appearance settings as well as the viewport.

Continuous inputs are tracked by a set of interaction kinds—pan, pinch, iterations, and contrast—rather than a callback reference count. The first active input captures the departure state. That state is committed only after all active inputs finish. Duplicate begin/finish callbacks have no additional effect, preventing navigation from becoming permanently locked.

13.2 Cursor timeline

The v3 history implementation uses one bounded timeline and a currentIndex cursor. A transition stores both departure and arrival. Going Back or Forward first replaces the current timeline slot with the actual on-screen state, then moves the cursor. Recording a new transition after Back deletes the obsolete forward branch. Capacity 50 means up to 50 Back steps plus the current state.

```
record(departure, arrival):
    ignore if departure == arrival
    replace timeline[cursor] with departure
    delete timeline after cursor
    append arrival; cursor = last
    trim oldest entries to capacity + 1
```

13.3 Determinism boundary

Bookmarks and history can restore composition state, and SSAA sample placement is deterministic. Restoring that state does not guarantee identical output pixels. GPU/CPU/Infinity path selection, OS and GPU revision, Float operation ordering, command scheduling, texture conversion, and color-management behavior can change pixels while preserving the same mathematical viewport.

Version 3 reproduces saved state within the limits described above. A more complete render manifest would also need shader/build identity, device/OS identity, the specific arithmetic path, reference anchor, iteration policy, LUT bytes and color metadata, and output conversion settings.

13.4 Responsive iPhone and iPad interface

ContentView is the SwiftUI composition root around an MTKView-backed renderer. iPhone uses a compact bottom control area within the safe area and a separate top utility row; iPad uses a wider floating control bar and adapts to portrait, landscape, and narrow split-view widths. Both layouts bind to the same state and renderer methods, preventing phone- and tablet-specific control semantics from diverging.

The principal controls are pan, focal-point pinch zoom, 2× double-tap zoom, Back, Forward, Reset, Mandelbrot/Julia selection, iterations, Auto Iterations, contrast, palette selection, Gradient Import, Coordinate Jump, 3D Look, Capture, bookmark management, and the overflow/settings menu. Controls show their availability and Pro requirements where they are used. Hidden or disabled controls must not silently alter renderer behavior.

13.5 Gesture continuity and full-frame progress

The UIKit gesture bridge supplies pan, pinch focal-point, and double-tap coordinates in the actual Metal canvas space. While a gesture is active, the retained frame follows the interaction immediately. On commit, the compensated center and scale are recorded and stale work is invalidated. The public Infinity path then progressively replaces an initialized preview with completed center-out regions. The preview and final image have distinct roles; uninitialized pixels must never be presented.

Deep-render progress is a percentage of the requested full frame. Completed regions may appear before the percentage reaches 100. The progress overlay ends only after successful presentation of the complete current-generation image. Cancellation and generation checks prevent obsolete tiles or delayed completions from replacing the latest view.

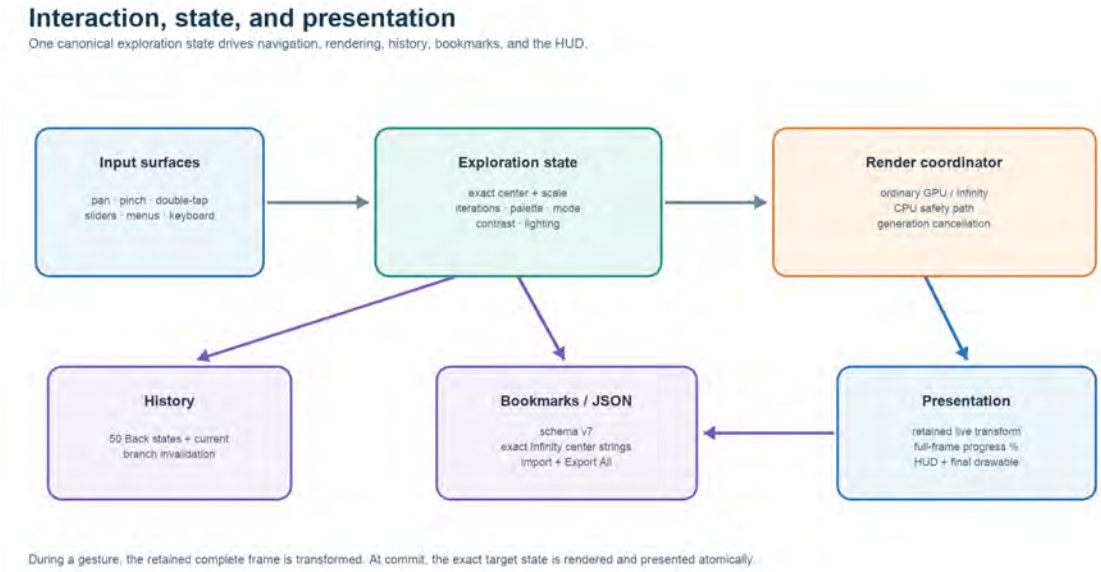

Figure 9 — State flow. Version 3 progressively defines the committed view. Final completion follows the full frame's presentation.

13.6 Exploration HUD semantics and numeric formatting

The exploration heads-up display (HUD) summarizes navigation state; it is not the stored state representation. It reports fractal family, center, scale, iteration budget, contrast, quality/path, palette identity, LUT/color mode, and—when relevant—Julia c and loaded bookmark identity. Center coordinates use 15 significant digits from the compensated projection so the overlay remains legible; exact fixed-point center strings remain in renderer and bookmark state.

Below 10^{20}, scale is shown as a locale-grouped integer followed by scientific notation in parentheses. At and above 10^{20}, only a three-decimal scientific form is shown—for example 4.171×10^{24} ×—avoiding two

redundant long numbers. A low compensated limb can change even when the nearest Double does not; the state model, not the rounded HUD string, is authoritative.

13.7 Bookmark model and management

A bookmark captures a complete creative state: UUID and optional creation date; bookmark name; Double center plus residual; optional canonical Infinity real/imaginary strings; current and baseline scale; fractal mode and Julia constant; palette index and optional custom name; 3D Look; deep and perturbation flags; iterations and Auto state; and contrast. Older local records decode with safe defaults for fields introduced later.

The manager lists, sorts, loads, renames, deletes, imports, and exports the selected bookmark, or all bookmarks. Sort options include newest, oldest, name, palette, mode, iterations, and scale. Free supports five saved bookmarks. Pro removes that organizational limit without changing existing saved data.

State group	Fields retained	Why it matters
Viewport	center, residual, exact Infinity strings, scale, baseline	Preserves deep composition and navigation reference
Dynamics	Mandelbrot/Julia mode, Julia c, iterations, Auto	Restores the finite mathematical problem
Appearance	palette index/name, contrast, 3D Look	Restores creative presentation
Routing	deep and perturb flags	Maintains compatibility intent; runtime still requalifies
Identity	UUID, name, creation date	Supports management, sorting, and duplicate handling

Table 7 — Local bookmark state captured by Version 3.0.

13.8 Portable bookmark schema and Version 2 compatibility

Portable bookmark JSON is currently schema version 7. The decoder accepts a single object or an array and validates versions 1 through 7. Version 7 adds canonical Infinity center strings alongside Double centerX/centerY and optional low limbs. This preserves the canonical v3 center strings during transfer while retaining numeric fields older readers understand.

The palette reference is one of builtIn, customByName, or embedded. Current bookmark export uses a built-in index or a custom name; embedded stops are accepted during import. To reliably transfer a custom-by-name bookmark to a device that does not already own that palette, include the palette JSON with the bookmark. Import resolves [C] suffix variants, assigns unique bookmark names, registers embedded palettes, and preserves Version 2 defaults.

Schema group	Representative fields	Compatibility behavior
Envelope	type, schemaVersion, name	Rejects future/unsupported versions
Coordinates	centerX/Y, centerLowX/Y, Infinity strings	Older numeric centers remain valid; v7 retains exact center
Fractal	fractalMode, juliaCX/Y	Missing fields default to Mandelbrot and standard Julia c
Rendering	scale, baseline, iterations, Auto, contrast, deep, perturb	Validated, then runtime path is requalified
Palette / 3D	reference union, optional 3D structure	Supports built-in, named custom, embedded, and older no-3D files

Table 8 — Bookmark schema v7 and backward-compatibility contract.

13.9 Coordinate Jump and Mandelbrot–Julia linkage

Coordinate Jump accepts a complex-plane rectangle, derives its center, and chooses an aspect-preserving scale for the live canvas. The result passes through the same scale clamp, exact-center model, history transition, and renderer qualification as gestures. Tap-to-Julia maps the selected Mandelbrot coordinate to the fixed Julia parameter c and records a Mandelbrot return snapshot, making the family transition reversible through the interface and history model.

13.10 Persistence, localization, and accessibility

At launch, the app restores the last valid exploration state and preferences only after a positive-size viewport exists. The launch presentation belongs to the current explorer window, preserving the live MTKView while artwork is shown. The standard opening lasts three seconds including a 0.35-second fade and can be skipped. With Reduce Motion enabled, the opening uses a shorter transition. Backgrounding dismisses the opening; late callbacks cannot dismiss another window's launch. Phone, tablet, rotation, and narrow-window artwork are selected from actual canvas geometry.

Version 3 localizes application and guide text through String Catalogs in 16 languages and regional variants: Arabic, German, English, Spanish, French, Hebrew, Italian, Japanese, Korean, Dutch, Polish, Brazilian Portuguese, Russian, Swedish, Ukrainian, and Simplified Chinese. SwiftUI controls inherit platform accessibility behavior, while explicit labels are used where an icon alone would be ambiguous. Right-to-left layout, Dynamic Type, compact iPhone geometry, iPad split view, and long translated strings remain part of the release test matrix.

Automatic review requests follow a successful bookmark or artwork-sharing action after the canvas becomes idle. Eligibility starts with one active day and one completed action, with a 120-day cooldown, at most three attempts in a rolling year, and at most one attempt per marketing version. Launch alone, canceled shares, failed shares, inactive scenes, and stale callbacks cannot trigger an attempt. StoreKit decides whether to show a request and does not report whether a review was submitted.

14. Free + Pro Entitlement Internals

14.1 StoreKit 2 ownership model

Version 3.0 introduces one app binary with a Free state and four Pro-granting entitlement states: legacyPro, monthly, annual, and lifetime. Views depend on the single derived property hasProAccess rather than issuing StoreKit queries. This centralizes feature-access decisions and makes them testable.

State	Source	Access semantics
free	No verified Pro or legacy entitlement	Free feature set
legacyPro	Verified original application version predates v3.0 boundary	Permanent Pro access for existing owner
monthly	Current verified monthly transaction	Pro while transaction remains current
annual	Current verified annual transaction	Pro while transaction remains current
lifetime	Current verified non-consumable transaction	Permanent Pro access

Table 9 — StoreKit 2 ownership states and Pro access semantics.

14.2 Products and transaction resolution

The controller loads three App Store products: MSC.MandelbrotMetal.pro.monthly, MSC.MandelbrotMetal.pro.annual, and MSC.MandelbrotMetal.pro.lifetime. It observes Transaction.updates for the lifetime of the app, finishes verified purchases, ignores revoked or expired transactions, and resolves the highest-ranked current entitlement. Restore invokes AppStore.sync and then repeats the same verified-resolution pass.

Purchase outcomes include purchased, pending, canceled, and unverified. Unverified transactions do not grant access. Display prices come from StoreKit; catalog prices are development fallbacks. The local StoreKit configuration marks all three products as Family Shareable, while App Store Connect must configure the corresponding real products. Permanent legacy/lifetime access is cached for offline launch. A verified lifetime revocation clears the matching permanent entitlement.

14.3 Version 3.0 feature policy and reference pricing

Free and Pro use the same precision-selection and rendering code. Free includes Mandelbrot and Julia exploration, the built-in palette catalog, five saved bookmarks, and Canvas Size PNG capture. Pro adds unlimited bookmarks, custom/photo palette tools, 3D lighting, and 4K/6K/8K/custom PNG capture. Scene documents and motion/video authoring remain future creation-system work; their feature labels and entitlement cases do not establish shipping functionality in v3.0.

Plan	US reference price	Access
Free	$0	Full exploration engine; bounded organization and output
Pro Monthly	$2.99 / month	Complete Pro feature set while current
Pro Annual	$19.99 / year	Complete Pro feature set; recommended subscription
Pro Lifetime	$39.99 once	Permanent complete Pro feature set
Legacy Pro	No new charge	Permanent Pro for verified pre-v3 owners

Table 10 — Version 3.0 reference pricing and access terms.

These are the v3.0 development-catalog prices in the United States; App Store Connect approval, tax, and storefront localization remain authoritative. The upgrade screen presents live StoreKit prices, Restore Purchases, subscription management, privacy and EULA links, and contextual explanations when a Free limit is reached.

14.4 Legacy Pro migration

AppTransaction.originalAppVersion supplies the verified original-download version in production. On iOS and iPadOS, verified original versions below build 547 map to legacyPro. Numeric components are compared component-by-component, so 2.10 correctly sorts after 2.9 and missing trailing components are treated as zeros.

Sandbox reports an original version of 1.0, which would incorrectly classify every new test user as legacy. Migration is therefore production-only. Debug arguments −MMForceFree and −MMForceLegacyPro provide deterministic local tests without weakening production verification.

15. Verification and Qualification

15.1 Deterministic tests

The latest build contains 68 deterministic tests in four suites. Coverage includes precision routing, the enforced 10^{25} navigation ceiling, center-out coverage, stale-frame cancellation, retained previews,

compensated gestures, history, bookmark compatibility, localization, pricing IDs, Free/Pro access, legacy migration, review eligibility, and the launch experience. Section 15.2 records the execution evidence separately from the historical physical-device benchmarks.

Gate	Pass condition	Fallback or failure behavior
Uniform ABI	Swift and Metal layouts compile and bind consistently	Build/test failure; do not ship
Metal compile	Shader library compiles without diagnostics	Retain previous shader
Infinity step	Both split step magnitudes finite and normal	Legacy CPU deep path
Infinity runtime	Successful command with finite, positive timing; 250 ms retained as diagnostic target	Keep valid slow frame; failed/unusable command opens circuit and redraws on CPU
Export Infinity	Export-specific mapping passes full eligibility	CPU DD for deep Mandelbrot export
Store transaction	Verified, current, known product	No Pro grant
Legacy migration	Verified production AppTransaction predates boundary	Free unless another entitlement exists

Table 11 — Version 3 qualification gates and fallback behavior.

15.2 Regression checks completed for the v3 branch

- Signed Release builds completed for the production target and for separately identified v3.0 and v2.2.3 physical benchmark bundles.
- The historical build 548 qualification used 24 tests. Later coverage expanded to 68 tests in four suites, including six launch tests, review-prompt timing, center-out tile coverage, preview filling, interrupted redraws, and the enforced 10^{25} boundary. The September 8 full run passed 68 tests; the final three-second launch revision passed its six-test suite separately. A fresh build 570 run on September 9 also passed all 68 tests on the iOS 26.5 simulator.
- Physical-device benchmark launches bypassed onboarding without changing production preferences and required a foreground, positive-size, window-attached Metal drawable.
- The final build 548 forward and reverse benchmark passes kept every Infinity-eligible Mandelbrot case on the bounded Metal path; no circuit opened, and no deep row used an ordinary Float fallback.
- The adaptive hybrid renderer was byte-identical to an all-QS control for the 440×956 Earth Elephants comparison: 420,640 pixels, 0 differing pixels, and 0 channel errors.

The build 548 physical benchmark used an iPhone 17 Pro Max with iOS 26.6.1. Later simulator checks include phone/tablet launch compositions and deterministic rendering/state tests; those checks do not replace physical iPad thermal, rotation, memory-pressure, or export qualification. StoreKit sandbox purchases, renewals, cancellation, restoration, Family Sharing, and the complete high-resolution export matrix remain distinct release checks.

15.3 Image comparison with an all-QS control

To check the adaptive path against the all-QS control, a 440×956 adaptive hybrid render of Earth Elephants was compared byte-for-byte with a build forced through QS for every pixel. Across 420,640 pixels, there were zero differing pixels and zero channel errors. This checks agreement with the all-QS control, not independent mathematical correctness. The timing corpus instead uses the full 1320×2868 drawable and the production adaptive path. Additional fixtures should continue to cover cancellation-heavy, reference-escape, orbit-exhaustion, Julia, export, and multiple-reference scenes.

15.4 Benchmark—physical-device v3.0 versus v2.2.3

Method

For the historical build 548 benchmark, I ran release builds with separate bundle identifiers on an iPhone 17 Pro Max (iOS 26.6.1). The benchmark required the foreground MTKView to be window-attached and fixed the live surface at 440×956 points, scale 3, for a full 1320×2868 drawable. Timing began immediately before bookmark state application and ended only after the complete command path—including DD reference preparation, recurrence, color/lighting, final blit, and presentation submission—completed. I excluded app launch and drawable-readiness waiting. High Quality Idle was off because it is a global state not present in exported bookmarks; each bookmark retained its iterations, palette, contrast, 3D state, fractal family, and Julia parameter.

Each version ran once in forward corpus order and once in reverse. The table reports both passes and their arithmetic mean; speedup is v2.2.3 mean divided by v3.0 mean. Effective scale is drawable pixels per complex-plane unit—the stored point scale multiplied by the device scale of 3. Route labels show v2.2.3 → v3.0; Glassy Spirals changed between GPU and CPU in the two v2 orders, so its label is GPU/CPU → GPU.

Raw data

Scene, effective scale, iterations, appearance, & routes	v2 F	v2 R	v2 mean	v3 F	v3 R	v3 mean	Speedup
Julia Dragon $\mathbf{1.30 \times 10^{3}}$ px/u · 2,170 · 3D GPU→GPU	0.075	0.074	0.075	0.068	0.073	0.070	1.06×
Laser Quad $\mathbf{1.55 \times 10^{3}}$ px/u · 117 · 3D GPU→GPU	0.035	0.042	0.039	0.039	0.042	0.041	0.95×
Mini Brot All Dressed Up $\mathbf{9.90 \times 10^{6}}$ px/u · 3,610 · 3D GPU→GPU	0.205	0.209	0.207	0.217	0.182	0.199	1.04×
Glassy Spirals $\mathbf{8.03 \times 10^{7}}$ px/u · 4,250 · 2D GPU/CPU→GPU	0.530	2.169	1.350	0.530	0.151	0.341	3.96×
Elephant Sweep 3D $\mathbf{5.24 \times 10^{8}}$ px/u · 1,090 · 3D CPU→Infinity	3.487	3.653	3.570	0.417	0.422	0.419	8.52×
Stranger Things $\mathbf{6.11 \times 10^{8}}$ px/u · 8,180 · 2D CPU→Infinity	0.853	0.885	0.869	0.198	0.204	0.201	4.33×
Valley Of Bulbs $\mathbf{5.85 \times 10^{9}}$ px/u · 35,710 · 2D CPU→Infinity	56.571	58.852	57.711	9.491	12.164	10.827	5.33×
Banana Spears $\mathbf{9.54 \times 10^{11}}$ px/u · 14,930 · 2D CPU→Infinity	12.497	13.004	12.750	1.521	1.707	1.614	7.90×
Deep Deep Triplets $\mathbf{4.88 \times 10^{13}}$ px/u · 47,550 · 2D CPU→Infinity	133.163	96.987	115.075	24.919	25.066	24.992	4.60×
Earth Elephants $\mathbf{1.25 \times 10^{15}}$ px/u · 9,860 · 2D CPU→Infinity	12.749	11.741	12.245	3.972	3.455	3.714	3.30×
Total of all 10 scenes	**220.164**	**187.616**	**203.890**	**41.372**	**43.467**	**42.419**	**4.81×**

Table 12 — Full app-to-present physical-device timing in seconds. F = forward corpus order; R = reverse. Arithmetic means and ratios use unrounded values.

Summary

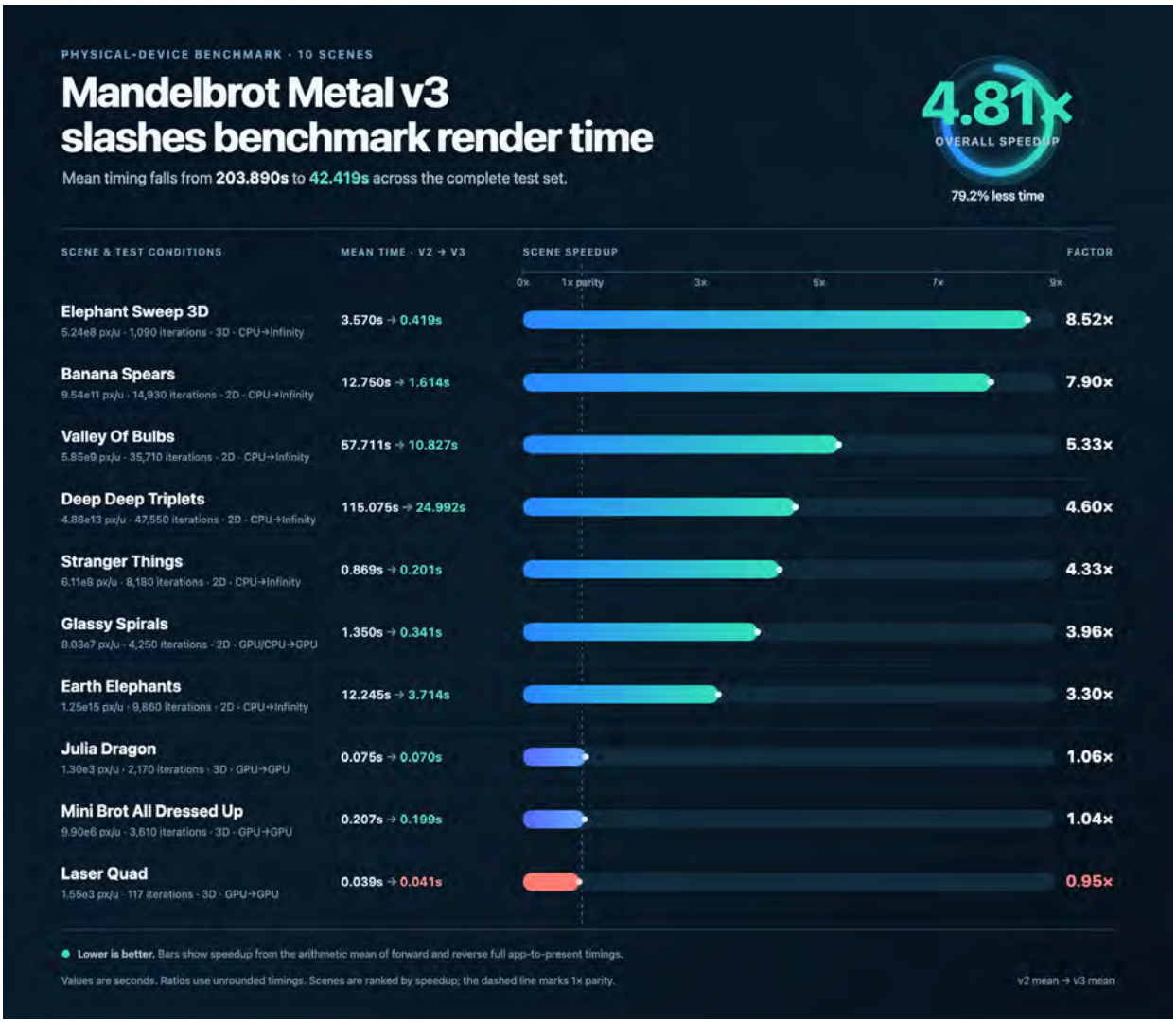

Figure 10 — Historical build 548 benchmarks. See this section for conditions and limits.

Results

The mean total time for all ten scenes was 203.890 s on v2.2.3 and 42.419 s on v3.0, a 4.81× aggregate speedup and a 79.2% reduction in elapsed time. Deep Deep Triplets improved from 115.075 s to 24.992 s (4.60×); Earth Elephants from 12.245 s to 3.714 s (3.30×); and Valley of Bulbs from 57.711 s to 10.827 s (5.33×). The low-zoom GPU controls were similar: speedup ratios were 1.06× for Julia Dragon, 1.04× for Mini Brot, and 0.95× for Laser Quad, which took roughly two milliseconds longer. Build 548's bounded scheduler eliminated the status error seen in a rejected monolithic reverse-order development pass and kept the final forward and reverse deep routes entirely on the Infinity path.

Interpretation and limits

Order mattered most on the legacy CPU path: v2.2.3 Deep Deep Triplets ranged from 96.987 to 133.163 s, while v3 ranged from 24.919 to 25.066 s. Valley of Bulbs showed the largest v3 spread (9.491–12.164 s), consistent with thermal and scene-dependent GPU cost, though these runs do not isolate the cause. These measurements apply to build 548, this device/OS, one sample per Infinity pixel, High Quality Idle Off, and the exported corpus. They do not establish performance on other devices, exports, deep Julia views, or arbitrary-precision reference backends.

16. Known Limits

- Finite $N_{\max}$ cannot prove membership for arbitrary boundary samples.
- Infinity currently applies to Mandelbrot mode, not Julia mode.
- The Julia parameter is stored as Double but quantized to float2 for the current Metal kernel.
- QS escape and glitch diagnostics reduce expanded values to Float.
- Infinity currently requests one sample per pixel; multi-sample qualification remains separate work.
- The $10^{25}\times$ release ceiling does not guarantee equal detail across all locations.
- BGRA8Unorm output and LUTs do not establish a scene-linear or HDR pipeline.
- State restoration is not a bit-identical cross-device render manifest.
- Scene documents and motion/video export are future capabilities, not current v3.0 deliverables.

The renderer handles scale and path limits explicitly. Public v3.0 scale requests above 10^{25} are clamped consistently before state is committed; unusable in-range steps select the established fallback instead of inventing coordinates. A valid completed slow Metal frame is retained and reported; only a failed or unusable command opens the CPU circuit. Entitlement verification follows the separate rules in Section 14.

17. Release-Series Engineering Direction

The roadmap in Table 1 identifies future work in reference-orbit acceleration, a deterministic creation system, broader access, and reliability qualification. These are engineering directions rather than dated release commitments. Build 570 contains experimental extended-range coordinate and reference machinery, but the shared public scale policy prevents navigation beyond 10^{25}. Future releases need independent correctness and device qualification before those paths become product features.

After the validated 10^{25} Version 3.0 boundary is stable across devices, research may resume toward 10^{100}. Reaching that target reliably requires more than raising the scale limit. It requires a higher-precision canonical coordinate and reference-orbit backend, reference selection and rebasing that remain stable across long navigation sequences, bounded GPU work, portable state serialization, and a new image-correctness and physical-device qualification corpus. Until those gates pass, zooming up to 10^{100} remains a development objective, not a product specification.

Feature entitlement must preserve the correctness and speed of the shared rendering engine. Free and Pro may differ in tools, limits, workflows, assets, and export options, but every available path must meet the same rendering-correctness requirements. Where limits differ, the UI and persisted document model should make those limits explicit.

Appendix A — Infinity Pseudocode

```
buildReferenceOrbit(center c0, maxIterations):
    Z = DD(0)
    for n in 0 ..< maxIterations:
        Z = DD.square(Z) + DD(c0)
        upload[n] = (floatExpansion4(Z.real), floatExpansion4(Z.imag))
        if DD.normSquared(Z) > 4: break

renderPixel(sample p):
    deltaC = QS(referenceRelativeOrigin + splitStep * p)
    result = perturb(referenceOrbit, deltaC)
    if result == invalid:
        result = directQS(referenceAnchor + deltaC)
    return shade(smoothEscape(result))
```

Appendix B — Entitlement Pseudocode

```
resolved = verifiedLegacyOwner ? legacyPro : free
for transaction in Transaction.currentEntitlements:
    require verified && !revoked && !expired
    candidate = knownProductEntitlement(transaction.productID)
    resolved = maxByRank(resolved, candidate)
publish resolved
```

Entitlement rank is an implementation convenience for selecting a single display state when multiple verified products coexist. Any non-free state grants Pro. The rank does not imply that an annual subscription is intrinsically more capable than a monthly one; they resolve to the same access Boolean.

Appendix C — Symbols and Terms

Symbol / term	Meaning
c	Complex parameter of the quadratic map
z_n	Actual orbit value at iteration n
c_0	Reference parameter used to build the high-precision orbit
Z_n	Reference orbit value at iteration n
δc	Pixel parameter offset $c - c_0$
δz_n	Orbit offset $z_n - Z_n$
$N_{\max}$	Effective finite iteration budget for a render; distinct from a proposed target or the selectable cap
S	Scale in pixels per complex-plane unit; UI Auto Iterations uses points instead (Section 3.2).
DS	Two-Float compensated expansion
DD	Two-Double compensated expansion
QS	Quad-single software expansion of four Float limbs used by Infinity in Metal
SSAA	Supersampling anti-aliasing within one output thread
FMA (fused multiply-add)	Multiplies two values and adds a third as one operation with a single final rounding, improving numerical accuracy.
Metal	Apple's programming framework for running graphics and general-purpose computations directly on the GPU.

Symbol / term	Meaning
BGRA8Unorm	A 32-bit pixel format containing 8-bit blue, green, red, and alpha channels represented as normalized values from 0 to 1.
Perturbation	A deep-zoom technique that computes each pixel's small deviation from a precomputed reference orbit instead of repeating the entire orbit at high precision.
Reference orbit	A high-precision orbit calculated once for a selected reference point and reused by nearby pixels during perturbation.
Rebasing	In this implementation, setting the perturbation state to the actual orbit value and resetting the reference index to zero, without rebuilding the CPU orbit.
Bailout	A magnitude threshold whose crossing establishes divergence in exact arithmetic for the applicable parameter domain.
Critical orbit	The sequence generated by repeatedly applying the quadratic map beginning at z0 = 0.
Escape-time rendering	A fractal-rendering method that colors each sample according to whether, and how quickly, its orbit escapes.
Smooth escape coordinate	A fractional escape value that reduces visible color bands between integer iteration counts.
Iteration budget	The maximum number of recurrence steps evaluated before a sample is classified as non-escaping within the current limit.
Finite precision	Arithmetic performed with a fixed number of representable digits and therefore subject to rounding and range limits.
Floating-point expansion	A number represented as the unevaluated sum of multiple floating-point components to retain additional precision.
Limb	One ordered component of a floating-point expansion, with successive limbs carrying progressively smaller contributions.
Residual	The small error remaining after a rounded arithmetic result, retained to improve the accuracy of an expansion.
Cancellation	Loss of significant digits when subtracting or combining nearly equal values.
Representability	Whether a numeric value can be stored distinctly and finitely in the selected arithmetic format.
Compute kernel	A GPU function executed concurrently across many threads to perform the per-pixel rendering work.
LUT (lookup table)	A sampled array of colors that converts a normalized escape value into the rendered pixel color.
Linear filtering	Texture sampling that interpolates between neighboring texels to produce a smooth intermediate value.
Clamp-to-edge	Texture addressing that replaces coordinates outside the valid range with the nearest edge coordinate.
Command buffer	A submitted collection of GPU operations that Metal schedules and executes as a unit.
Blit	A GPU operation that copies or transfers image or buffer data without recomputing its contents.
ABI (application binary interface)	The required memory layout and calling contract shared by independently compiled code, such as Swift and Metal.
Hysteresis	The use of different entry and exit thresholds to prevent rapid switching between renderer states.
Scene-linear color	Color values proportional to physical light intensity rather than encoded for display.
sRGB	A standard color space and transfer curve widely used for screens and web imagery.
Display P3	A wide-gamut color space capable of representing more saturated colors than sRGB.
HDR (high dynamic range)	Imaging that represents a wider brightness and color range than standard dynamic-range output.
StoreKit 2	Apple's framework for processing App Store products, purchases, subscriptions, and verified ownership.

Symbol / term	Meaning
Entitlement	A verified ownership state that grants access to a feature or product tier.
Legacy Pro	Permanent Pro entitlement for verified pre-v3 owners.

Table 13 — Symbols and terms used throughout the white paper.

Appendix D — Notices

Mandelbrot Metal is developed for iPhone and iPad and uses Apple Metal, StoreKit 2, Swift, SwiftUI, UIKit interoperability, and system image/export frameworks. Mathematical descriptions are implementation documentation, not a claim to have discovered the underlying results. This white paper describes the inspected v3 development snapshot and may be revised as individual releases pass qualification.

Copyright

DBA

Mandelbrot Metal is a registered DBA (Doing Business As) of Michael Stebel Consulting, LLC, a Florida limited liability company.

Trademarks