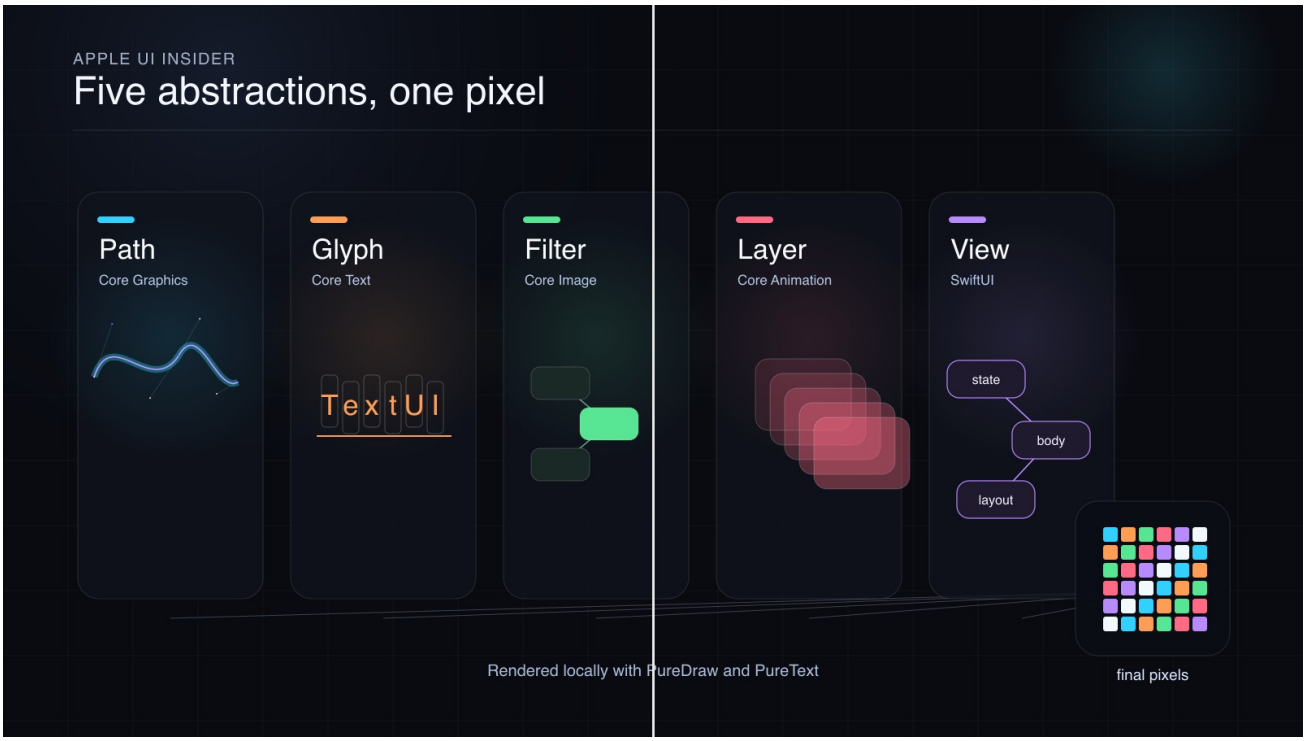


Introducing Apple UI Insider

I just shipped the first issue of a new newsletter: Apple UI Insider. It is a field guide to Apple's UI rendering stack, Core Graphics, Core Text, Core Image, Core Animation, and SwiftUI, written from the inside of a clean-room renderer I built to match it.

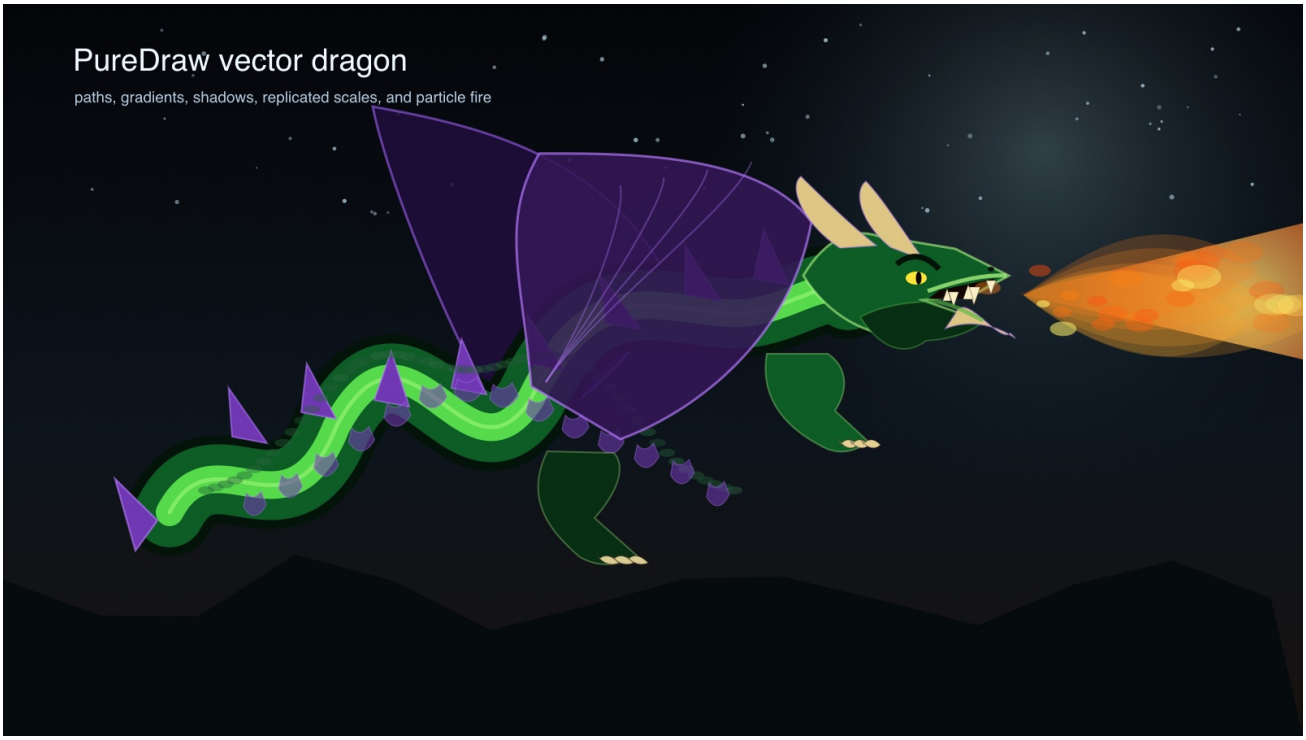
What each issue does. Every issue picks a corner of the stack and asks what each tier actually owns, where the boundary between two tiers runs, and which bugs live there specifically because of that boundary. Not "how to use SwiftUI." Where SwiftUI stops owning the pixels and Core Animation starts. Core Animation does not shape text. Core Text does not composite your window. Core Graphics does not know SwiftUI identity. A surprising number of confusing bugs come from expecting one tier to quietly do another tier's job.

Every picture is drawn twice. That claim needs a way to check it, so each figure in the newsletter is rendered twice from the identical scene source: once by SlayerMotion, my clean-room renderer, and once by Apple's own frameworks. The two outputs are compared and the differences are published alongside the picture, not hidden in a footnote.

I don't just assert that the comparison is real, I make it falsifiable. Issue 1 ships a mutation proof: take the hero scene's source, change exactly one fill to pure magenta, and re-run it through Apple's own pipeline. The untouched render contains zero magenta pixels. The mutated one contains exactly 3,600, in exactly the region I changed, a 60 by 60 pixel patch and nothing else. If the twin renderer were faking agreement instead of actually reproducing Apple's output, that number would be wrong.

That same discipline is what caught a real bug while I was building the issue. An early version of the animation twins showed the two renderers disagreeing on rotated layers by 3.7 to 3.9 percent, against a healthy baseline of about 0.1 percent for every other scene. That gap traced to a coordinate-flip bug in my own engine's device-space compositing, not a rounding difference and not Apple's fault. I fixed it, re-ran the exact same comparison, and the same scene now agrees to within 0.07 percent. The original, divergent renders are still in the issue's proof bundle next to the fixed ones, because a caught bug is evidence too, not something to quietly delete.

Not every scene in the issue is a diagram. One of them is a vector dragon breathing particle fire, drawn entirely through the same graphics tier as everything else, paths, gradients, shadows, and replicated scales down its tail, then driven frame by frame into an animated GIF.



Same discipline, less diagram. The dragon is one of the issue's Core Animation scenes, rendered through Core Graphics and compared frame by frame.

Most of the twin pairs in the issue, dragon included, agree closely enough that you cannot tell the two renders apart by eye, which is the entire point, and why the measured percentage next to each figure is doing the real work rather than the picture alone.

I'd rather name the current edge than round up. Issue 1's Apple twin is Core Graphics for every scene, so it compares rasterizers on identical, already-lowered geometry. It is not yet comparing my text engine against real Core Text shaping, or my image engine against a live `CIContext`, or my layer engine against an actual `CALayer` tree animating on screen. Those framework-native comparisons are exactly what the next issues build, one claim at a time, with the same measure-and-publish discipline, not a bigger promise made once and never checked again.

Issue 1: The Map, Drawn Twice is live now. Read it, or subscribe to get the next one in your inbox first.