MapLibre Native Metal Status Report: July 2023

This is the fifth status report for the MapLibre Native Metal project. For a deeper dive into the project's background, consult the first report.

Overall the status is **Green.** We are on track for a Metal implementation by the end of Q3, and to finish the migration in December.

Background

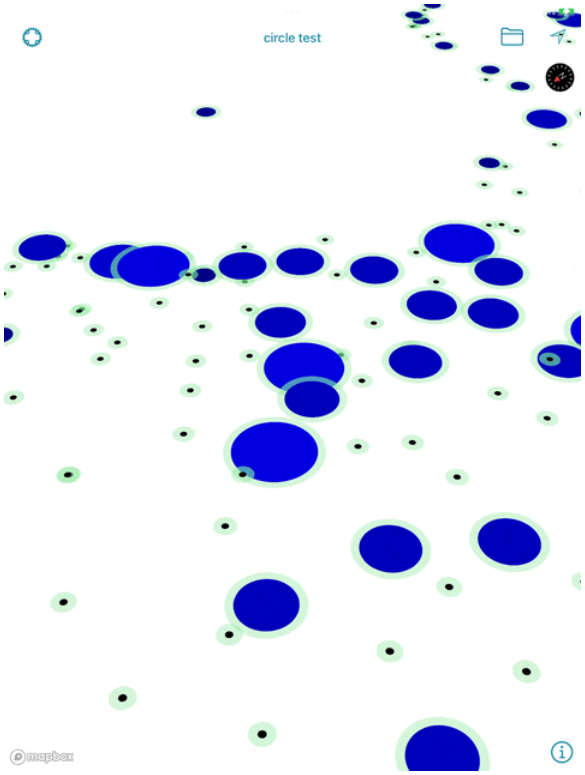
As a quick reminder, we're updating the MapLibre Project to support Metal for iOS and generally modernize the renderer.

We are still wrapping up the first phase, Renderer Modularization, but only the community merge remains. The team is well into the second phase, the Metal Port. We'll dig into the status of the first and show some good progress on the second.

Status

We're **green** and have lost no additional time.

The theme for July's report is "Finally! We caught a break!"



Here's the circle layer working in Metal. Did you even know there was a circle layer? I tend to forget about it, but it makes a nice development focus.

I can't emphasize enough that getting any layer to work in Metal is a big achievement for the project. This is great, great news.

For once we have no updates to the schedule. Everything seems to be on track for a Q3 implementation on Metal.

Mega Merge

There is a lot more discussion on the merge of the Phase 1 results in the last report. As a quick overview, here are the steps:

1. Architecture
   We merge in entirely new files and concepts.

2. Rendering Infrastructure Changes
   Changes to the core rendering modules to support the new renderer.

3-6. Layers, Layers, Layers
   Raster, Fill, Line, Symbol, individually or in groups

7. Activate New Rendering Path
   Up to this point we've had the old rendering path turned on. There are #ifdefs in the files for this. This is when we turn on the new renderer by default.

8. Remove Old Renderer
   A while later (August would be my guess) we completely remove the old renderer.

We've finished (1) and will shortly do (2). These aren't time critical or holding up development.

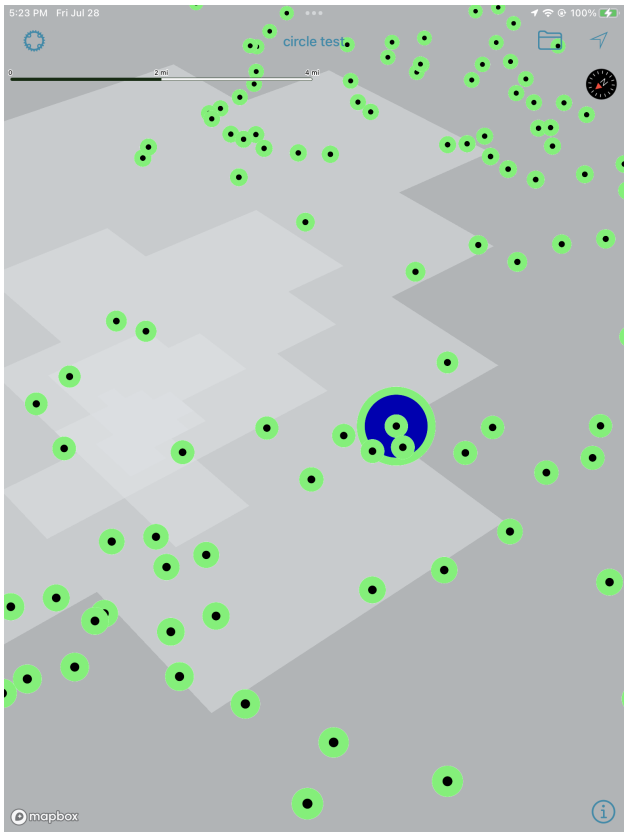
The Layers phases have been squished down to two phases to speed things up a bit. We have two formal technical reviewers, one from Grab and one from TomTom as well as Bart from the consortium. Otherwise, we're not getting a ton of feedback from the community, so we're going to skip a few steps. More on that later.

Activating the New Rendering Path is the big one and we need all the tests to pass before we can do that. That's looking very likely next week.

We should have the merge wrapped up within two weeks.

Wins

The big win this month was Metal output. The core concepts are working in Metal!



They also took less work to get going than anticipated. Getting the Drawable ported over took less time than we feared, suggesting that porting to new SDKs in the future may be easier.

The render tests are working again. These are the conformance tests designed for map display from OpenGL. From late last week:

```
1177 passed (89.7%)
9 passed but were ignored (0.7%)
81 ignored (6.2%)
45 failed (3.4%)
```

Test output always involves a bit of discussion. We're focused on that last line, the 3.4% failing. The *ignored* category are tests which didn't work before we started. It's a big test suite that the maintainers have recently brought back on line and it's still got some dark corners.

But that is a lot of render tests and we're down to three or so core causes, which we're addressing right now. These are proving more tractable than expected in the last report.

Challenges

The biggest remaining problem from Phase 1 is toolkit size. The compiled size of the binary toolkit seems to be up 1.5% from when we started.

We'll need to go back and revisit size. But as I've said to the team, I can explain a toolkit that works on Metal but is a little too big, much easier than I can explain a small toolkit that doesn't work on Metal.

Insights

Right now when you zoom in and the opacity of a lake changes, math is being done on the CPU. It should really be done on the GPU. How we were going to get there has been something of a mystery for me until the other day.

We were working through some input data logic for the new shaders and sketched out how to pass that information through. The problem with the CPU -> GPU transition has been doing it incrementally. We'll want to carve off a few cases and implement them one by one. What we came up with will allow for that and I can see that path more clearly.

Risks

As mentioned in the last report, the secondary goals continue to be at risk. Those were the modernization of the renderer, threading, toolkit size and speed.

Some of that is beginning to come clear and I'm less worried about the Metal side of things. Metal is almost certainly going to be faster and more able to thread. It's also clear we'll be able to move some of the data driving styling to the GPU… for Metal.

The OpenGL version I worry about. It may end up larger and slower than we'd like. As mentioned before, we're up 1.5% in size. Conversely, we may have to spend precious time dealing with that in Q4. We'll know more as the month proceeds.

If we have to confront a render speed problem I'd rather do so by using the GPU, even with OpenGL, but we may be constrained there.

Q3 Goals

Last time we put out a few Q3 goals. Let's revisit those and see where we're at.

- **Finish the Mega Merge**
  Another week or two.

- **Direct Version of Metal Port**
  Definitely on task for this one.

- **Equal Performance**
  We shall see, but quite likely.

- **Try it out with a big user**
  Let's get a few screenshots of it working and then begin engaging with the users in late August.

It turns out that Q3 ends in September. I had a few days where I was wondering why I agreed to a working Metal port at the end of August. Who doesn't want an extra month!?

On the subject of when Metal is 'working', we're not looking for every rendering test to pass. We're just looking for a credible implementation that you might want to try using. We'll get to the full rendering test suite, but if it looks right, that's enough to start using it.