MOBILE VECTOR CARTOGRAPHY

Designing For Infinite Scale Factors • NACIS 2014

Justin Miller • Mapbox • @incanus77

I'm very glad to be here. Like Ken Kato, I'm here from Oregon, but also a Pittsburgh native! Thanks for having me.

INTRODUCTION

- Engineering-focused talk (though from a carto perspective)
- Mapbox is building tools for designers & developers
  - Enabling more cartographers
  - Programming/design/combination
- As much about what we're building as what we've learned

This will be an engineering-heavy talk, relatively, but I hope it's useful for cartographers. I'd like to share what goes into some of the tools we make for people making maps.

THE JOURNEY SO FAR

- Raster-based ***tile*** design with tools like TileMill
  - Deterministic (at least in principle)
  - Technically limiting for large geographic focus
    - Quantity of tiles (generally need 1:1 scale-to-tile layer parity)
    - Inability to change cartography on the fly
    - Limited situational flexibility (rotation, localization, animation)

We're starting to move from design of tiles themselves, where the final, delivered product is rasterized imagery.

THE JOURNEY (CONT'D)

- Moving to vector-based ***style*** design with Mapbox Studio & GL
- Goal: still deterministic design (again, in principle)
- But offers new benefits
  - Lower number of tiles through overzooming of vector features
  - Ability to change style 60 times/second (FPS)
  - Rotation, label changes, interpolated style transitions all possible

We're moving to style design, where styles and data tiles are compiled to be rendered on the server. I say "in principle" because for the worldwide base maps that we're enabling, no one checks every corner of the globe. They need things to render predictably.

VECTOR…?

- Can refer to either or both sides of the stack
- Vector data: point & polyline features essentially as data on a grid
  - However, can still be rasterized into tiles, then served
- Vector rendering: live drawing of features on the client side
  - Every viewport change is an opportunity for re-render

We're mostly talking about vector rendering here, by way of converting data into a portable, compact, pre-tiled vector data format.

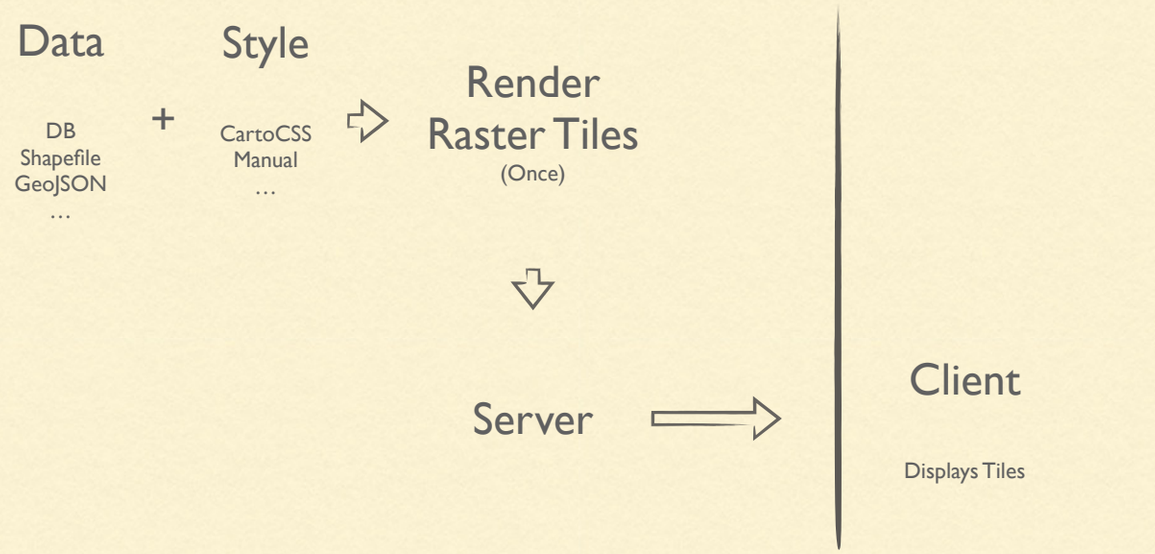
VECTOR RENDERING

- Approaches: in software (e.g. SVG) or in hardware (e.g. OpenGL)
- We went with OpenGL (hardware accelerated)
  - Based on a 22 year old API used for 3D visualizations & games
  - OpenGL ES (Embedded Systems) on mobile
  - Highly parallel & rapid processing of pixels on screen

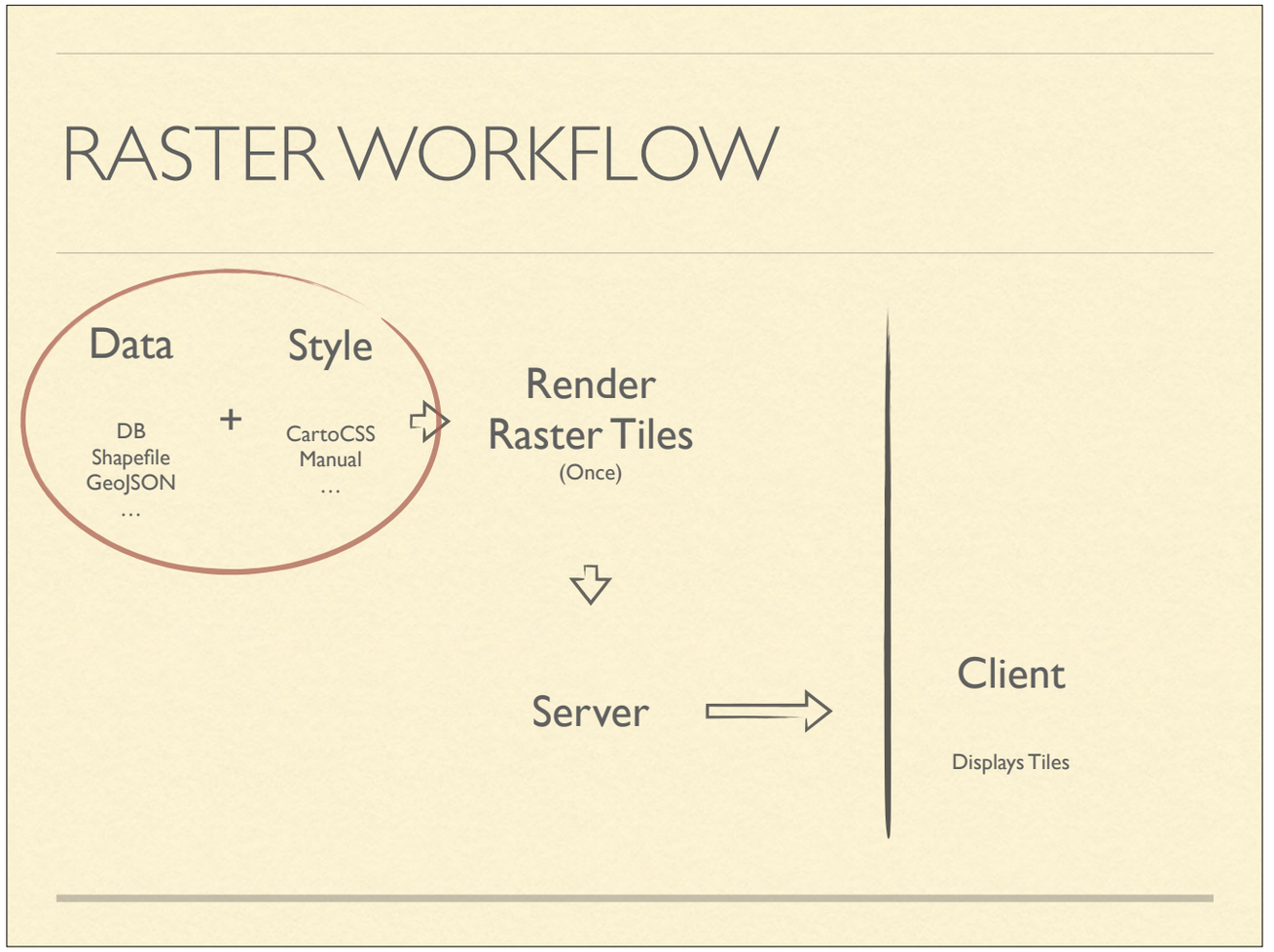
OPENGL TRADEOFFS

- Makes use of dedicated hardware (GPU)
  - Very good at its subset of capabilities
- Drawing work is offloaded from the CPU
  - Freed up for other tasks like network & disk activity
- But much more primitive & laborious (luckily, we do that part)
  - Bridges styling language to primitives like points & triangles

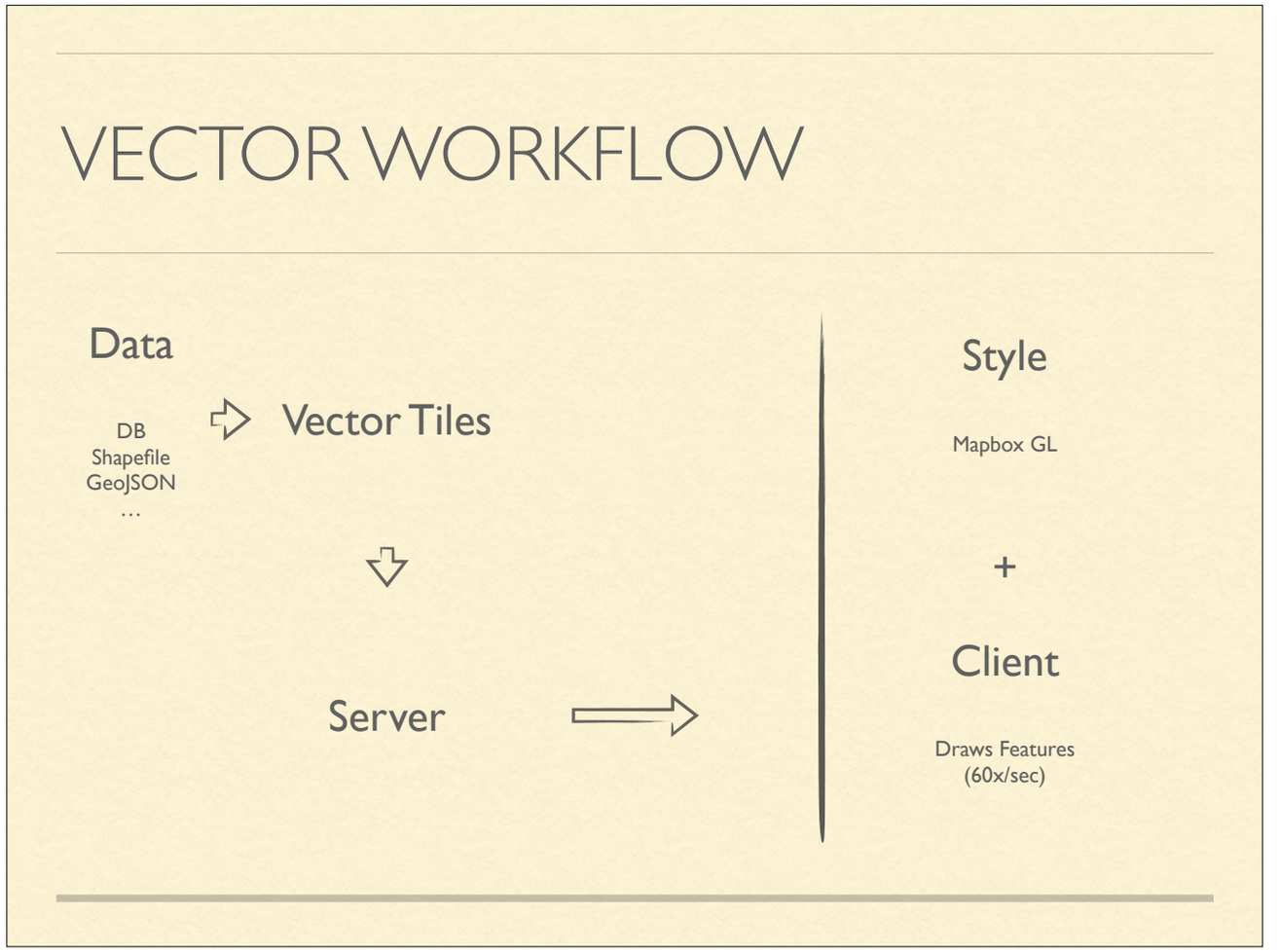
RASTER WORKFLOW



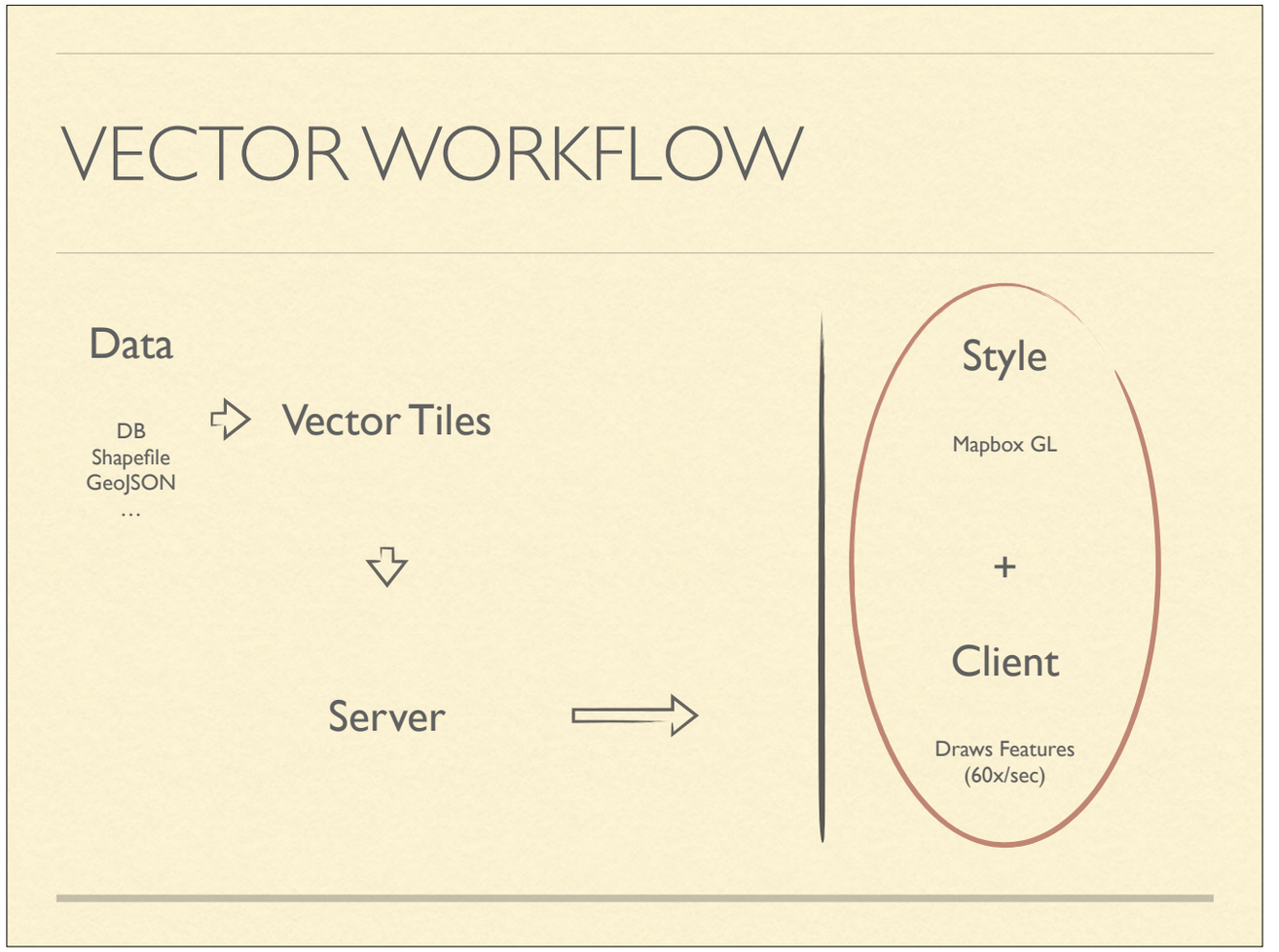
A wall is put up between cartographic control and the mobile device where the map is shown and used. The styling step happens on the left side of this wall.



A wall is put up between cartographic control and the mobile device where the map is shown and used. The styling step happens on the left side of this wall.

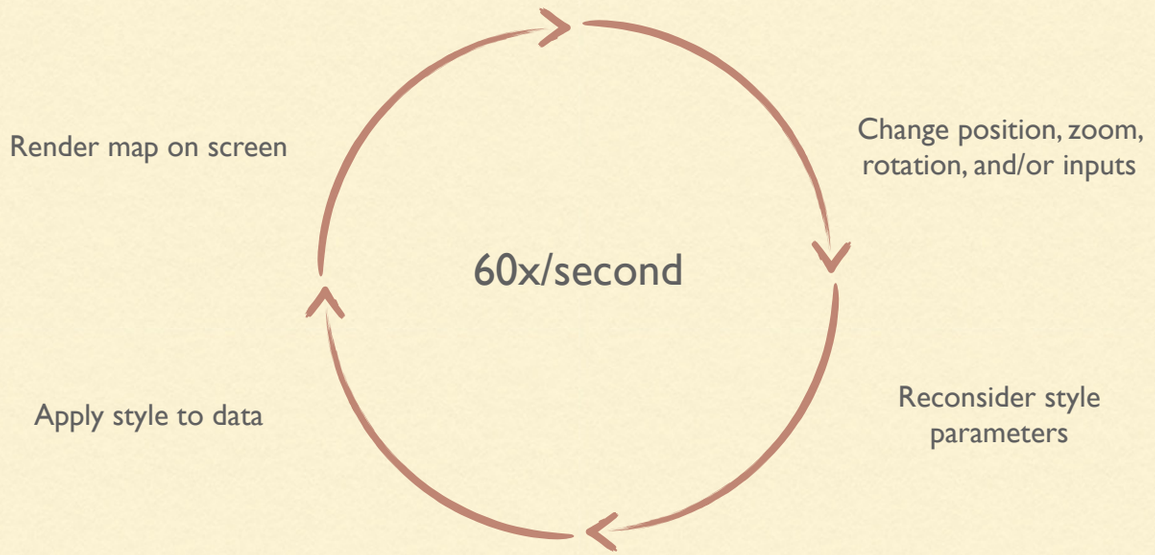


With vector rendering, opportunities exist on the right side of that wall to continue to change and respond to context when styling the map.

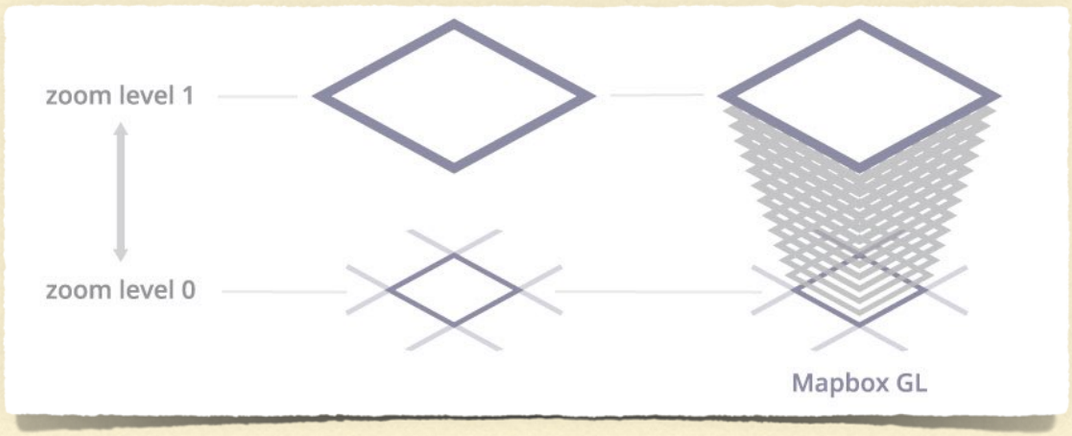


With vector rendering, opportunities exist on the right side of that wall to continue to change and respond to context when styling the map.

THE CIRCLE OF LIFE

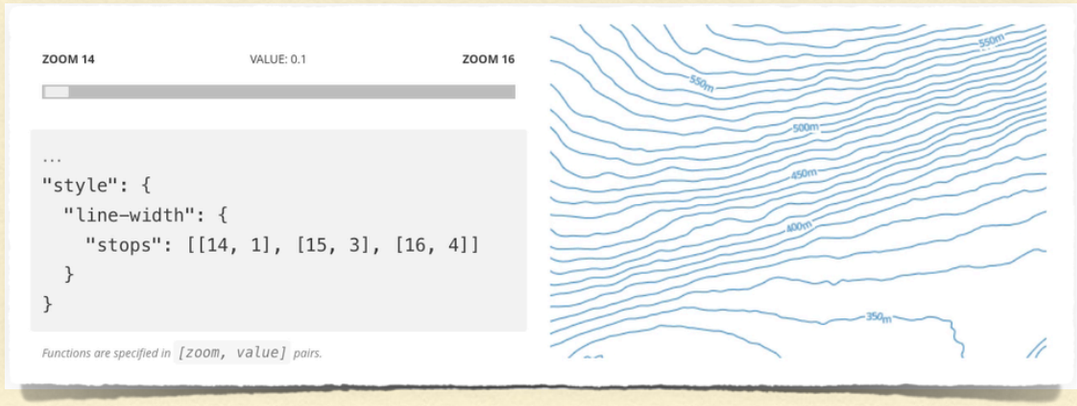


INCREMENTAL ZOOM



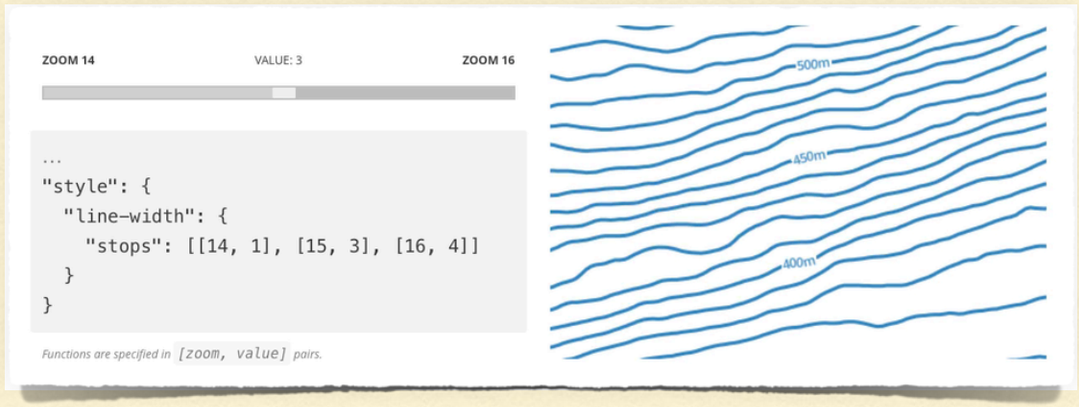
There are now incremental zoom levels, where every minor gesture creates a slightly new map scale that can be rendered uniquely.

ZOOM-DEPENDENT STYLING



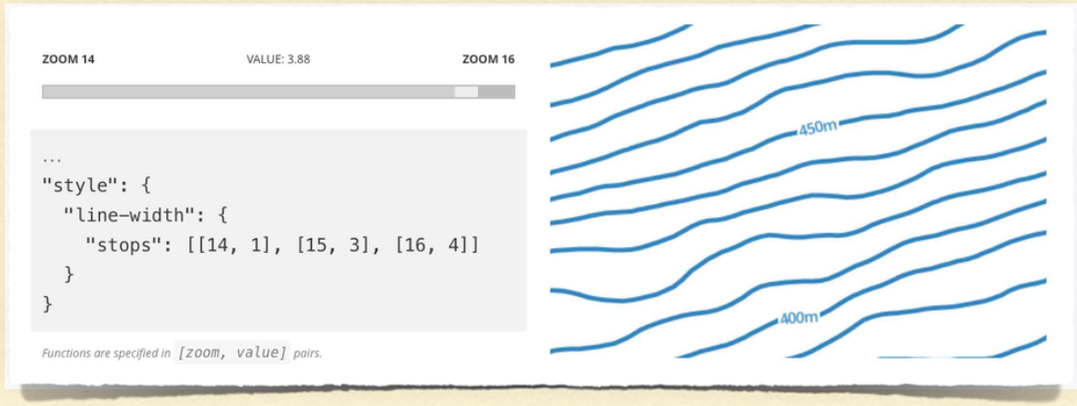
In this example, line width is interpolated along a series of “stops” which describe pixel width against the current zoom level.

ZOOM-DEPENDENT STYLING



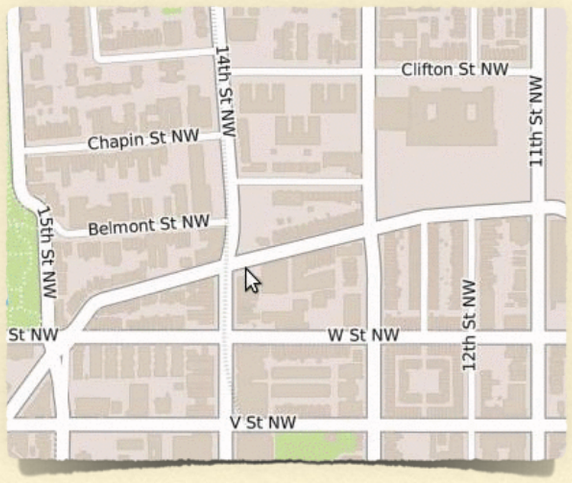
In this example, line width is interpolated along a series of “stops” which describe pixel width against the current zoom level.

ZOOM-DEPENDENT STYLING



In this example, line width is interpolated along a series of “stops” which describe pixel width against the current zoom level.

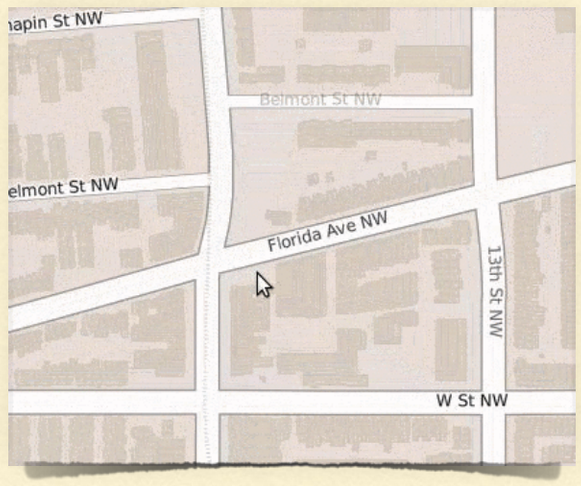
ZOOMING LABEL COLLISION



K. Been, E. Daiches, and C. Yap. *Dynamic map labeling.*, IEEE VGTC, Vol. 12, No. 5, 2006

Here we show how zooming label collision plays out.

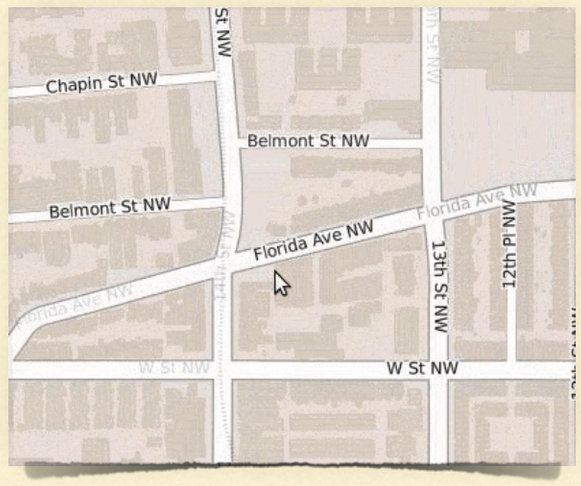
ZOOMING LABEL COLLISION



K. Been, E. Daiches, and C. Yap. *Dynamic map labeling.*, IEEE VGTC, Vol. 12, No. 5, 2006

Here we show how zooming label collision plays out.

ZOOMING LABEL COLLISION



K. Been, E. Daiches, and C. Yap. *Dynamic map labeling.*, IEEE VGTC, Vol. 12, No. 5, 2006

Here we show how zooming label collision plays out.

ROTATION LABEL COLLISION

A. Gemsa, M. Nöllenburg, and I. Rutter. *Consistent labeling of rotated maps*., arXiv:1104.5634, 2011

As well as rotation producing label collisions.

ROTATION LABEL COLLISION

A. Gemsa, M. Nöllenburg, and I. Rutter. *Consistent labeling of rotated maps*., arXiv:1104.5634, 2011

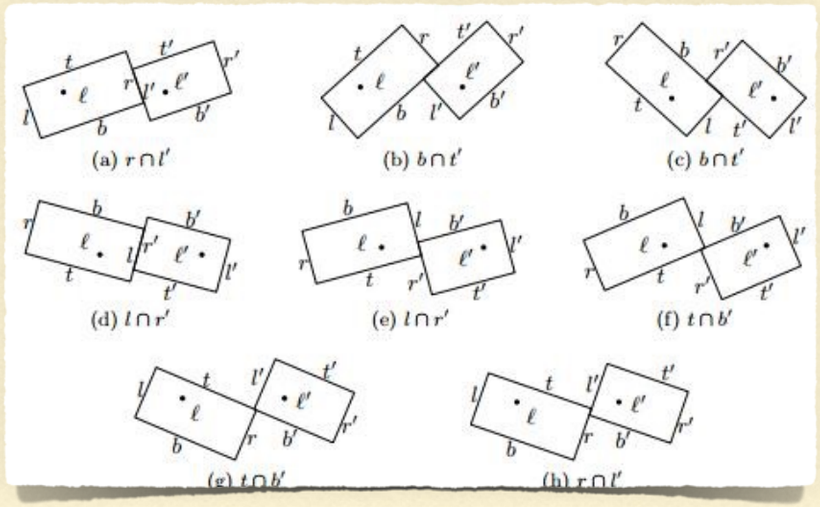
As well as rotation producing label collisions.

ROTATION LABEL COLLISION

A. Gemsa, M. Nöllenburg, and I. Rutter. *Consistent labeling of rotated maps*., arXiv:1104.5634, 2011

As well as rotation producing label collisions.

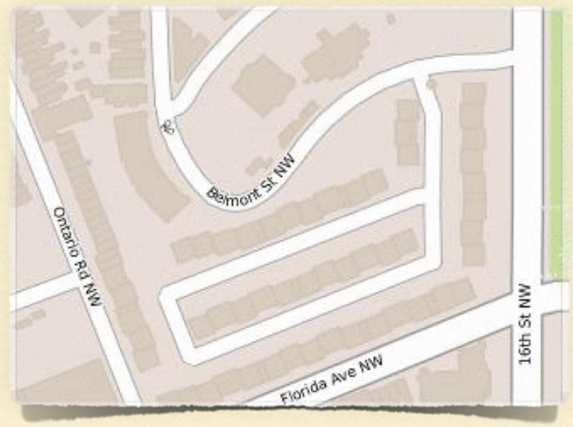
LABEL COLLISION SITUATIONS



A. Gemsa, M. Nöllenburg, and I. Rutter. *Consistent labeling of rotated maps*., arXiv:1104.5634, 2011

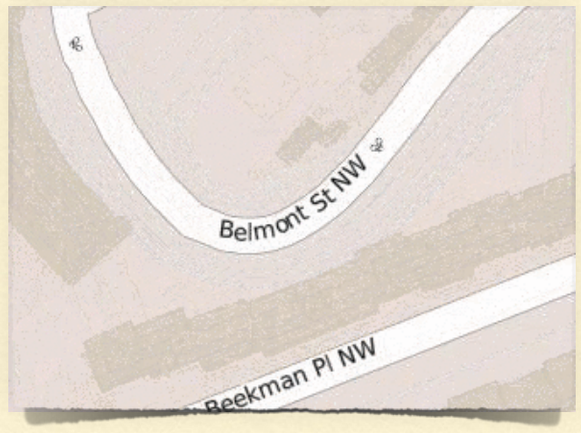
We used some computer graphics research to determine how to solve this collisions and always render legible and useful labels.

ZOOMING LABEL WRAP



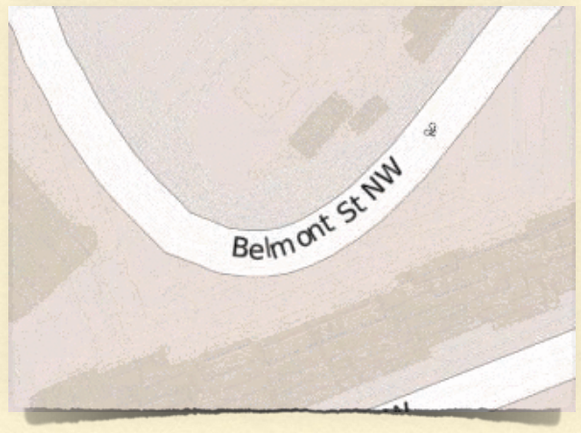
We’ve even gotten labels to wrap at angles which depend on the curvature of the map.

ZOOMING LABEL WRAP



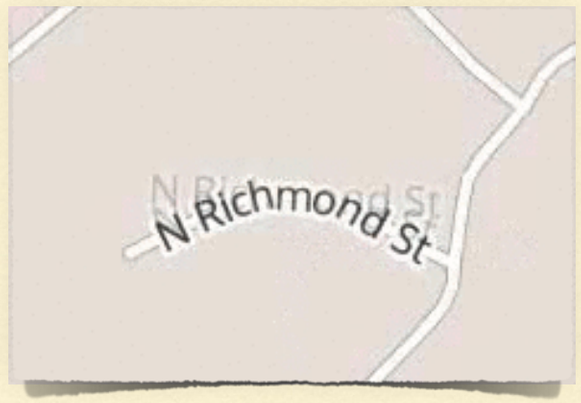
We’ve even gotten labels to wrap at angles which depend on the curvature of the map.

ZOOMING LABEL WRAP



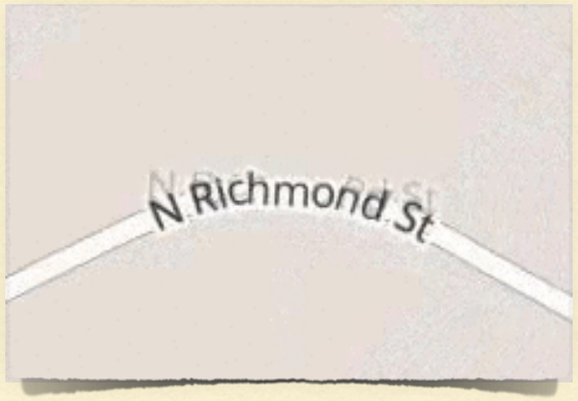
We’ve even gotten labels to wrap at angles which depend on the curvature of the map.

LABEL WRAP EXPOSED



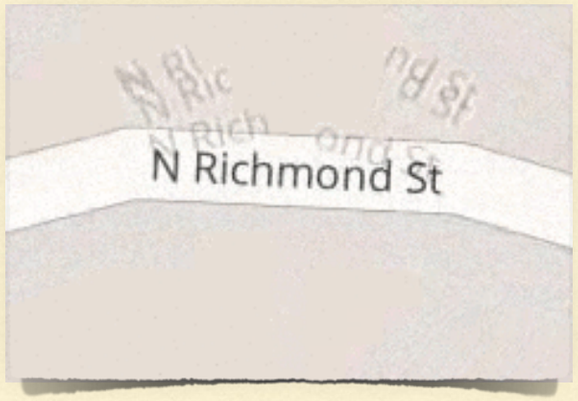
Here’s a look behind the scenes of how that is done.

LABEL WRAP EXPOSED

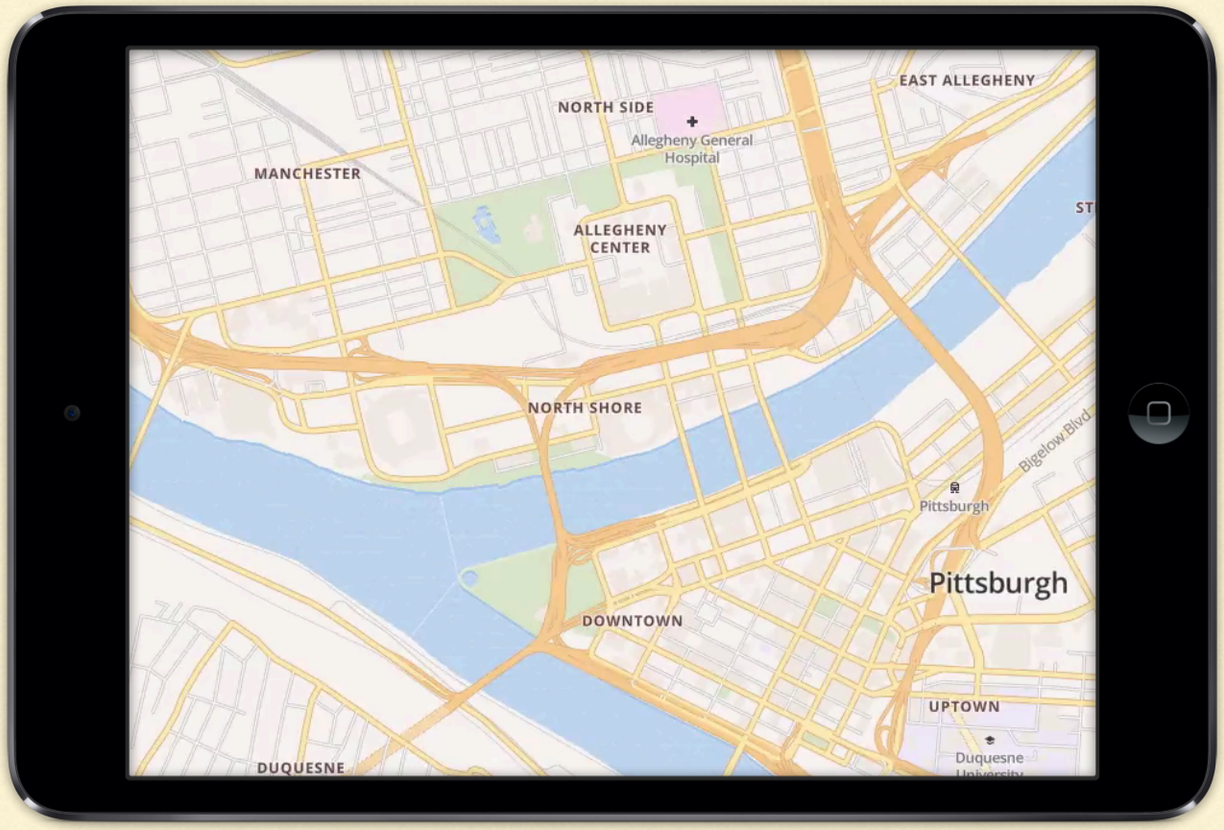


Here’s a look behind the scenes of how that is done.

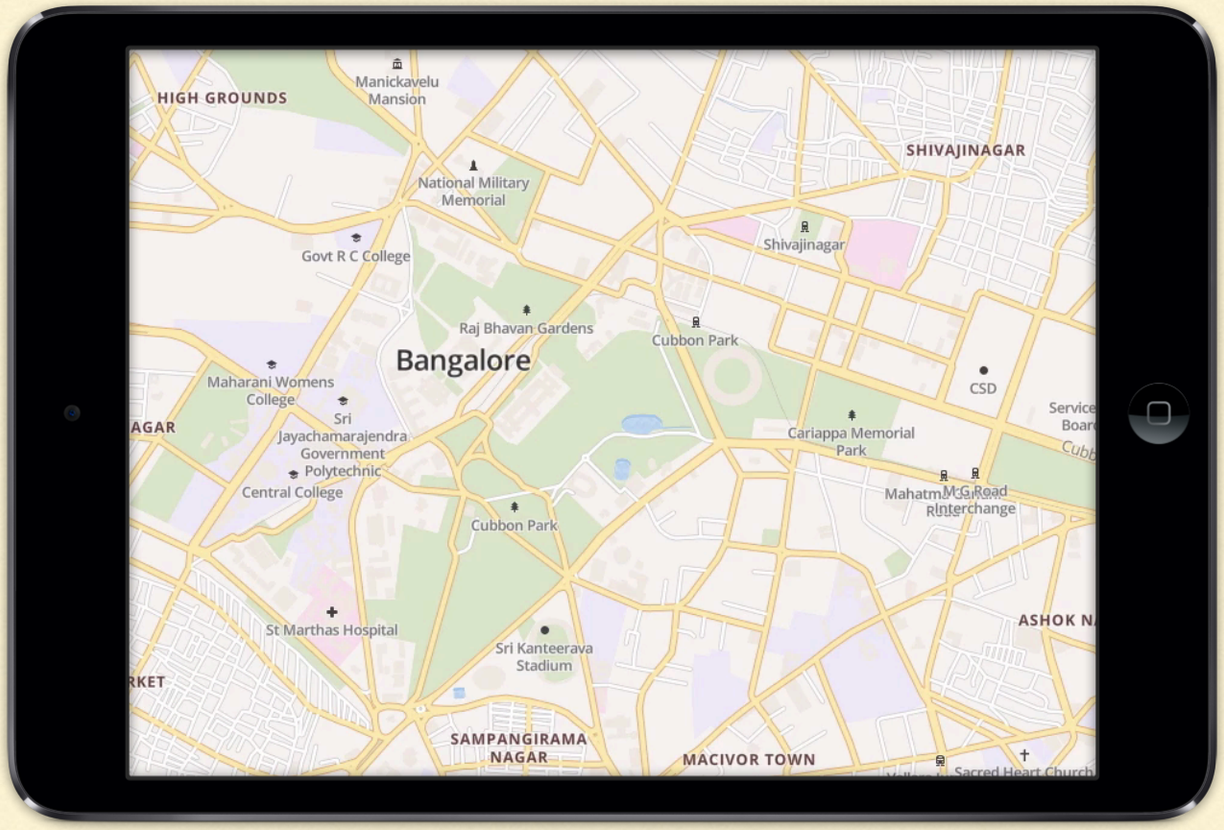
LABEL WRAP EXPOSED



Here’s a look behind the scenes of how that is done.



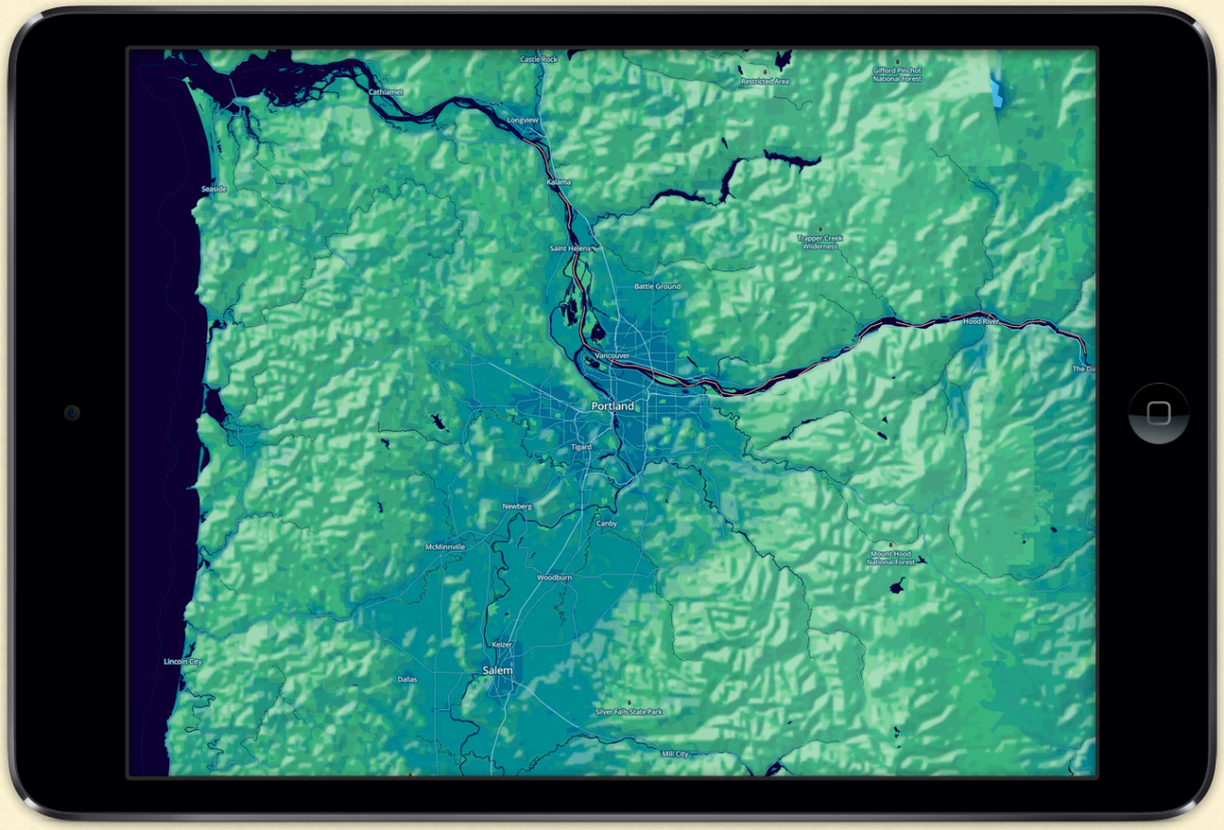
Labels should render the same in Pittsburgh, which you've planned for, as in Bangalore, which you may not have tested.



Labels should render the same in Pittsburgh, which you've planned for, as in Bangalore, which you may not have tested.

Here’s a contrived example of smoothly transitioning between two similar but also very different styles, day and night modes for a map. These transitions can happen at 60 frames per second.

Here’s a contrived example of smoothly transitioning between two similar but also very different styles, day and night modes for a map. These transitions can happen at 60 frames per second.



Here’s a contrived example of smoothly transitioning between two similar but also very different styles, day and night modes for a map. These transitions can happen at 60 frames per second.

Once you've got this foundation, even raster imagery can make use of it. Here is drone-collected aerial video imagery atop raster satellite tiles.

CONCLUSION

- We're building tools for the heavy lifting of hardware acceleration
- Combining pre-tiled vector data with flexible styling
- Oriented towards a more fluid, responsive map experience
- Hoping to create new potential for map design & interaction
- Open source & designed with interoperability in mind

THANK YOU

- justin@mapbox.com & @incanus77
- mapbox.com/mapbox-gl
- mapbox.com/blog
- github.com/mapbox