Mapbox Vector Tile Specification 2.0

Blake Thompson - Software Engineer, Mapbox

About Me

- Developer at OKC Mapbox Office
  - Mapnik
  - Node Mapnik
  - Mapnik Vector Tile
- Author of Mapbox Vector Tile Specification

Coffee during morning scrum

What are Vector Tiles

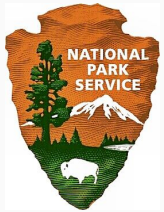
Binary format for encoding geometries and their metadata

- Points
- Lines
- Polygons

Structure of Vector Tiles

- Layer
  - Feature
    - Geometry

Lots of Companies Using Vector Tiles!



What I Will Cover

- Why use Vector Tiles?
- Why we made Version 2.0 of the Spec.
- What changed in 2.0?
- Common Misconceptions about Vector Tile Spec.
- Future of the Spec.

Why use Vector Tiles?

(besides all the cool features)

Your #1 Limiting Resource is $$$$$$$$

Cost of Making a Map

Steps To Make Map:

- Query
- Render
- Transfer

COST FORMULA:

R = Total Number of Requests (Work)

q = query

r = render

t = transfer

s = storage of data

$$\$ (Cost) = R (q + r + t + s)$$

WMS

- R (Requests) increases a lot if the map is interactive
- Query per request
- Rendered on demand
- Transfer is an image (t is large)
- **Storage must scale with R because databases must scale with requests**

Cost is linear but STEEP!

$$\$ = R (q + r + t + s)$$

Image Tile Rendering from Database

- Requests are reduced because caching can be done!
  - R_c = Requests with Caching
- Transfer is an image (t is large)
- **Storage must scale with R because databases must scale with requests**

Cost is linear and steep but caching can greatly reduce costs! Cheaper than WMS!

$$\$ = R_c\,(q + r + s) + R\,(t)$$

Serving Static Image Tiles

- Create all the images before serving!
  - S = Number of styles
- **Storage can become quite massive as you are storing both the source data and the image tiles!**
- Great for static datasets with high traffic
- Query and Rendering are now based on number of styles!

Cost is linear and almost constant if you have a very small number of styles and static data. However, initial cost is HIGHER.

$$\$ = R(t) + S(s + q + r)$$

Super Easy to Scale!

Extremely expensive if you need a lot of different map styles

Vector Tile Server Side Rendering

- Precut all the geometry into Vector Tiles and store!
- Storage (s) is smaller as VT take much less space than images
- Storage not repeated per style!
- Great for large number of different styles!

Cost increases based on requests, but much cheaper if you have a lot of styles of the same data!

$$\$ = R_c\,(r) + R\,(t) + (q + s)$$

Much smaller storage cost then storing images

Vector Tile Client Side (GL) Rendering

- Transfer size is smaller as you are no longer sending images!
- **No rendering cost as client's hardware does the rendering!**

$$\$ = R(t) + (q + s)$$

Very low cost!!!

Vector Tiles on the Client

- Query data on map
- Interactive maps
- Smooth Transition (no transition between zoom levels of data)
- Easy to still render tiles on the server side
- Very quick transfer of data to client!

Why Vector Tile 2.0?

Make it a “Real” Specification

- Version 1.0 left many things unexplained
- Implementations were simply based off Mapnik Vector Tile library

Vector Tile Specification 1.0 fits in one slide

Vector Tiles use Google Protocol buffers as a container format. It is exclusively geared towards square pixel tiles in Spherical Mercator projection.

A vector tile can consist of one or more named layers and containing one or more features.

Features contain an id, attributes, and geometries: either point, linestring, or polygon.

Geometries are stored as an a single array of integers that represent an command,x,y stream (where command is a rendering command like move_to or line_to). Commands are encoded only when they change. Geometries are clipped, reprojected into spherical mercator, converted to screen coordinates, and delta and zigzag encoded.

Feature attributes are encoded as key:value pairs which are dictionary encoded at the layer level for compact storage of any repeated keys or values. Values use variant type encoding supporting both unicode strings, boolean values, and various integer and floating point types.

```
// Contains a stream of commands and parameters (vertices). The
// repeat count is shifted to the left by 3 bits. This means
// that the command has 3 bits (0-7). The repeat count
// indicates how often this command is to be repeated. Defined
// commands are:
// - MoveTo:    1   (2 parameters follow)
// - LineTo:    2   (2 parameters follow)
// - ClosePath: 7   (no parameters follow)
//
// Commands are encoded as uint32 varints. Vertex parameters
// are encoded as deltas to the previous position and, as they
// may be negative, are further "zigzag" encoded as unsigned
// 32-bit ints:
//
//   n = (n << 1) ^ (n >> 31)
//
// Ex.: MoveTo(3, 6), LineTo(8, 12), LineTo(20, 34), ClosePath
// Encoded as: [ 9 6 12 18 10 12 24 44 15 ]
//               |       |             `> [00001 111] command type 7 (ClosePath), length 1
//               |       |       ===== relative LineTo(+12, +22) == LineTo(20, 34)
//               |       | ===== relative LineTo(+5, +6) == LineTo(8, 12)
//               |       `> [00010 010] = command type 2 (LineTo), length 2
//               | ==== relative MoveTo(+3, +6)
//              `> [00001 001] = command type 1 (MoveTo), length 1
//
// The original position is (0,0).
repeated uint32 geometry = 4 [ packed = true ];
```

Community Direction

- Vector Tiles should be useable across different vendors
- Community should participate in the future of the specification

Friendlier Data

- Invalid geometry data complicates display and analysis
  - Self Intersections
  - Self tangency
  - Repeated points
- No repeated layer names
- No repeating feature IDs

A Process for Changing Specification

- An understood process for changing the specification
- Definitive way to Version specification

What Changed in 2.0?

Compatibility

- Version 1.0 decoders will be able to decode 2.0 tiles
- Version 1.0 tiles should be able to be converted to 2.0 tiles
  - Correct Geometry
  - Remove “Empty” Layers and Features

Identify Vector Tiles

- File Extension is “.mvt”
  - “.vector.pbf” is not descriptive
  - Protobuffers requires a knowledge of data structure
- MIME Type
  - application/vnd.mapbox-vector-tile
  - Registered with IANA

Projections

- Each Vector Tile has its own “projection”
  - Data in a coordinate system relative to the top left corner of the vector tile.
- Clarified that data of almost **any** projection could be put into a vector tile
  - Most data still utilizes mercator!

Multi Type Geometries

- Well Defined Multi Types
  - Multi Point
  - Multi Linestring
  - Multi Polygon
- No “Geometry Collections”

Geometry Encoding

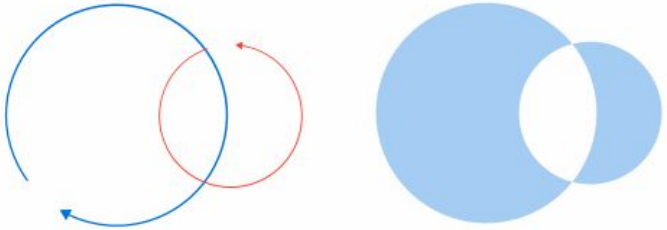
- Completely defined all “commands” and how to encode geometry in Vector Tiles
- Provided many examples in specification of encoding geometry
- A “LineTo” command must make a line
  - Can not result in a repeated point!

Winding Order

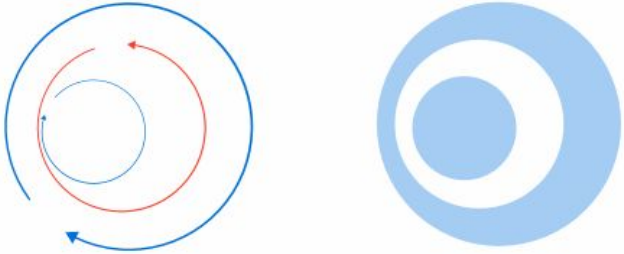
- Polygons now have a clearly defined winding order
- Area Positive is Exterior Ring
- Area Negative is Interior Ring
- Polygon
  - One Exterior Ring
  - Followed By Interior Rings
- New Polygon starts with each Exterior Ring

Polygon with Hole

Invalid Polygon



MultiPolygon

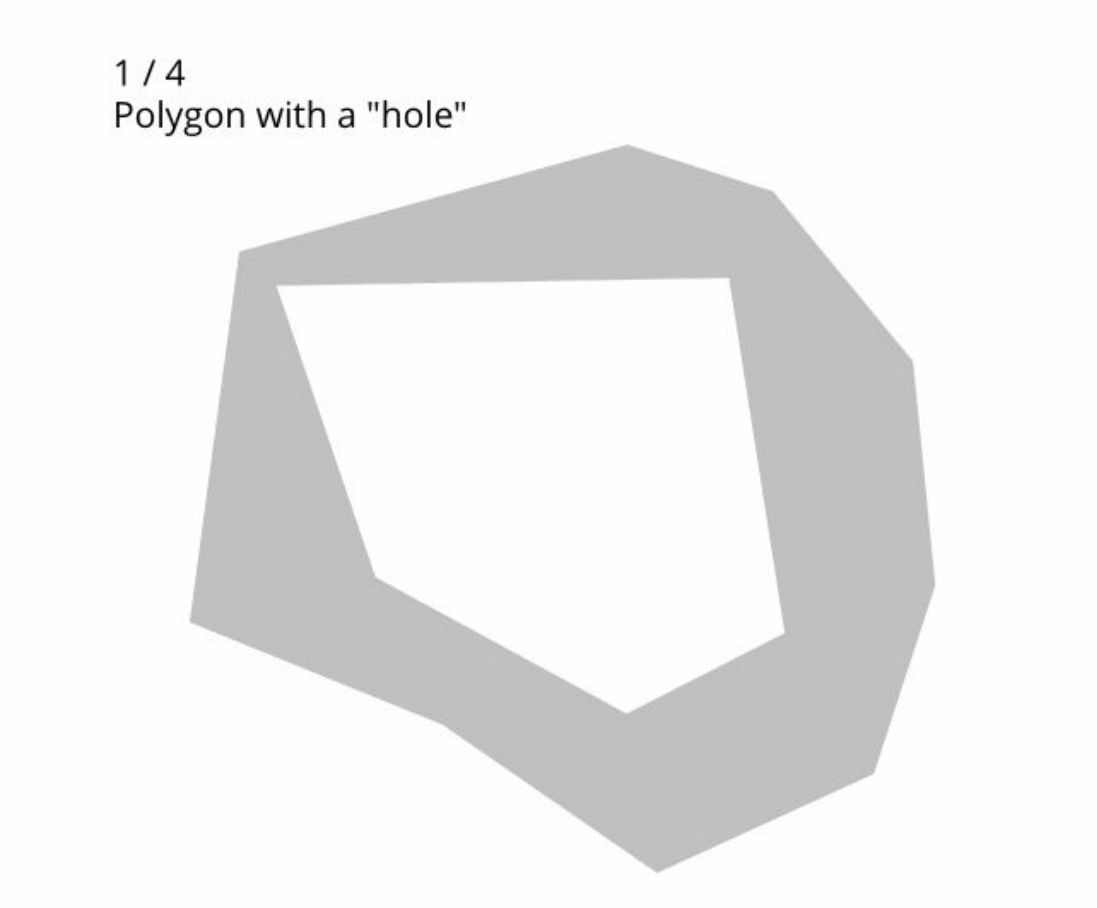


Polygon Validity

- Intersections have been a common problem
- Cause extensive problems for rendering applications

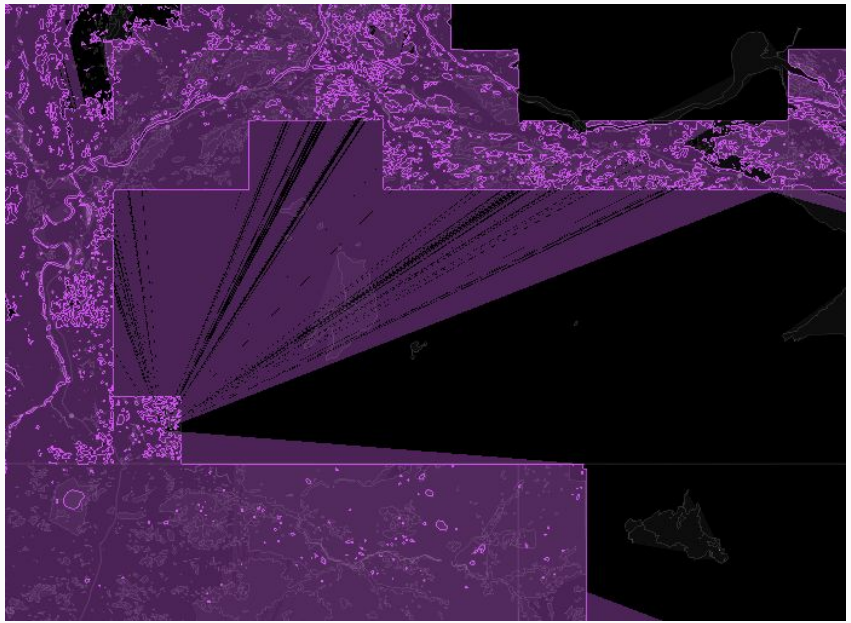
V 2.X Requires:

- No Self Intersections
- No Self Tangencies
- No Repeated Points
- “OGC Valid”



Benefits of Valid Geometries

- Improved GL rendering
  - Tessellation errors removed
  - Faster processing
- Analysis Tools using VTs
  - More Accurate
  - No clean up required



Example Tessellation Error in GL Client from Invalid Geometry

Common Misconceptions

Collection of Tiles

- Projection or location of a tile is **NOT** encoded in a tile
  - Typically determined by filename or data when stored in database.
- The specification does not define how collection of tiles should be formed
  - This might be formed later in its own specification?
  - Just like PNG format does not specify for image tiles

Vector Tiles as “Source Data”

- Vector tiles are not intended to be **precise**
  - Lose precision due to rounding to integer coordinates of Vector Tile coordinates
  - Optimized for size, speed, and display
- Best viewed as static datasource
  - Vector tiles are not designed to be edited.

Compression

- Vector Tiles are encoded efficiently but are **not** required to be compressed
  - Compression is not part of the specification!
- Gzip compression is common
  - Extension should be “.mvt.gz”
  - Web servers often send vector tiles compressed!
  - Node Mapnik automatically recognizes gzip compression

Future of the Specification?

(Experiments taking place - no promises)

Raster/Images in Vector Tiles

- Serve raster data as a layer with geometry
  - Grayscale rasters
  - Encoding of rasters?
- Subject of Experiences
- Might be best left out of specification

Other Possible Additions

- Feature Collections
- Null Values
- Improved Compression of Geometry

Questions?

Backup Slides

Vector Tiles vs PNG

- Store Geometry
- Not Human Readable
- **Not ready for display**
- No Projection
- Not Precise

- Store Raster (Image)
- Not Human Readable
- **Ready for display**
- No Projection
- Not Precise

Vector Tiles vs GeoJSON

- Store Geometry
- Not Human Readable
- Not ready for display
- No Projection
- Not Precise
- **Very Small (< 1KB)**

- Store Geometry
- **Human Readable**
- Not ready for display
- **Can Provide Projection**
- **Precise**
- Can be very large (> 1 MB)