OpenStreetMap

GSoC 2018 Project
Support For Vector Tile Data For iD Editor

Project Title

iD Editor - Support for vector tile data.
Mentor : Bryan Housel, Sajjad Anwar.

Overview

iD is an OpenStreetMap editor that allows anybody to make edits to OpenStreetMap in their web browser.
iD currently uses publicly available reference data when editing OpenStreetMap. This data is often in the form of raster tiles (aerial imagery published as bitmap, gif, jpeg).
This project aims to provide support for vector tiled data. Vector tiles are a popular open standard to deliver map geometry and metadata, in form of pre-defined roughly-square shaped "tiles", to a browser or other client application.

Why adding support for Vector tiles will be beneficial?

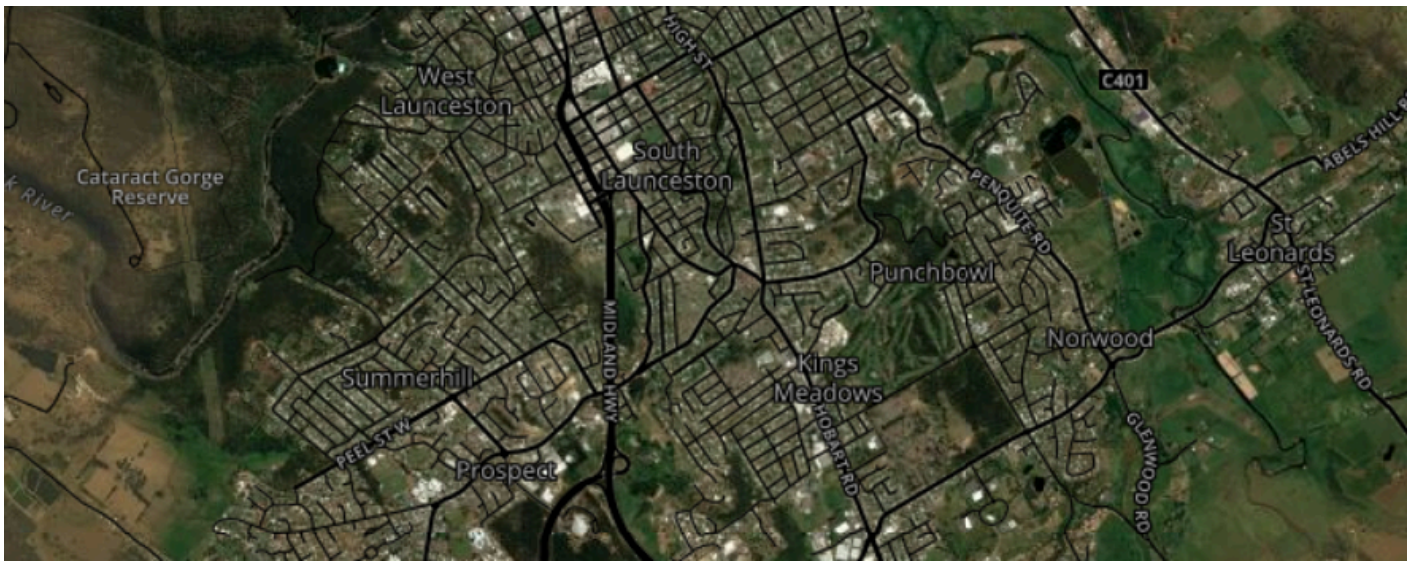
- Vector tiles are much more space efficient. They are only 20-50 percent the file size of raster tiles hence vector tiles can be rendered quickly.
- More tiles can be produced per second.
- While streaming to devices vector tiles would be a better choice as they need less bandwidth due to the smaller size of tile packages.
- Map styles (colour, grey, night mode, etc.) can be changed without needing to download more information or other tile sets.
- Dynamic labels can be placed on-the-fly. It allows to change size and font types on the fly.
- Vector tiles have data separated on different layers, and each layer can be rendered independently as per the users requirement.

Technical Details

Currently, iD supports various reference data sources as backgrounds. Some of these include :

1. Bing Aerial Imagery (Default)
2. DigitalGlobe Premium Imagery
3. Esri World Imagery
4. Stamen Terrain

One thing all of these have in common is that these api's provide reference data in the form of Raster Images. These are image tiles in a .jpg/.png format.



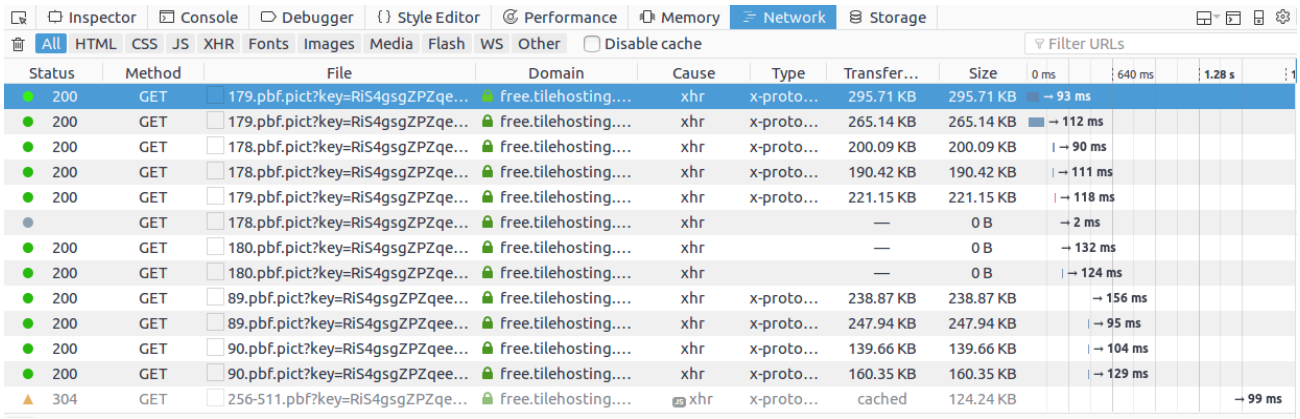
Here is a simple network capture of requests sent by the editor to the api for fetching raster tiles.

200	GET	6765.png?access_token=pk.eyJ1Ij...	b.tiles.mapbox.com	img	png	3.61 KB	3.61 KB	→ 1521836226434 ms
200	GET	6765.png?access_token=pk.eyJ1Ij...	c.tiles.mapbox.com	img	png	2.46 KB	2.46 KB	→ 488 ms
200	GET	6765.png?access_token=pk.eyJ1Ij...	d.tiles.mapbox.com	img	png	1.17 KB	1.17 KB	→ 228 ms
200	GET	6766.png?access_token=pk.eyJ1Ij...	b.tiles.mapbox.com	img	png	1.13 KB	1.13 KB	→ 502 ms
200	GET	6766.png?access_token=pk.eyJ1Ij...	c.tiles.mapbox.com	img	png	9.63 KB	9.63 KB	→ 317 ms
200	GET	6766.png?access_token=pk.eyJ1Ij...	d.tiles.mapbox.com	img	png	4.81 KB	4.81 KB	→ 226 ms
200	GET	6766.png?access_token=pk.eyJ1Ij...	a.tiles.mapbox.com	img	png	792 B	792 B	→ 253 ms
200	GET	6767.png?access_token=pk.eyJ1Ij...	c.tiles.mapbox.com	img	png	4.19 KB	4.19 KB	→ 311 ms
200	GET	6767.png?access_token=pk.eyJ1Ij...	d.tiles.mapbox.com	img	png	5.57 KB	5.57 KB	→ 695 ms
200	GET	6767.png?access_token=pk.eyJ1Ij...	a.tiles.mapbox.com	img	png	2.17 KB	2.17 KB	→ 253 ms
200	GET	6767.png?access_token=pk.eyJ1Ij...	b.tiles.mapbox.com	img	png	799 B	799 B	→ 415 ms

As it can be seen, the api returns .png's which are rendered by iD to facilitate editing of features.

Getting raster images as tiles is a great way for rendering satellite images, however, it is not the best use case for computer-generated maps like the following :

For these, data can be compressed in the form of vector tiles.
Right now, iD doesn't support these. This project would add backgrounds from api's that provide data in the form of vector tiles.
A network capture when using api's providing vector tile data looks as follows :



The server sends the tiles in .pbf format, which is what we need to interpret and render accordingly on the frontend. .pbf files are a popular open standard for transferring compressed map geometry and metadata. Another format for sharing vector tile data which makes more sense for this project is .mvt.

TIMELINE

27th March - April 23rd : Initial phase.
Fixing minor issues, getting to know the code base.

April 24th - May 15th : Community Bonding.
Community Bonding, Reading about various mapping technologies, Understanding various file formats (like mvt, pbf) used for storing vector tile data. Research on technologies used for rendering vector tiles. Read documentation to speed up the coding process later on.

May 16th - 31st May : Understanding how rendering works in iD.

1. Start working on the codebase to interpret and render vector tiles.
2. This would involve using D3.js. iD uses D3 for rendering maps. Vector tiles received in the form of .mvt files can be serialised into json using Mapbox's vector-tile-js library.
3. The library provides a toGeoJSON method that will convert the vector tile data into json.

1st June - 15th June : Looking for the best sources of vector tile data.

1. Look for api's online that provide areal data in the form of vector tiles which might be useful for iD.
2. Try rendering vector tile data from these sources.
3. Understand the endpoints of these api's.
4. Make sure their license is compatible with ours.
5. Generate access tokens.

16th June - 30th June : Create the mvt module.

1. Start working on adding functionality to request for .pbf/ .mvt files from various sources.
2. Understand how raster tiles are rendered on the frontend using D3.
3. This would involve adding a new file along the lines of https://github.com/openstreetmap/iD/blob/master/modules/svg/gpx.js , something like https://github.com/openstreetmap/iD/blob/master/modules/svg/mvt.js which would handle the events on mvt files.

1st July - 15th July : Adding functionality to the mvt module.

1. Add functions and event handlers for mvt files.
2. A few of these may include
    a. parseSaveAndZoom: Parsing Zooming
    b. getGeoJSON: Get geoJSON data from .mvt file
    c. drawMVT: Render the mvt data
3. Start working on functionality to render layers separately

16th July - 31st July : Add functionality to toggle between multiple layers.

1. Complete work on rendering multiple layers separately.
    a. Mvt files allow the functionality of writing multiple layers in them
    b. It also provides the functionality to decide which layer to write on the basis of the zoom.
    c. The final phase of the project would involve allowing the user to render separate layers, toggle which layers to render.
2. Start working on documentation of written code.

1st August - Final Evaluation : Final testing and bug fixing.

1. Write tests
2. Create an OSM Wiki page for all the work done.
3. Invite the OSM community to review the work and get suggestions on how we can improve the project in the future.

Reference

- https://wiki.openstreetmap.org/wiki/Vector_tiles
- https://en.wikipedia.org/wiki/Vector_tiles#Client_libraries
- https://mapdataservices.wordpress.com/2017/02/22/vector-tiles-vs-raster-tiles-the-pros-and-cons/
- https://wiki.openstreetmap.org/wiki/Google_Summer_of_Code/2018/Project_Ideas
- https://wiki.openstreetmap.org/wiki/Google_Summer_of_Code/2018#Project_Proposal
- https://www.mapbox.com/vector-tiles/
- https://github.com/mapbox/awesome-vector-tiles