mapbox Vector Tile formátum

hatékony adatmegjelenítés a weben

Gelencsér Gergő

Szoftverfejlesztő mérnök

MindiGIS

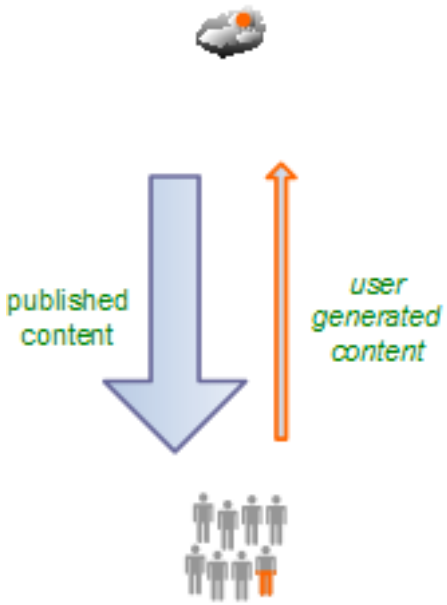
Miről lesz szó?

- Alkalmazásfejlesztési tapasztalatok
- Adat és fájl formátumok, betöltési stratégiák
- Az MVT formátum és miért ***kell*** ismernünk
- Hogy lehet GeoServerben beállítani?
- Hogy lehet közvetlenül adatbázisból generálni?
- Performancia!

Web 1.0

"the mostly read-only Web"

250,000 sites



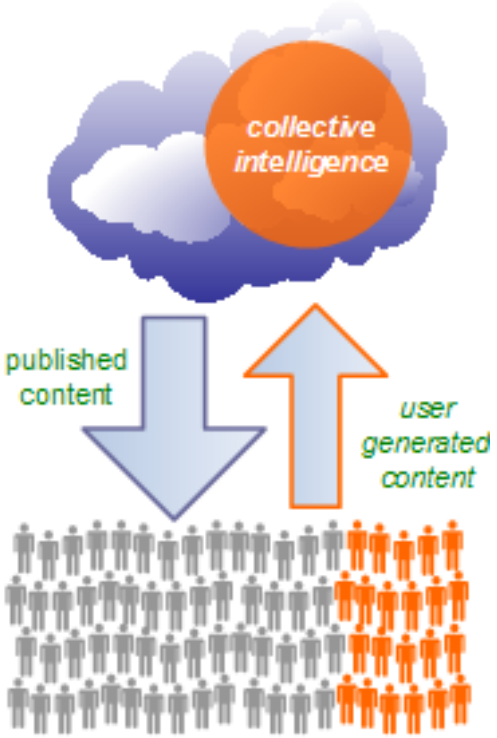
45 million global users

1996

Web 2.0

"the wildly read-write Web"

80,000,000 sites



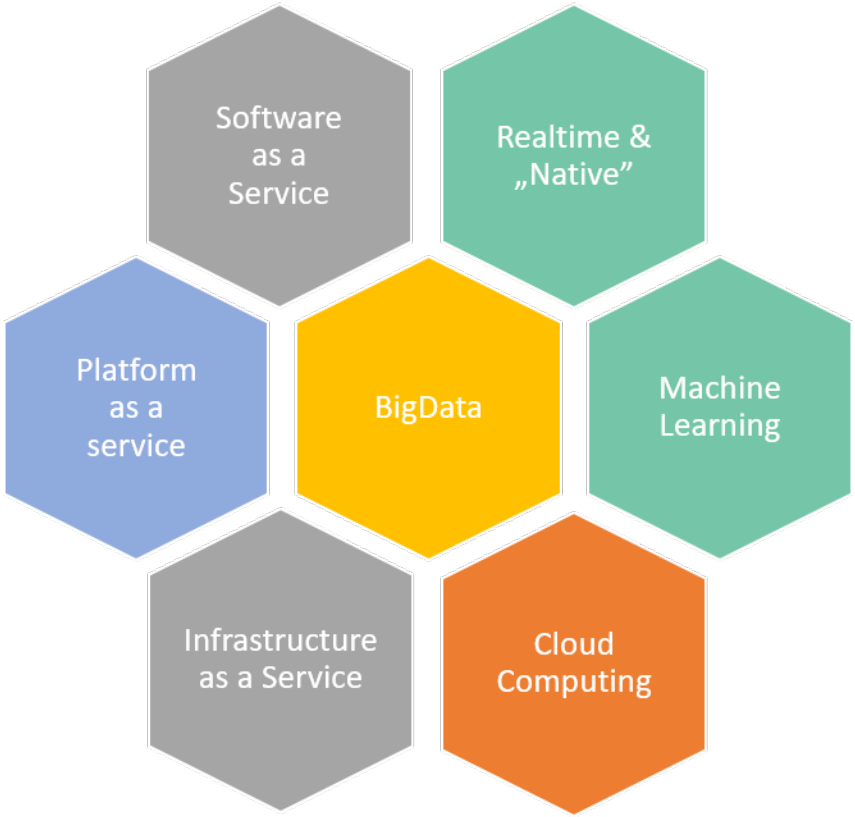
1 billion+ global users

2006

Web 3.0

„the executable Web"

1,000,000,000 sites



2016

WebGIS stack – még egyszer utoljára…

Adattár	Térkép szerver	Alkalmazás	Web szerver	User interface
PostgreSQL + PostGIS	GeoServer	spring Java	APACHE HTTP SERVER PROJECT	OpenLayers
MariaDB MySQL	MapServer	django	NGINX	Leaflet
mongoDB.	QGIS	Express JS	node	mapbox GL JS
File storage	Tegola.io	laravel php		

HTTP

Front-end

Back-end

Evolúció? – Hogy működnek a webes térképek?

Tárolás

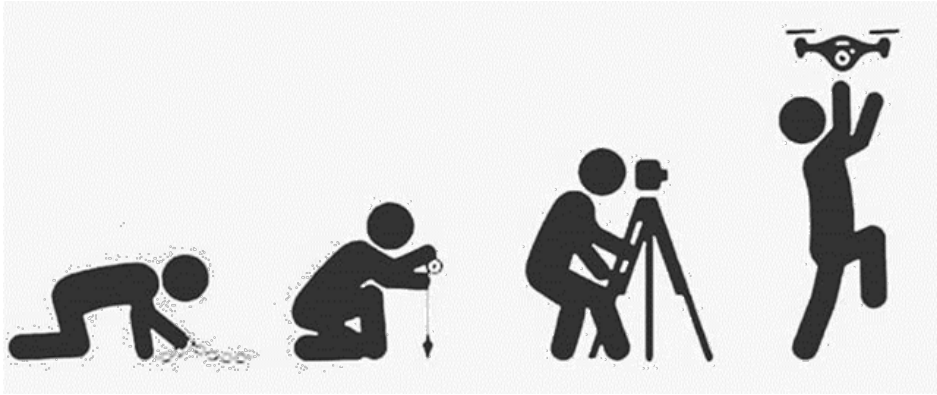
- Raszter: előre v. röptében generált kép
- Tile: raszter csempék, piramisok, redundancia
- Vektor: nyers adat, kliens renderel
- **Vektor csempék?!**

Stratégiák

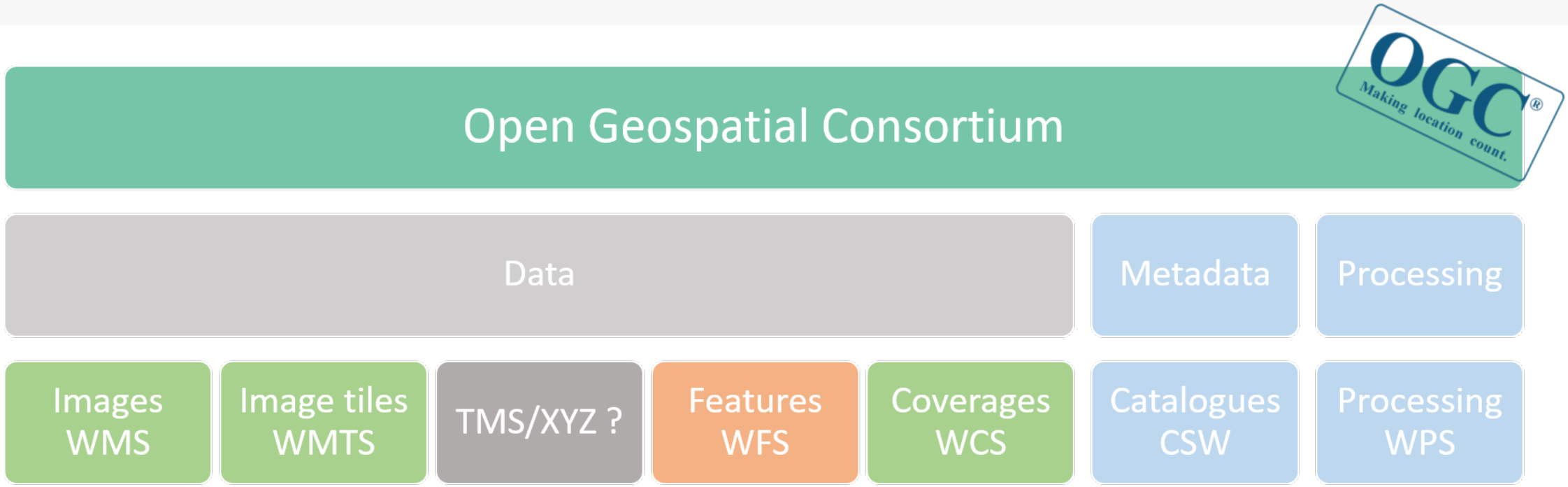
- All: mindet egyben letöltünk
- BBOX: ami a képernyőn látszik, újra és újra
- **TileGrid**: rácsháló

Formátumok

- Szöveges: CSV, WKT, DXF
- [Raszter]: GeoTIFF+TFW, GeoJPEG, PNG
- <XML/>: GML, GPX, KML
- {JSON}: EsriJSON, GeoJSON, TopoJSON
- B1nár1s: **MVT**, SHP+DBF+SHX+PRJ

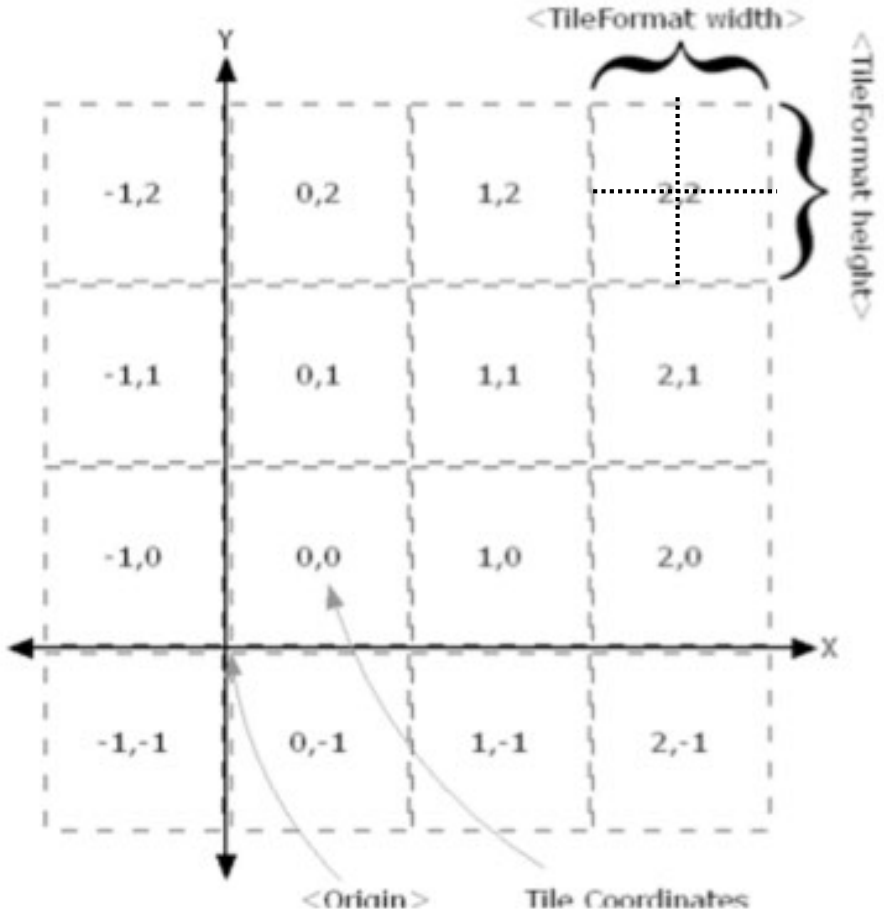


A nyílt szabvány mindig jó választás…

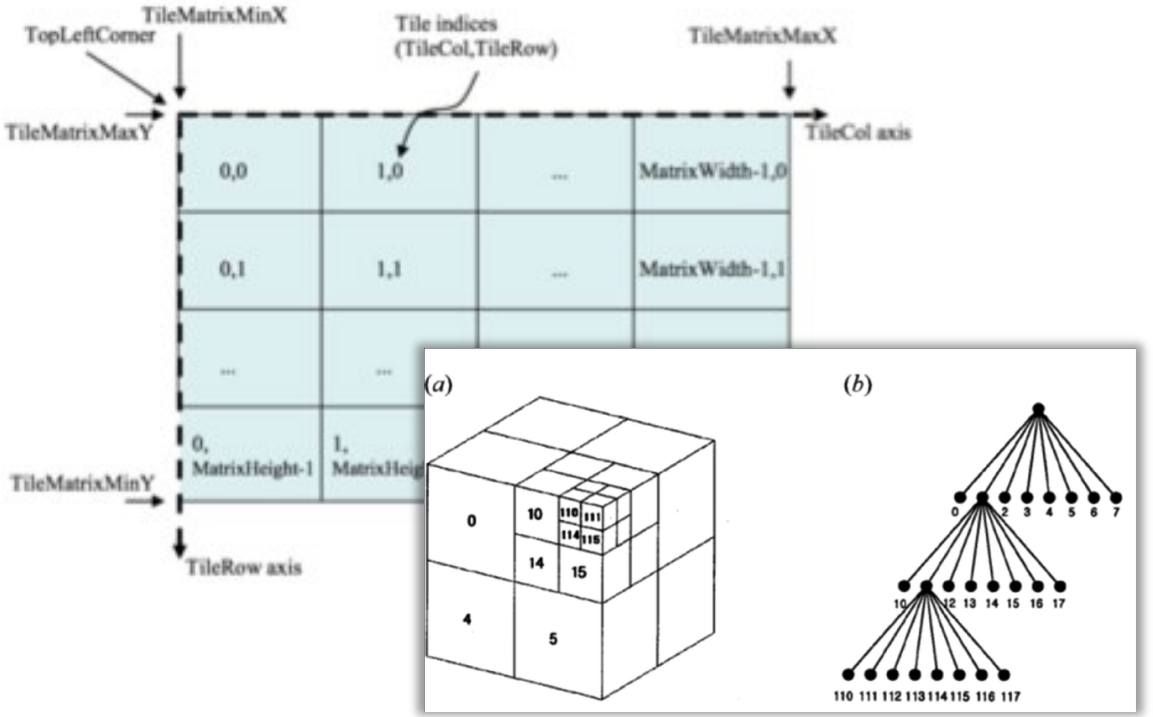


A TMS nem szabvány, „csak” az OSGeo (Open Source Geospatial Foundation) ajánlása.
Csak a működés módját határozzák meg, a protokollt (GET-KVP, POST-XML, SOAP, REST)
és formátumot (GML, GeoTiff, GeoJSON, MVT) nem!

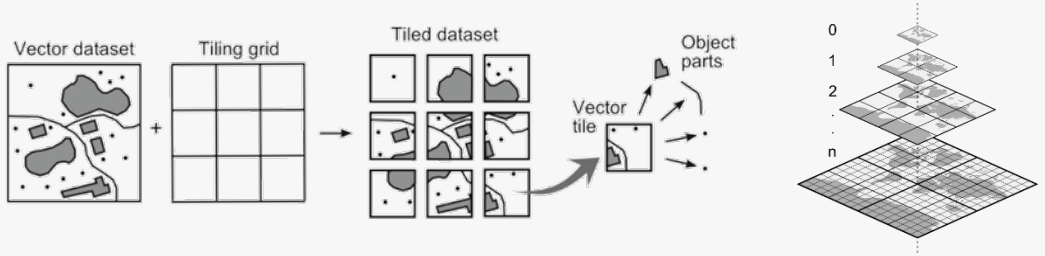
TMS, (XYZ)


<TileFormat width>
<TileFormat height>
Y
X
-1,2
0,2
1,2
2,2
-1,1
0,1
1,1
2,1
-1,0
0,0
1,0
2,0
-1,-1
0,-1
1,-1
2,-1
<Origin>
Tile Coordinates

WMTS


TopLeftCorner
TileMatrixMinX
Tile indices
(TileCol,TileRow)
TileMatrixMaxX
TileMatrixMaxY
TileCol axis
0,0
1,0
...
MatrixWidth-1,0
0,1
1,1
...
MatrixWidth-1,1
...
...
0,
MatrixHeight-1
1,
TileMatrixMinY
TileRow axis
(a)
(b)
0
10
110
111
114
115
14
15
4
5
0 2 3 4 5 6 7
10 12 13 14 15 16 17
110 111 112 113 114 115 116 117

Hibrid megoldás



Előnyök a képpiramisokhoz képest

- **Stílusok** a kliens oldalon, nincs kőbe (képbe) vésve a megjelenítés
- **Letöltés** csak egyszer, attribútum adatokkal
- **Tárolás** csak egyszer
- **Méret**: a csempék tényleg meglepően kicsik. Lehetővé teszi nagy méretű, részletgazdag adatok megjelenítését gyors betöltés és gyorsítótárazás, streamelés lehetősége mellett.
- **Gördülékeny érzés**, snappelés, animációk. A 21. században érezzük magunkat tőle. ☺

Hátrányok a vektorhoz képest

- **Kartografálás** a kliens képességeihez mérten
- **Vastagabb vékony kliensek**, a régi egyszerűbb szoftverek nem támogatják
- **Generalizálás**: csak a legalsó zoom szinten kapjuk meg az eredeti adatot (megjelenítésre optimalizáltan)
- **Pontatlanabb koordináták** hatására topológiai hibák jelentkezhetnek. Pl.: a lyuk kilóg a felületből?
- **Extra számítás igény** mindkét oldalon, ha nem gyorsítótárazunk

A formátumról – tévhitek és tények

Tévhit: PBF = MVT

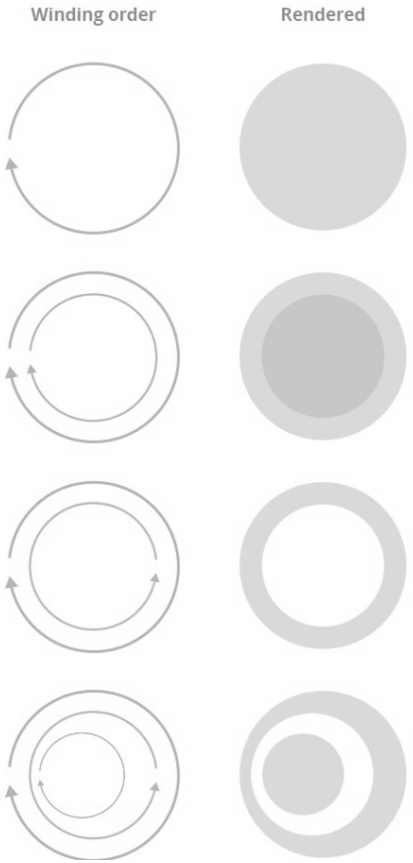
- A Protocol buffert a Google készítette. Ez egy nyelv- és platformsemleges mechanizmus strukturált adatok továbbítására. „Olyan mint az XML, csak kisebb gyorsabb és egyszerűbb.” https://github.com/protocolbuffers/protobuf
- A PBF-ek különböző struktúrákat használnak az adatok leírására. A Google Maps, az OpenStreetMap és a Mapbox PBF fájlok semmilyen kapcsolatban sem állnak egymással. https://docs.mapbox.com/vector-tiles/specification/

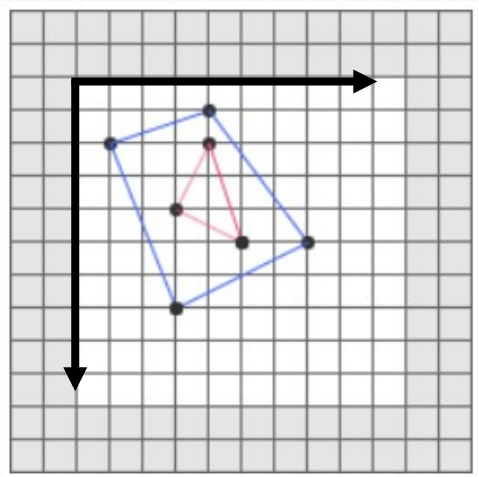
Tény: az MVT open standard

- Mapbox Inc. 2010-ben alakult, 2013-ban készült el a Vector Tiles spec.
- Ismerős lehet még: Tilemill, Mapbox GL JS, Mapbox Studio, SDK for iOS and Android, Geocoding, Turn-by-turn navigation, ARKit.

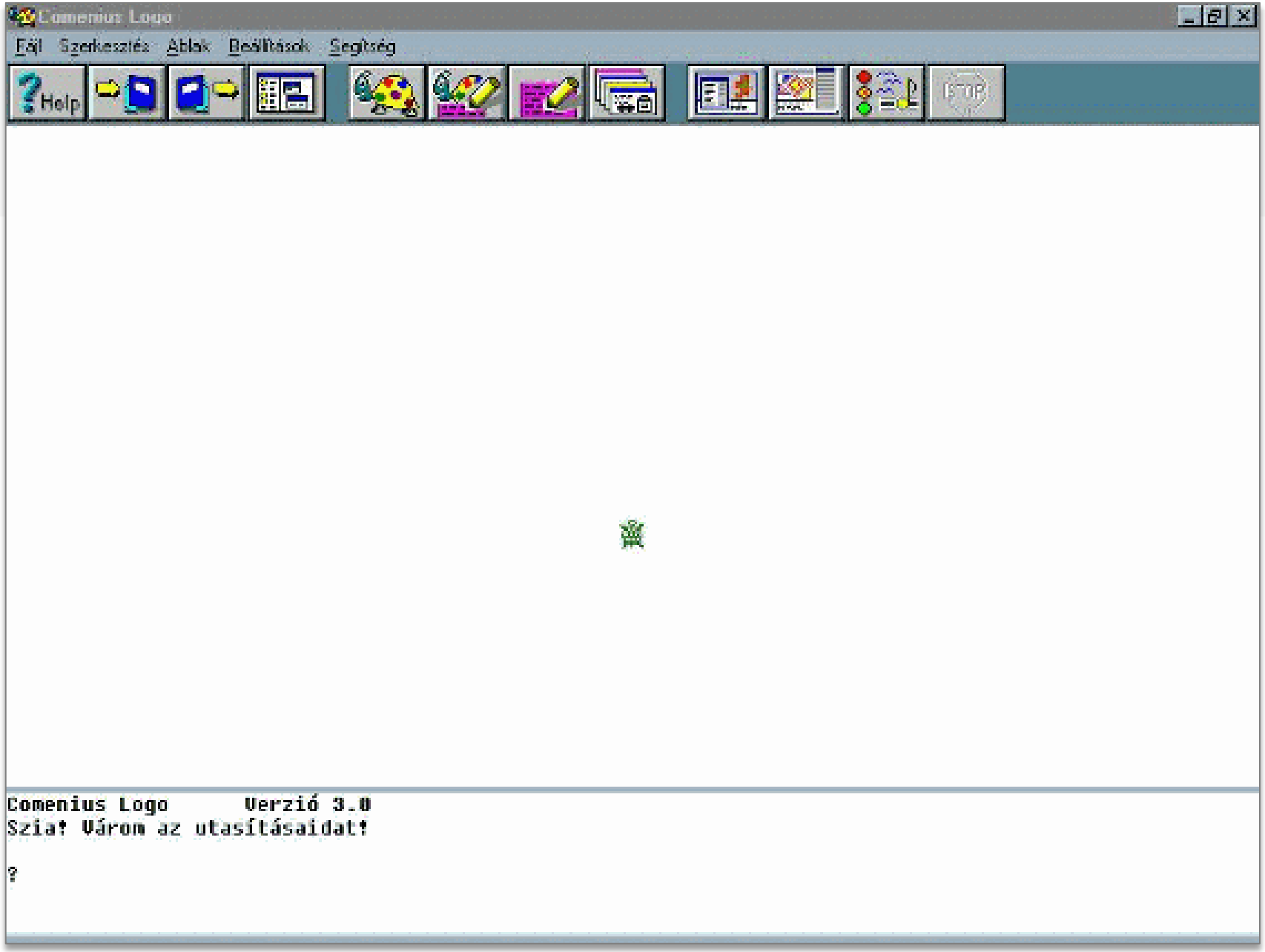
Geometria kódolás

- Tárolás: térbeli rácspontokban (kis egész számok) Pl.: 10x10 rács, 2 cella pufferrel
- Entitások: pont, vonal és felület
- Minden utasítás relatív koordinátákkal dolgozik az utolsó ponthoz képest
- Felületek zárása: ClosePath = MoveTo first coord.
- A folytonosságot az irányultság határozza meg
- Vágás: nem kötelező, de a puffer mentén vágni is lehet. ID attribútum lehet a kapocs.
- „Z” szerinti generalizálás, 1 px alatti elemek és vetöréspontok rtexek elrejtése
- Comenius logo? ☺





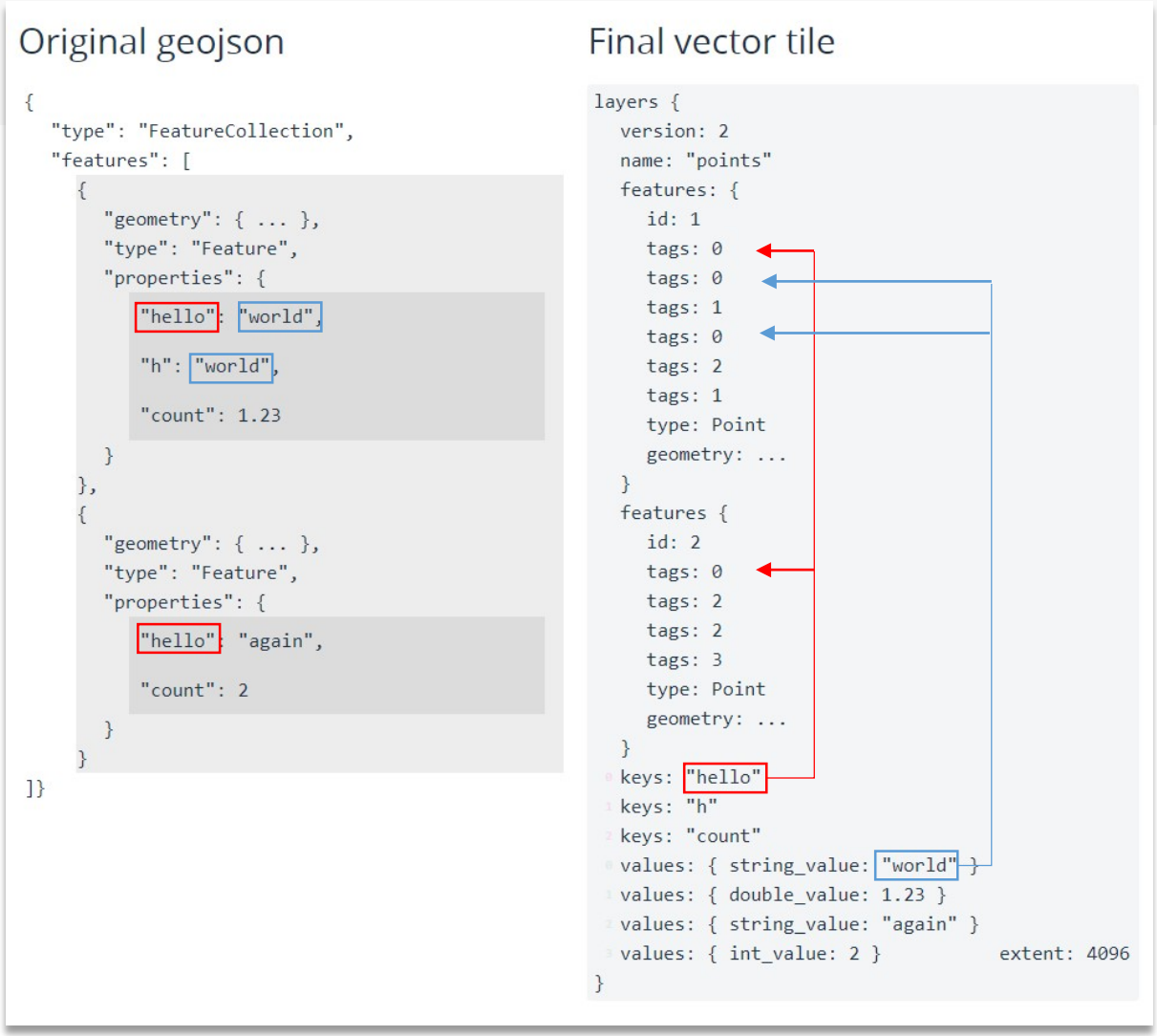
```
MoveTo(1,2)
LineTo(3,-1)
LineTo(3,4)
LineTo(-4,2)
ClosePath()
MoveTo(1,-5)
LineTo(-1,2)
LineTo(2,1)
ClosePath()
```


Comenius Logo
Fájl Szerkesztés Ablak Beállítások Segítség
Help
STOP
Comenius Logo Verzió 3.0
Szia! Várom az utasításaidat!
?

Attribútum kódolás

- Bonyolultabb az attribútumok tárolása
- Az eredeti kulcs - érték párok felbontása úgy, hogy az ismétlődéseket minimalizáljuk
- A specifikáció nem rendelkezik az összetett adatstruktúrák tárolásáról, ezért a nem szám és nem szöveg típusú attribútumok szövegként kerülnek tárolásra.

  Pl.: "categories": ["*one*", "*two*"]
  "categories": "*[\"one\",\"two\"]*"



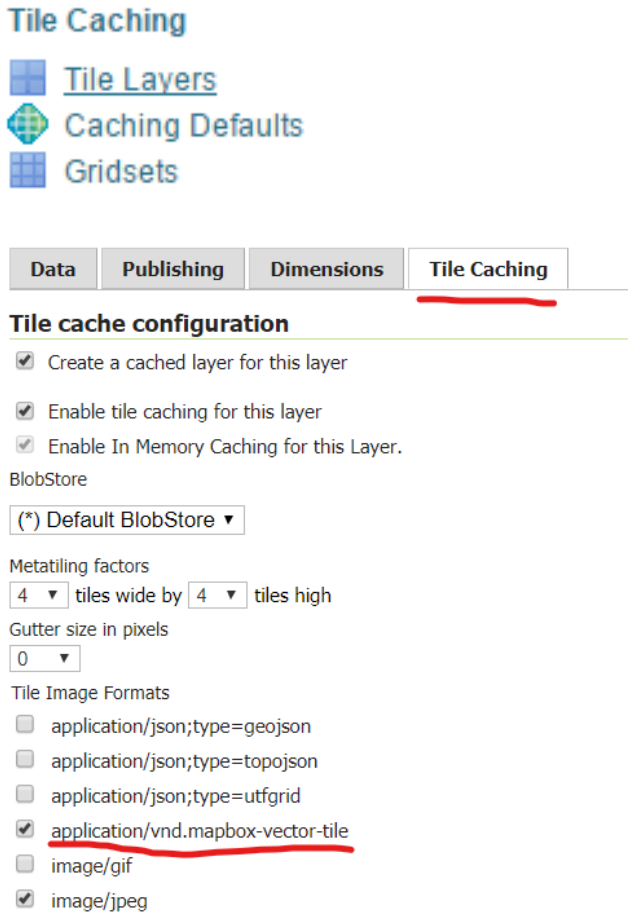
Támogatottság

- Implementációk listája a Mapbox oldalán: https://github.com/mapbox/awesome-vector-tiles
- QGIS 2.18: http://plugins.qgis.org/plugins/vector_tiles_reader/
- GDAL 2.3: https://gdal.org/drivers/vector/mvt.html
- GeoServer 2.11: https://docs.geoserver.org/latest/en/user/extensions/vectortiles/index.html
- MapServer 7.2: https://mapserver.org
- PostGIS 2.4: https://postgis.net/docs/ST_AsMVT.html
- OpenMapTiles: https://openmaptiles.org

GeoServer + OpenLayers

- C:\...\GeoServer\webapps\geoserver\WEB-INF\lib
- http://sourceforge.net/projects/geoserver/files/GeoServer/2.16.0/extensions/geoserver-2.16.0-vectortiles-plugin.zip

```
202   // MVT file
203   new ol.layer.VectorTile({
204     title: 'MVT',
205     source: new ol.source.VectorTile({
206       format: new ol.format.MVT(),
207       tileGrid: ol.tilegrid.createXYZ({ maxZoom: 10 }),
208
209       // Filesystem
210       url: 'data/boundaries.mvt/{z}/{x}/{y}.pbf',
211
212       // TMS service
213       url: 'http://localhost:8080/geoserver/gwc/service/'
214          + 'tms/1.0.0/boundaries@EPSG%3A3857@pbf/{z}/{x}/{y}.pbf',
215
216       // WMTS service
217       url: 'http://localhost:8080/geoserver/gwc/service/'
218          + 'wmts?SERVICE=WMTS&VERSION=1.0.0&REQUEST=GetTile&LAYER=boundaries'
219          + '&STYLE=&FORMAT=application%2Fvnd.mapbox-vector-tile'
220          + '&TILEMATRIX=EPSG:3857:{z}&TILEMATRIXSET=EPSG:3857&TILECOL={x}&TILEROW={y}',
221     }),
222     style: vectorStyle,
223     visible: false
224   }),
```



PostGIS MVT

```sql
-- Generate Mapbox Vector Tiles from SQL
SELECT ST_AsMVT(features, 'tablename', 4096, 'geom', 'id')
FROM (
  SELECT
    layer.id as id,
    ST_AsMvtGeom(
      ST_SimplifyPreserveTopology(ST_Transform(geom, 3857), 156412.0 / 2 ^ tileZ),
      ST_TileEnvelope(tileZ, tileX, tileY), -- bounds of the tile (x, y, z)
      4096, -- tile extent, grid dimensions will be 4096x4096
      256,  -- 256 grid cells in addition to the tile extent
      true  -- geometries will be clipped at the grid + buffer boundaries
    ) AS geom,
    layer.* -- exclude layer.id and layer.geom
  FROM tablename As layer
  WHERE ST_Transform(geom, 3857)
    && ST_Transform(ST_TileEnvelope(tileZ, tileX, tileY), 3857)
    -- check that the bbox of the geometry overlaps the tile bbox
    AND ST_Intersects(
      ST_Transform(geom, 3857),
      ST_Transform(ST_TileEnvelope(tileZ, tileX, tileY), 3857)
    ) -- check that the geometry overlaps the tile bbox
) AS features;
```

```sql
-- this funcion is part of PostGIS 3.0
CREATE OR REPLACE FUNCTION ST_TileEnvelope(
  zoom integer, x integer, y integer
)
RETURNS geometry AS $$
  DECLARE
    max numeric := 6378137 * pi();
    res numeric := max * 2 / 2^zoom;
    bbox geometry;
  BEGIN
    return ST_MakeEnvelope(
        -max + (x * res),
        max - (y * res),
        -max + (x * res) + res,
        max - (y * res) - res,
        3857);
  END;
$$
LANGUAGE plpgsql IMMUTABLE;
```

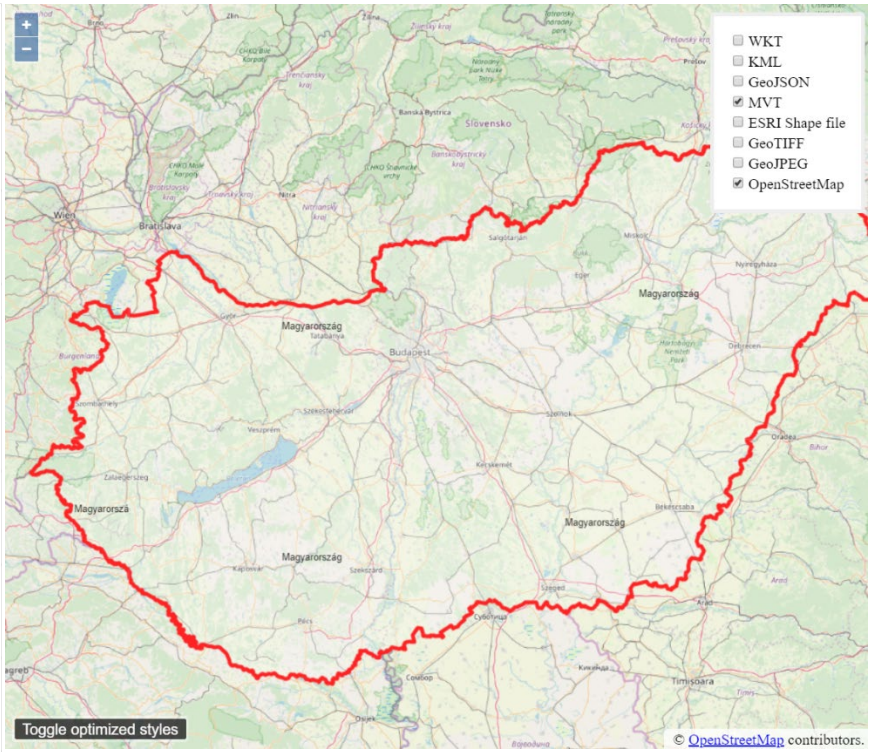
PostGIS GeoJSON, KML

```sql
-- Generate GeoJSON from SQL
SELECT row_to_json(featurecollection) As collection
FROM (
  SELECT
    'FeatureCollection' As type,
    json_build_object(
      'type', 'name',
      'properties', json_build_object('name', concat('EPSG:', 3857))
    ) As crs,
    array_to_json(COALESCE(array_agg(feature),'{}')) As features
  FROM (
    SELECT
      'Feature' As type,
      id As id,
      ST_AsGeoJSON(ST_Transform(geom, 3857), 9, 0)::json As geometry,
      row_to_json(layer.*) As properties -- exclude layer.geom
    FROM tablename As layer
    WHERE ST_Transform(geom, 3857)
      && ST_MakeEnvelope(minX, minY, maxX, maxY, 3857)
      -- check that the bbox of the geometry overlaps the tile bbox
      AND ST_Intersects(
        ST_Transform(geom, 3857),
        ST_MakeEnvelope(minX, minY, maxX, maxY, 3857)
      ) -- check that the geometry overlaps the tile bbox
  ) As feature
) As featurecollection
```

```sql
-- Generate KML from SQL
SELECT
  xmlelement(
    NAME kml, XMLATTRIBUTES ('http://www.opengis.net/kml/2.2' AS xmlns),
    xmlelement(
      NAME "Document", XMLATTRIBUTES ('doc_root' AS id),
      xmlelement(
        NAME "Folder",
        xmlelement(name "name", 'tablename'),
        xmlelement(
          NAME "Schema",
          XMLATTRIBUTES ('data' AS NAME, 'dataSchema' AS id),
          xmlelement(
            NAME "SimpleField", XMLATTRIBUTES ('[FieldType]' AS TYPE, '[FieldName]' AS NAME)
          ) -- repeat this for every field exclude layer.geom
        ),
        -- <Style id="[StyleID]"><LineStyle/><PolyStyle/></Style>
        xmlagg(placemarks)
      )
    )
  ) As document
FROM (
  SELECT
    xmlelement(name "Placemark",
      xmlconcat(
        xmlelement(name "name", layer.id),
        xmlelement(name "description", layer.*),
        -- <TimeSpan><begin/><end/></TimeSpan>
        -- <styleUrl>#[StyleID]</styleUrl>
        xmlelement(
          NAME "ExtendedData",
          xmlelement(
            NAME "SchemaData",
            XMLATTRIBUTES ('#dataSchema' AS "schemaUrl"),
            xmlelement(
              NAME "SimpleData", XMLATTRIBUTES ('[FieldName]' AS NAME), layer.FieldName
            ) -- repeat this for every field exclude layer.geom
          )
        ),
        ST_AsKML(ST_Transform(geom, 4326), 9)::xml
      )
    )::xml As placemarks
  FROM tablename As layer
  WHERE ST_Transform(geom, 3857)
    && ST_MakeEnvelope(minX, minY, maxX, maxY, 3857)
    -- check that the bbox of the geometry overlaps the tile bbox
    AND ST_Intersects(
      ST_Transform(geom, 3857),
      ST_MakeEnvelope(minX, minY, maxX, maxY, 3857)
    ) -- check that the geometry overlaps the tile bbox
) As placemarks
```

Performancia

Formátum	Méret	Letöltés	Feldolgozás	Összesen
SHP	11 MB	1.30s	1.79s	3.1s
GeoJSON	25 MB	0.46s	2.00s	2.4s
KML	25 MB	0.51s	2.21s	2.7s
WKT	15 MB	0.42s	1.20s	1.6s
GeoTIFF	30 MB	1.65s	2.30s	3.9s
GeoJPEG	1 MB	0.30s	0.10s	0.4s
MVT	~ 7,5 MB	0.10s	0.20s	0.3s



http://programmerg.github.io/GIS-file-formats/

Performancia

On-the-fly styling:	1150ms	Cached styles:	1000ms
Predefined style:	800ms	No styling:	770ms

```
76                  // client side styling
77                  // its very important for performance to cache
78      0.4 ms      let vectorStyle = (feature, resolution) => {
79                    let defaultStyle = new ol.style.Style({
80      2.1 ms          fill: new ol.style.Fill({
81                        color: [55, 126, 184, 0.24]
82                      }),
83      1.6 ms          stroke: new ol.style.Stroke({
84                        color: [255, 35, 35],
85                        width: 1.5
86                      }),
87      0.7 ms          text: new ol.style.Text({
88                        text: '',
89      1.0 ms            fill: new ol.style.Fill({
90                          color: [49, 132, 195]
91                        })
92                      })
93                    });
94      0.5 ms        let text = feature.get('name');
95      0.6 ms        defaultStyle.text_.text_ = text;
96      0.3 ms        return [defaultStyle];
97                  };
```

```
76                  // client side styling
77                  let defaultStyle = new ol.style.Style({
78                    fill: new ol.style.Fill({
79                      color: [55, 126, 184, 0.24]
80                    }),
81                    stroke: new ol.style.Stroke({
82                      color: [49, 132, 195],
83                      width: 1
84                    }),
85                    text: new ol.style.Text({
86                      text: '',
87                      fill: new ol.style.Fill({
88                        color: [49, 132, 195]
89                      })
90                    })
91                  });
92                  // its very important for performance to cache
93      0.5 ms      let vectorStyle = (feature, resolution) => {
94      1.0 ms        let text = feature.get('NAME');
95      0.6 ms        defaultStyle.text_.text_ = text;
96      0.3 ms        return [defaultStyle];
97                  };
```

Köszönöm a figyelmet!

http://github.com/programmerg