pgmapcss

advanced cartography for Mapnik

Stephan Bösch-Plepelits
http://plepe.at - http://twitter.com/plepe
plepe@openstreetmap.at

Slides of this talk: http://plepe.at/194

About the lecturer

- Name: Stephan Bösch-Plepelits
- Started mapping in 2008
- A variety of projects, mainly integrated in OpenStreetBrowser: Background style, interactive overlays, OSM / Wikipedia, Database structure, …
- New project started in summer 2013: pgmapcss

Outline

- What is MapCSS?
- Introduction: pgmapcss
- Some features of pgmapcss beyond MapCSS standard
- Questions

I have a dream ...

… to use the same map style on a variety of devices and platforms:

Web (background tiles), Web (interactive overlays), Mobile / Smartphone, Desktop applications, Editor, Print, …

MapCSS → Potlatch2, JOSM, Overpass Turbo, Kothic(JS), OpenStreetPad, pgmapcss, ...

MapCSS vs. CartoCSS

- 1 (type of) file:
  - .mapcss containing style info; query implicit through selectors
- renderer needs to know db structure

- specific for Mapnik
- 2 (types of) files:
  - .mml containing layers + datasource
  - .mms containing style info
- db queries defined in datasource

MapCSS - Example

```
canvas {
  fill-color: grey;
}
line|z12-[highway=primary],
line|z13-[highway=secondary],
line|z14-[highway=tertiary] {
  color: #ffffff;
  width: 6;
  casing-width: 1;
  casing-color: #a0a0a0;
  text: eval(tag("ref") . " - " .
    tag("name"));
}
area|z14-[leisure=park] {
  fill-color: green;
}
node|z10-13[place=city][population>150000] {
  text: name;
  font-size: 10;
}
node|z14-16[place=city][population>150000] {
  text: name;
  font-size: 12;
}
```

http://pgmapcss.openstreetbrowser.org/?style=0ceb7&zoom=14&lat=48.2078&lon=16.3700

MapCSS - Selectors

- Type of object:
  `node`, `point`, `way`, `line`, `area`, `relation`, `canvas`
- Zoom-Level: `|z15`, `|z15-`, `|z-15`, `|z13-15`
- Tag-comparison (can be used multiple times):
  - `[highway]` // has tag 'highway'
  - `[highway=residential]` // tag 'highway' has value 'residential'
  - `[highway!=footway]` // has no tag 'highway' or value of tag 'highway' not equal 'footway'
  - … many more (see documentation for details)
- „Layer“ resp. „Pseudo Element“ - several symbolizers for same object:

```
line[highway]::el1
line[railway]::el2
```

MapCSS – special features

- all ways which are member of a relation:
```
relation[type=route] way { … }
relation[type=route] > way { … }
```

- set tag “tag” to a specific value:
```
{ set tag = value; }
```

- Eval-Functions, e.g.: convert metric value to pixels:
```
{ width: eval(metric("30m")); }
```

pgmapcss – how does it work?

- Import OSM to database using osm2pgsql (check documentation for special parameters)
- Compile style: `pgmapcss -d osm -u user -p password -t mapnik-2.2 foobar.mapcss`
  - will create (and install) a database function called `pgmapcss_foobar` which does the magic (query for renderable objects, assess how each object should be displayed, calculations, …)
  - will create a file `foobar.mapnik` – the XML file for rendering
- Can be used with normal rendering tool chains (e.g. Apache2 / mod_tile, nik2img, ...)

pgmapcss – how does it work?

```
line[highway=residential] {
  width: 2;
  color: #000000;
}
```

→ compiles into following database function (Python3k) (very simplified):

```
for ob in db.query(“select * from planet_osm_line where
"highway"='residential' and way && !bbox!”):
  props = { 'geo': ob['geo'] }
  if ob.type==“line” and ob.tags[“highway”]=='residential':
    props['width'] = “2”
    props['color'] = “#000000”

  yield props
```

pgmapcss – how does it work?

- Normal Mapnik style file:

```
<xml><Style><Rule>
    <Filter>[highway] = “residential”</Filter>
    <LineSymbolizer stroke=”#000000” stroke-width=”2” />
  </Rule></Style>
  <Layer><DataSource><Parameter name=”table”>
  select * from planet_osm_line where highway is not null
  </Parameter></DataSource></Layer></xml>
```

- pgmapcss generated Mapnik style file:

```
<xml><Style><Rule>
    <LineSymbolizer stroke=”[color]” stroke-width=”[width]” />
  </Rule></Style>
  <Layer><DataSource><Parameter name=”table”>
    select * from pgmapcss_foobar(!bbox!, !scale_denominator!)
  </Parameter></DataSource></Layer></xml>
```

→ The magic is moved to the “pgmapcss_foobar” database function

pgmapcss – how does it work?

- pgmapcss generated Mapnik 3.0 style file:

```
<xml><Style><Rule>
    <LineSymbolizer stroke="[color]" stroke-width="[width]" />
  </Rule></Style>
  <Layer><DataSource><Parameter name="table">
    select * from pgmapcss_foobar(!bbox!, !scale_denominator!)
  </Parameter></DataSource></Layer></xml>
```

- pgmapcss generated Mapnik 2.x style file:

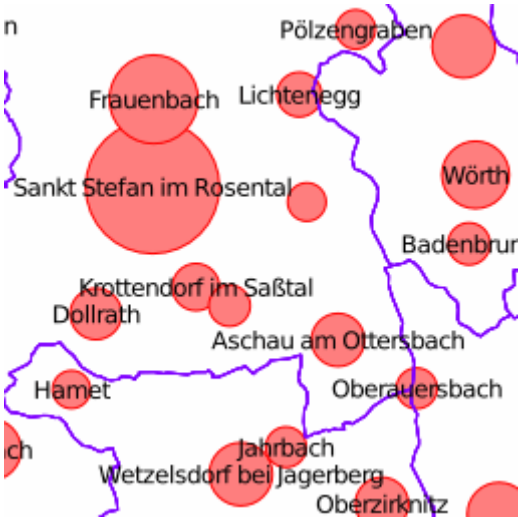
```
<xml><Style><Rule>
    <Filter>[color] = "#000000" and [width] = "2"</Filter>
    <LineSymbolizer stroke="#000000" stroke-width="2" />
  </Rule></Style>
  <Layer><DataSource><Parameter name="table">
    select * from pgmapcss_foobar(!bbox!, !scale_denominator!)
  </Parameter></DataSource></Layer></xml>
```

→ Mapnik 2.x can't read width and color (and most other parameters) directly from a database column

pgmapcss – geometric functions

- geometric functions, e.g. buffer(), translate(), line_length(), azimuth(), centroid(), ...
- calculations in pixel space; values can be converted with metric()

```
point[place]::diagram {
  geo: eval(buffer(prop(geo),
         sqrt(tag(population))));
  fill-color: #ff7f7f;
  color: #ff0000;
  width: 1;
}
```



http://pgmapcss.openstreetbrowser.org/?style=fc3ee&zoom=13&lat=46.8685&lon=15.7199

pgmapcss – combine map features

- “combine” creates new object type (e.g. “street”) with id from eval()-statement
- geo: geometry collection
- Example: fragmented roads:

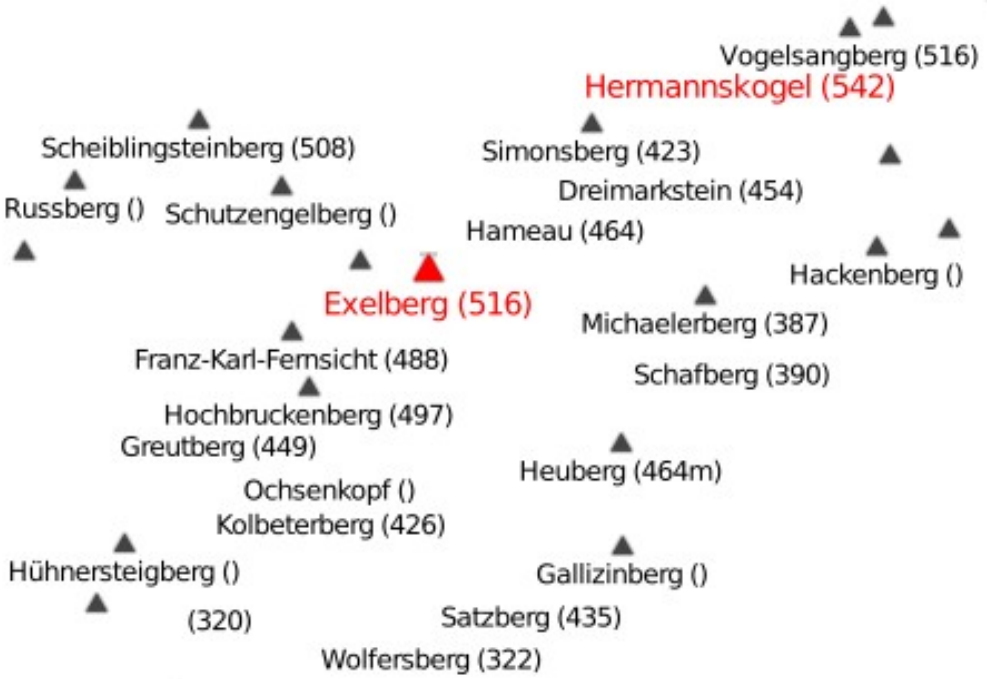
```
line[highway] {
  combine street eval(tag(„name“));
}

street {
  geo: eval(line_merge(prop(geo)));
  text: name;
}
```

w/o combine: http://pgmapcss.openstreetbrowser.org/?style=330ca&zoom=15&lat=48.1983&lon=16.3471
with combine: http://pgmapcss.openstreetbrowser.org/?style=a51b7&zoom=15&lat=48.1983&lon=16.3471

pgmapcss – highest peaks

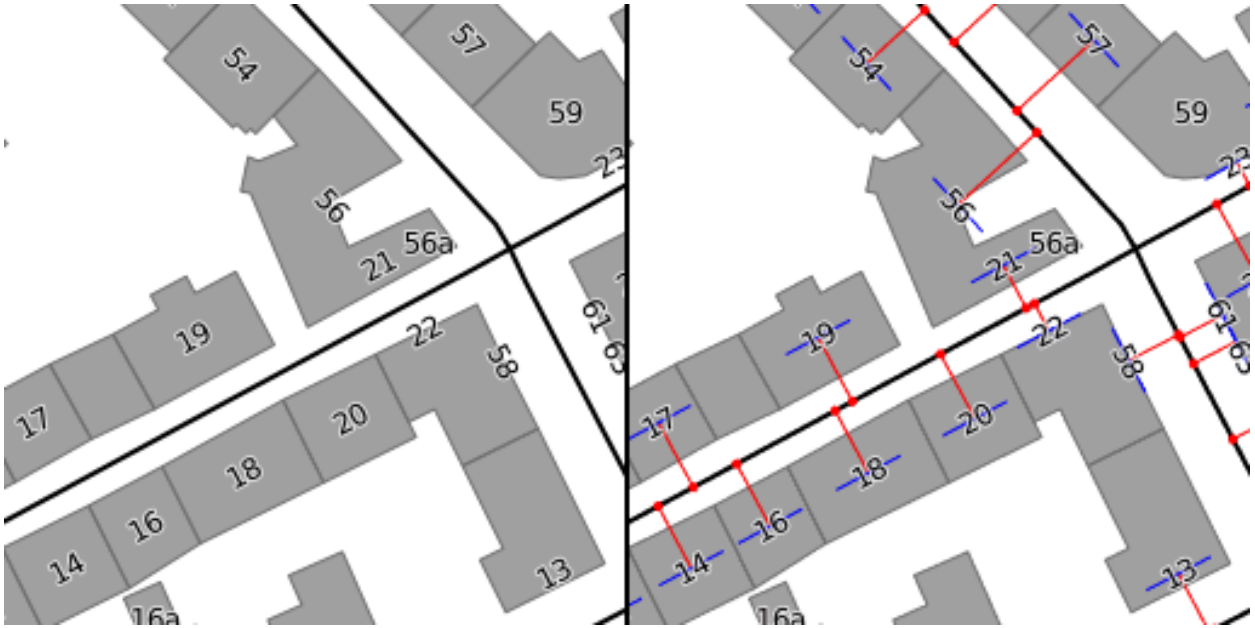
- Select objects “near” other objects:
  ```
  node[natural=peak] near[distance<128]
    node[natural=peak] { … }
  ```
- Compare elevation of peaks (see code)



http://pgmapcss.openstreetbrowser.org/?style=2956e&zoom=12&lat=48.2618&lon=16.2635

pgmapcss – rotate housenumbers

- search for closest road with same name next to house number
- construct a parallel line to road through housenumber node; set text on this line



Final: http://pgmapcss.openstreetbrowser.org/?style=7f913&zoom=18&lat=47.0693&lon=15.4530
Debug: http://pgmapcss.openstreetbrowser.org/?style=9e088&zoom=18&lat=47.0693&lon=15.4530

Future Work

- Get people to use/contribute to pgmapcss ;-)
- Improve compatibility with JOSM
- Try to improve MapCSS standard
- Find a sponsor

Thanks for your attention!

Slides: http://plepe.at/194
pgmapcss: https://github.com/plepe/pgmapcss
All examples:
https://github.com/plepe/pgmapcss/tree/master/examples