OSM-Daten-mit-Mapnik-und-Python-rendern

Hartmut Holzgraefe

hartmut@php.net

FOSSGIS- Mar. 22nd, 2018

Who am I?

- Hartmut Holzgraefe

Who am I?

- Hartmut Holzgraefe
- aus Bielefeld

Who am I?

- Hartmut Holzgraefe
- aus Bielefeld
- Informatiker und Elektro-Ingenieur

Who am I?

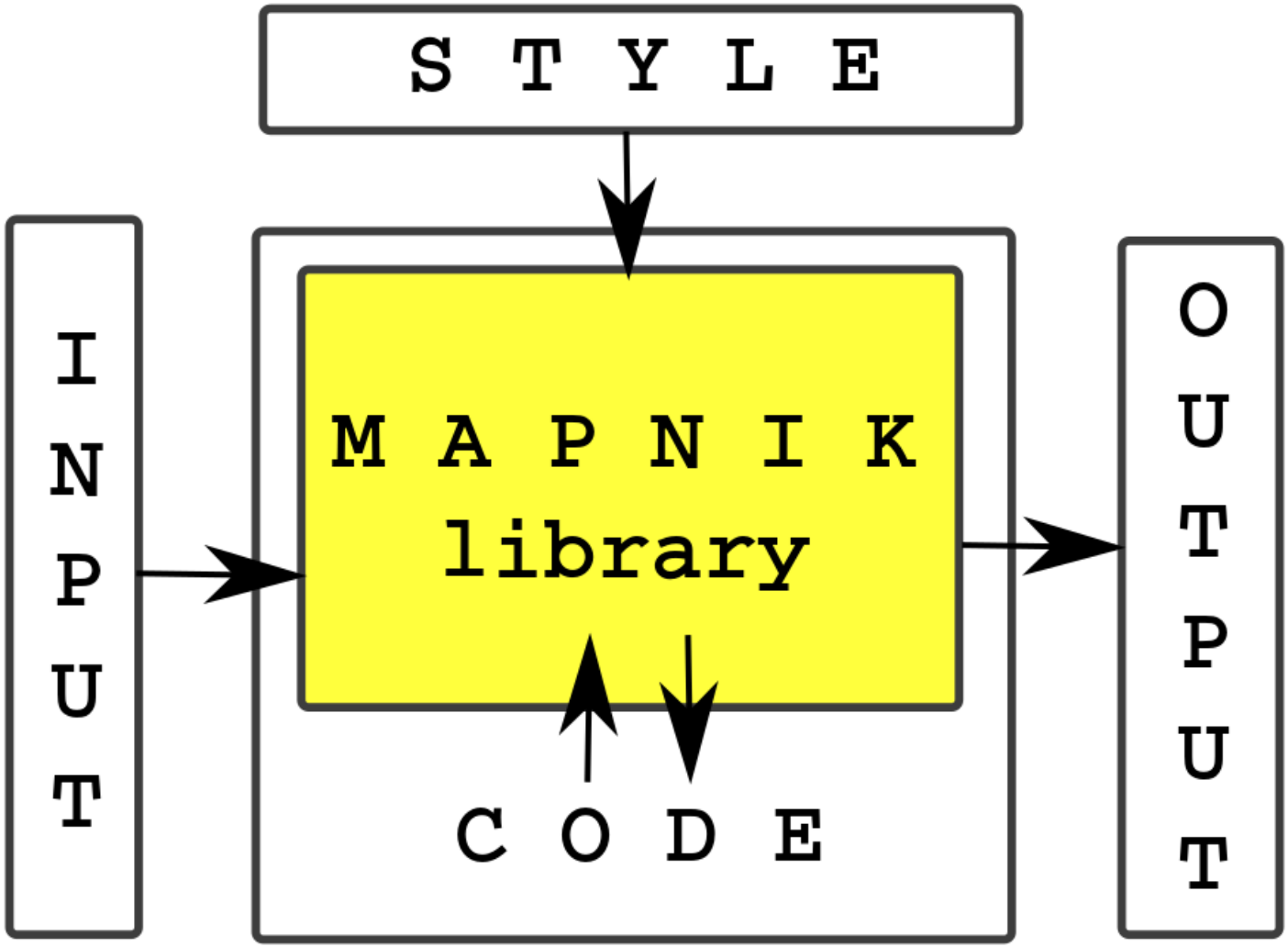
- Hartmut Holzgraefe
- aus Bielefeld
- Informatiker und Elektro-Ingenieur
- OpenStreetMapper seit 2007

Who am I?

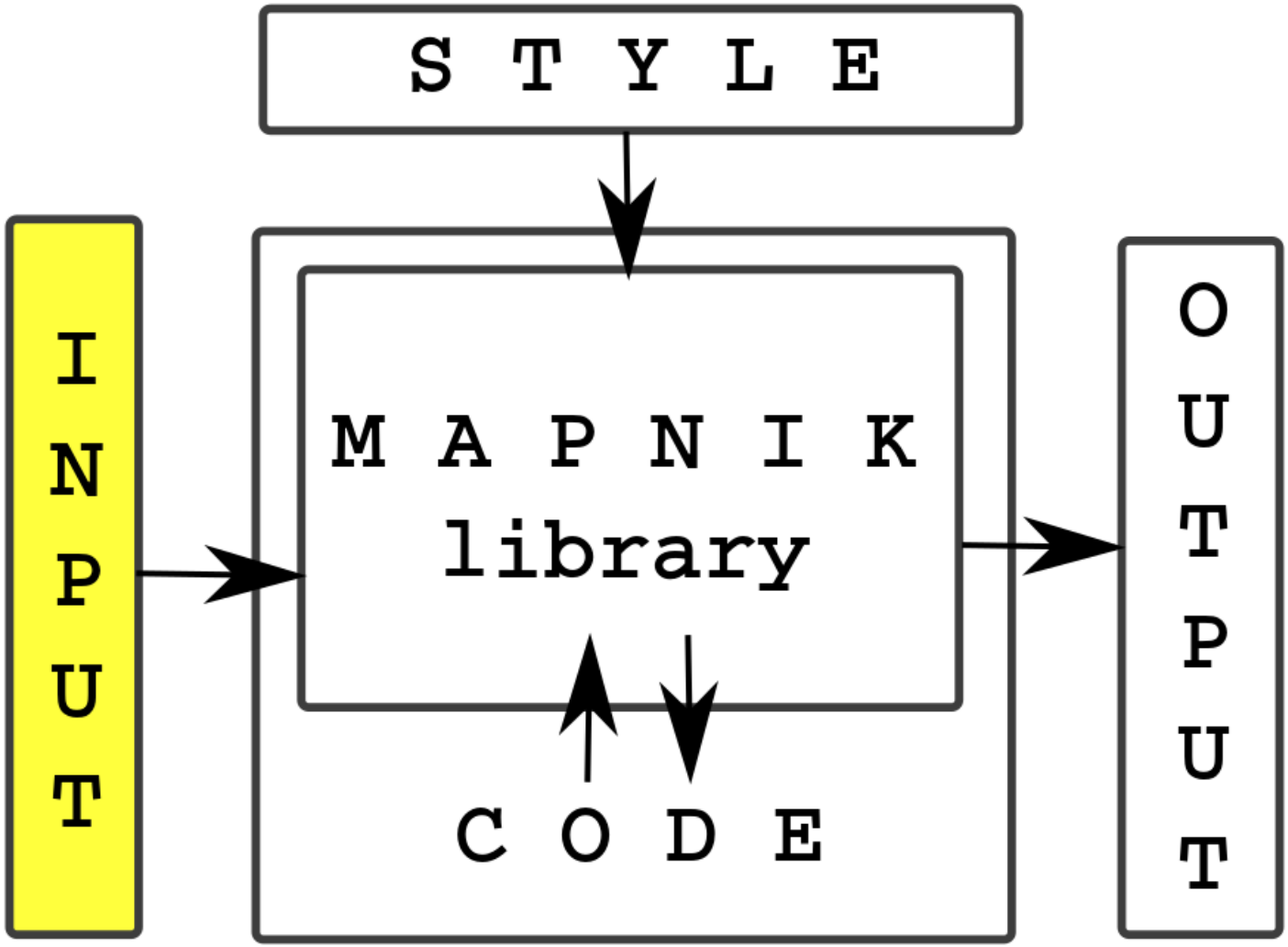
- Hartmut Holzgraefe
- aus Bielefeld
- Informatiker und Elektro-Ingenieur
- OpenStreetMapper seit 2007
- Principal Database Support Engineer at MariaDB Corp.
  (and previously MySQL, Sun, Oracle, SkySQL)

Mapnik Überblick

Mapnik Überblick



Eingabeformate



Datenquellen

Mapnik kann Geodaten aus verschiedensten Quellen lesen:

Datenquellen

Mapnik kann Geodaten aus verschiedensten Quellen lesen:

- Shapefiles

Plugins für weitere Formate

Datenquellen

Mapnik kann Geodaten aus verschiedensten Quellen lesen:

- Shapefiles
- SQL Datenbankabfragen

Plugins für weitere Formate

Datenquellen

Mapnik kann Geodaten aus verschiedensten Quellen lesen:

- Shapefiles
- SQL Datenbankabfragen
- GeoJson

Plugins für weitere Formate

Datenquellen

Mapnik kann Geodaten aus verschiedensten Quellen lesen:

- Shapefiles
- SQL Datenbankabfragen
- GeoJson

Plugins für weitere Formate

Datenquellen

Mapnik kann Geodaten aus verschiedensten Quellen lesen:

- Shapefiles
- SQL Datenbankabfragen
- GeoJson

Plugins für weitere Formate

- ... OGR für verschiedene Vektor- und Rasterformate, z.B. OSM XML und GPX

Datenquellen

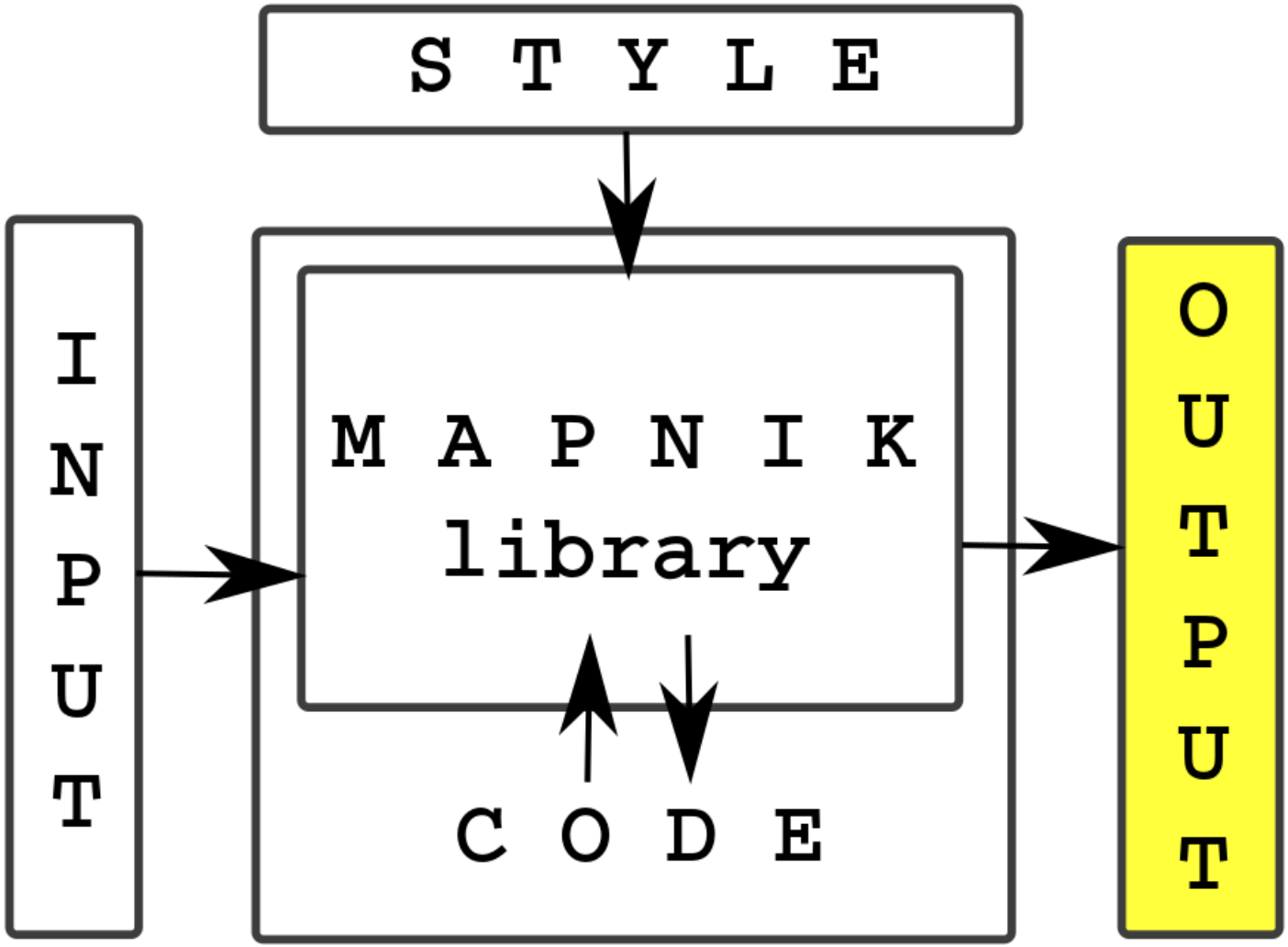
Mapnik kann Geodaten aus verschiedensten Quellen lesen:

- Shapefiles
- SQL Datenbankabfragen
- GeoJson

Plugins für weitere Formate

- ... OGR für verschiedene Vektor- und Rasterformate, z.B. OSM XML und GPX
- ... GDAL für verschiedene Rasterformate

Output



Ausgabeformate

Mapnik kann Karten in verschiedenen Formaten ausgeben:

Ausgabeformate

Mapnik kann Karten in verschiedenen Formaten ausgeben:

- PNG (32bit und 8bit)

Ausgabeformate

Mapnik kann Karten in verschiedenen Formaten ausgeben:

- PNG (32bit und 8bit)
- JPG

Ausgabeformate

Mapnik kann Karten in verschiedenen Formaten ausgeben:

- PNG (32bit und 8bit)
- JPG
- SVG

Ausgabeformate

Mapnik kann Karten in verschiedenen Formaten ausgeben:

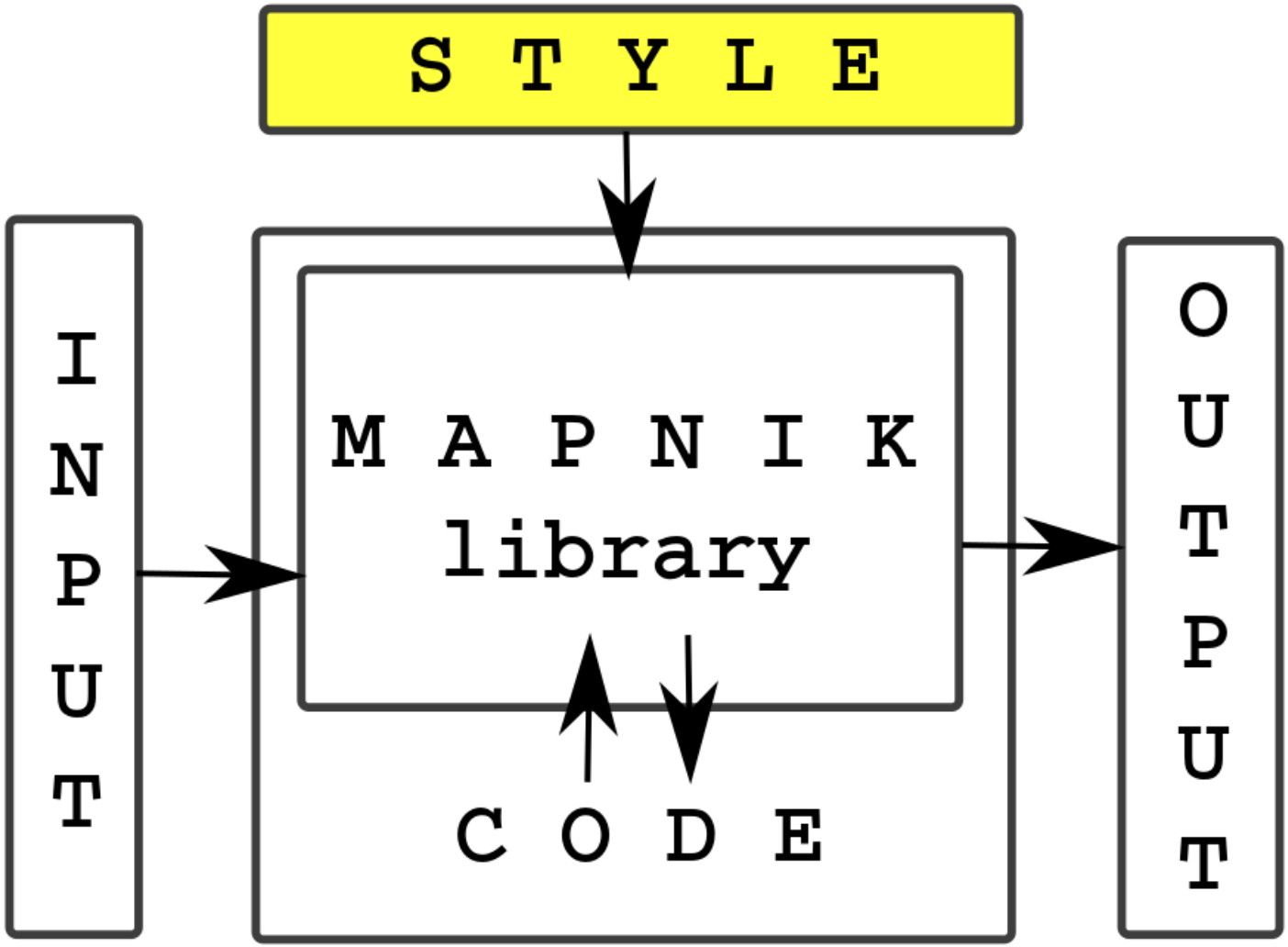
- PNG (32bit und 8bit)
- JPG
- SVG
- PDF

Ausgabeformate

Mapnik kann Karten in verschiedenen Formaten ausgeben:

- PNG (32bit und 8bit)
- JPG
- SVG
- PDF
- PostScript

Kartenstile



Kartenstile

Kartenstile bestimmen wie Kartendaten dargestellt werden:

Kartenstile

Kartenstile bestimmen wie Kartendaten dargestellt werden:

- Stile können in Programmcode definiert werden

Kartenstile

Kartenstile bestimmen wie Kartendaten dargestellt werden:

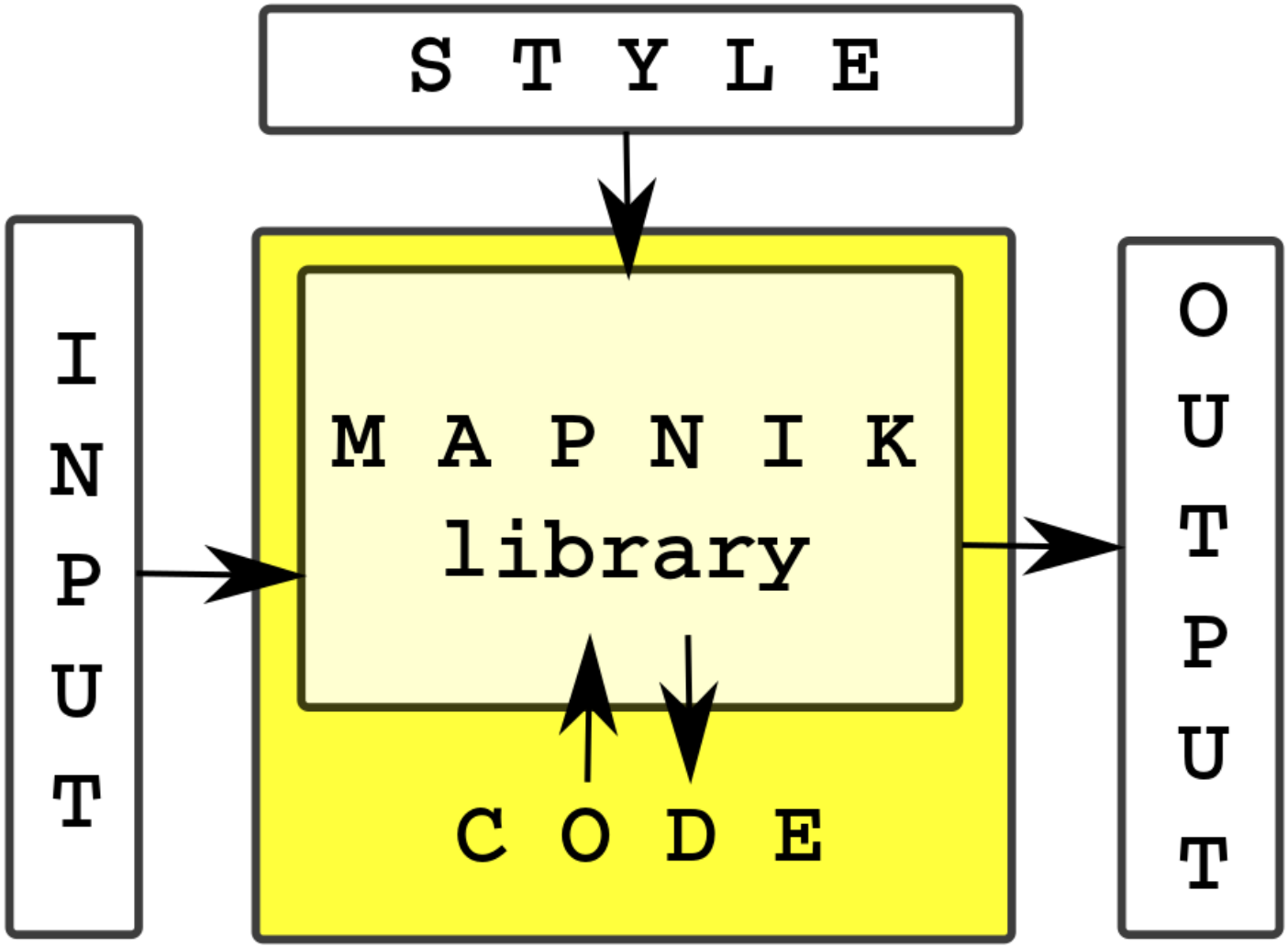
- Stile können in Programmcode definiert werden
- oder als XML-Dateien

Kartenstile

Kartenstile bestimmen wie Kartendaten dargestellt werden:

- Stile können in Programmcode definiert werden
- oder als XML-Dateien
- Bestimmte andere Formate können in Mapnik XML konvertiert werden, vor allem CartoCSS

Code



Control Code

Mapnik ist eine C++ Bibliothek, kein eigenständiges Programm. Wir brauchen also zusätzlichen Code um alles zum Laufen zu bekommen.

Control Code

Mapnik ist eine C++ Bibliothek, kein eigenständiges Programm. Wir brauchen also zusätzlichen Code um alles zum Laufen zu bekommen.

- native C++

Control Code

Mapnik ist eine C++ Bibliothek, kein eigenständiges Programm. Wir brauchen also zusätzlichen Code um alles zum Laufen zu bekommen.

- native C++
- Python bindings

Control Code

Mapnik ist eine C++ Bibliothek, kein eigenständiges Programm. Wir brauchen also zusätzlichen Code um alles zum Laufen zu bekommen.

- native C++
- Python bindings
- Experimentel auch: PHP 7

Voraussetzungen

Wir benötigen:

- Python (2 or 3)
- Mapnik 3 (2?)
- Python bindings für Mapnik, Cairo, and Pango

Installation

Debian/Ubuntu:

```
apt-get install \
  python3-mapnik \
  gir1.2-pango-1.0 \
  gir1.2-rsvg-2.0 \
  python3-gi-cairo
```

Punkte, Linien und Polygone

Punkte, Linien und Polygone

Alle Mapnik Datenquellen leifern Daten als eine Sammlung von:

- Punkten

Punkte, Linien und Polygone

Alle Mapnik Datenquellen leifern Daten als eine Sammlung von:

- Punkten
- Linien

Punkte, Linien und Polygone

Alle Mapnik Datenquellen leifern Daten als eine Sammlung von:

- Punkten
- Linien
- Polygonen

Punkte, Linien und Polygone

Alle Mapnik Datenquellen leifern Daten als eine Sammlung von:

- Punkten
- Linien
- Polygonen
- Rasterbildern

Punkte, Linien und Polygone

Alle Mapnik Datenquellen leifern Daten als eine Sammlung von:

- Punkten
- Linien
- Polygonen
- Rasterbildern

Je nach eigentlicher Datenquelle kann es zu Konvertierungen kommen.

Alle der vier genannten Objektarten können weitere Attribute tragen die zur Auswahl oder Darstellung herangezogen werden können, zB “name”.

Layers, Styles und Symbolizers

Layers

Ein Mapnik Layer importiert Daten, oder Teile davon, aus einer der verfügbaren Datenquellen und verknüpft diese zur Darstellung mit einem oder mehreren Darstellungsstilen.

Styles

Ein Style kann importierte Layer Daten weiter filtern und legt fest welche Symbolizer zur konkreten Darstellung verwendet werden sollen.

Symbolizers

Die Symbolizer übernehmen die eigentliche grafische Ausgabe. Hierfür gibt es vier grundlegende Symbolizer für die vier Objektarten, und verschiedene weitere für komplexere Aufgaben:

- PointSymbolizer

Symbolizers

Die Symbolizer übernehmen die eigentliche grafische Ausgabe. Hierfür gibt es vier grundlegende Symbolizer für die vier Objektarten, und verschiedene weitere für komplexere Aufgaben:

- PointSymbolizer
- LineSymbolizer

Symbolizers

Die Symbolizer übernehmen die eigentliche grafische Ausgabe. Hierfür gibt es vier grundlegende Symbolizer für die vier Objektarten, und verschiedene weitere für komplexere Aufgaben:

- PointSymbolizer
- LineSymbolizer
- PolygonSymbolizer

Symbolizers

Die Symbolizer übernehmen die eigentliche grafische Ausgabe. Hierfür gibt es vier grundlegende Symbolizer für die vier Objektarten, und verschiedene weitere für komplexere Aufgaben:

- PointSymbolizer
- LineSymbolizer
- PolygonSymbolizer
- RasterSymbolizer

Symbolizers (Fortsetzung)

Weitere Symbolizer:

- MarkerSymbolizer

Symbolizers (Fortsetzung)

Weitere Symbolizer:

- MarkerSymbolizer
- LinePatterSymbolizer

Symbolizers (Fortsetzung)

Weitere Symbolizer:

- MarkerSymbolizer
- LinePatterSymbolizer
- TextSymbolizer

Symbolizers (Fortsetzung)

Weitere Symbolizer:

- MarkerSymbolizer
- LinePatterSymbolizer
- TextSymbolizer
- ShieldSymbolizer

Symbolizers (Fortsetzung)

Weitere Symbolizer:

- MarkerSymbolizer
- LinePatterSymbolizer
- TextSymbolizer
- ShieldSymbolizer
- PolygonPatternSymbolizer

Symbolizers (Fortsetzung)

Weitere Symbolizer:

- MarkerSymbolizer
- LinePatterSymbolizer
- TextSymbolizer
- ShieldSymbolizer
- PolygonPatternSymbolizer
- BuildingSymbolizer

Code basics

Ein minimales Beispiel

```
import mapnik
```

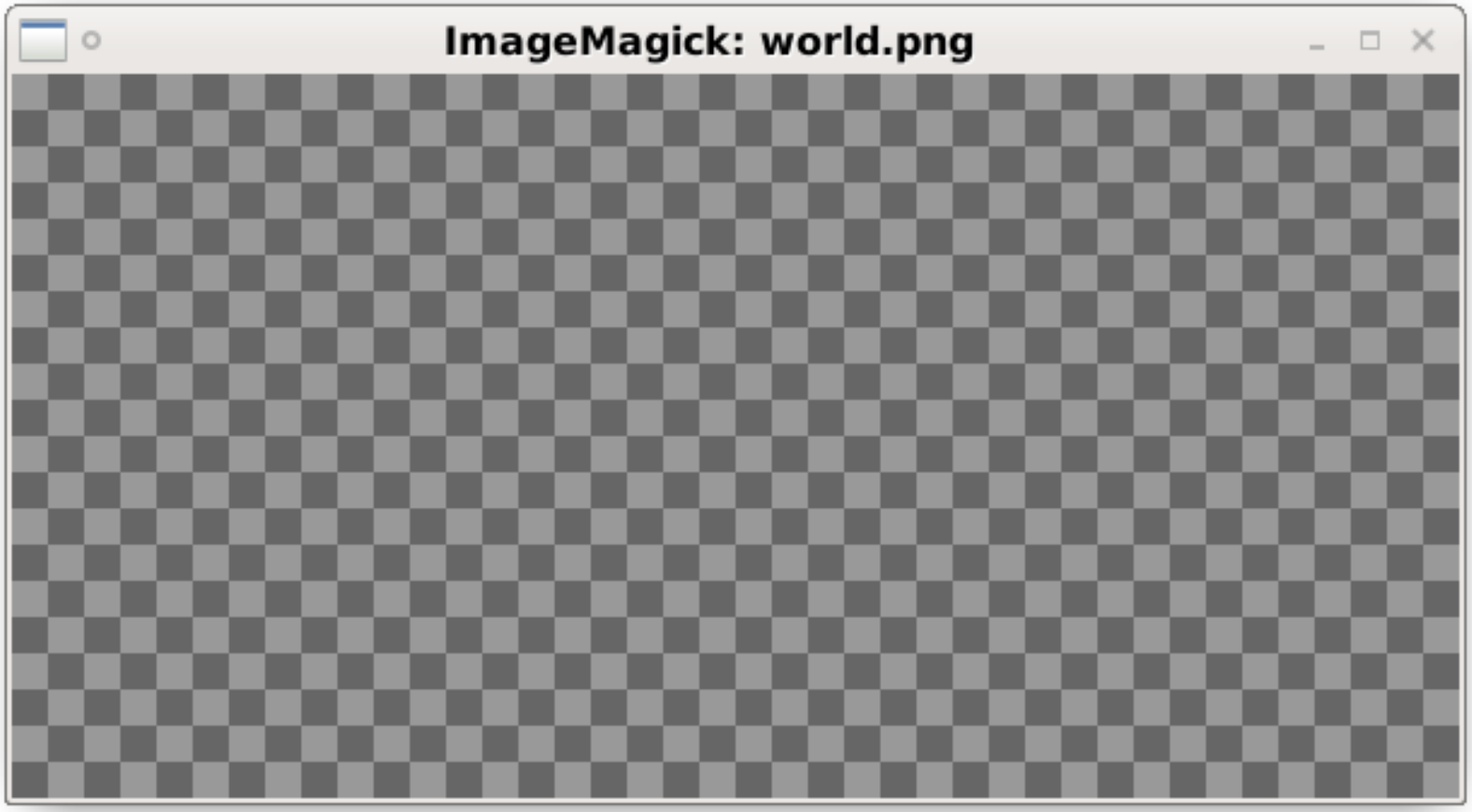
Ein minimales Beispiel

```
1 import mapnik
2
3 map = mapnik.Map(600,300)
```

Ein minimales Beispiel

```
1 import mapnik
2
3 map = mapnik.Map(600,300)
4
5 mapnik.render_to_file(map, 'world.png', 'png')
```

... und das Ergebnis



Ein minimales 'Hello World!'

```
1 import mapnik
2
3 map = mapnik.Map(600,300)
4 map.background = mapnik.Color('steelblue')
```

Ein minimales 'Hello World!'

```
1 import mapnik
2
3 map = mapnik.Map(600,300)
4 map.background = mapnik.Color('steelblue')
5
6 polygons = mapnik.PolygonSymbolizer()
7 polygons.fill = mapnik.Color('lightgreen')
```

Ein minimales 'Hello World!'

```
import mapnik

map = mapnik.Map(600,300)
map.background = mapnik.Color('steelblue')

polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('lightgreen')

rules = mapnik.Rule()
rules.symbols.append(polygons)
```

Ein minimales 'Hello World!'

```
import mapnik

map = mapnik.Map(600,300)
map.background = mapnik.Color('steelblue')

polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('lightgreen')

rules = mapnik.Rule()
rules.symbols.append(polygons)

style = mapnik.Style()
style.rules.append(rules)
map.append_style('Countries', style)
```

Ein minimales 'Hello World!'

```
import mapnik

map = mapnik.Map(600,300)
map.background = mapnik.Color('steelblue')

polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('lightgreen')

rules = mapnik.Rule()
rules.symbols.append(polygons)

style = mapnik.Style()
style.rules.append(rules)
map.append_style('Countries', style)

layer = mapnik.Layer('world')
layer.datasource = mapnik.Shapefile(file='countries.shp')
layer.styles.append('Countries')
```

Ein minimales 'Hello World!'

```
import mapnik

map = mapnik.Map(600,300)
map.background = mapnik.Color('steelblue')

polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('lightgreen')

rules = mapnik.Rule()
rules.symbols.append(polygons)

style = mapnik.Style()
style.rules.append(rules)
map.append_style('Countries', style)

layer = mapnik.Layer('world')
layer.datasource = mapnik.Shapefile(file='countries.shp')
layer.styles.append('Countries')

map.layers.append(layer)
```

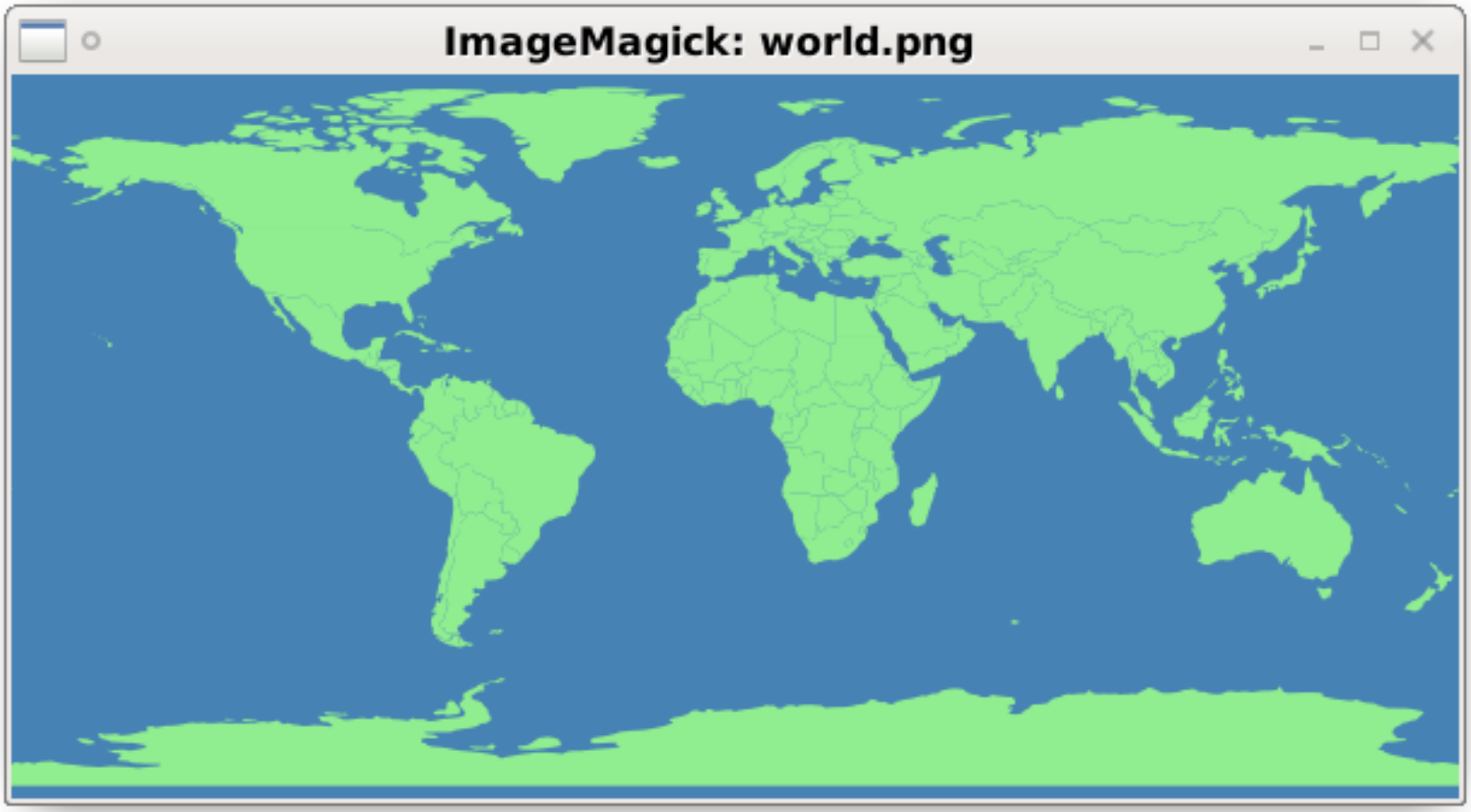
Ein minimales 'Hello World!'

```
import mapnik

map = mapnik.Map(600,300)
map.background = mapnik.Color('steelblue')

polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('lightgreen')

rules = mapnik.Rule()
rules.symbols.append(polygons)

style = mapnik.Style()
style.rules.append(rules)
map.append_style('Countries', style)

layer = mapnik.Layer('world')
layer.datasource = mapnik.Shapefile(file='countries.shp')
layer.styles.append('Countries')

map.layers.append(layer)
map.zoom_all()
```

Ein minimales 'Hello World!'

```
import mapnik

map = mapnik.Map(600,300)
map.background = mapnik.Color('steelblue')

polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('lightgreen')

rules = mapnik.Rule()
rules.symbols.append(polygons)

style = mapnik.Style()
style.rules.append(rules)
map.append_style('Countries', style)

layer = mapnik.Layer('world')
layer.datasource = mapnik.Shapefile(file='countries.shp')
layer.styles.append('Countries')

map.layers.append(layer)
map.zoom_all()
mapnik.render_to_file(map, 'world.png', 'png')
```

... und das Ergebnis



Finde Deutschland

```
[...]
polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('green')
polygons.gamma = 0.0

```

Finde Deutschland

```
[...]
polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('green')
polygons.gamma = 0.0

rules.symbols.append(polygons)
style.rules.append(rules)
```

Finde Deutschland

```
[...]
polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('green')
polygons.gamma = 0.0

rules.symbols.append(polygons)
style.rules.append(rules)

highlight = mapnik.PolygonSymbolizer()
highlight.fill = mapnik.Color('red')
```

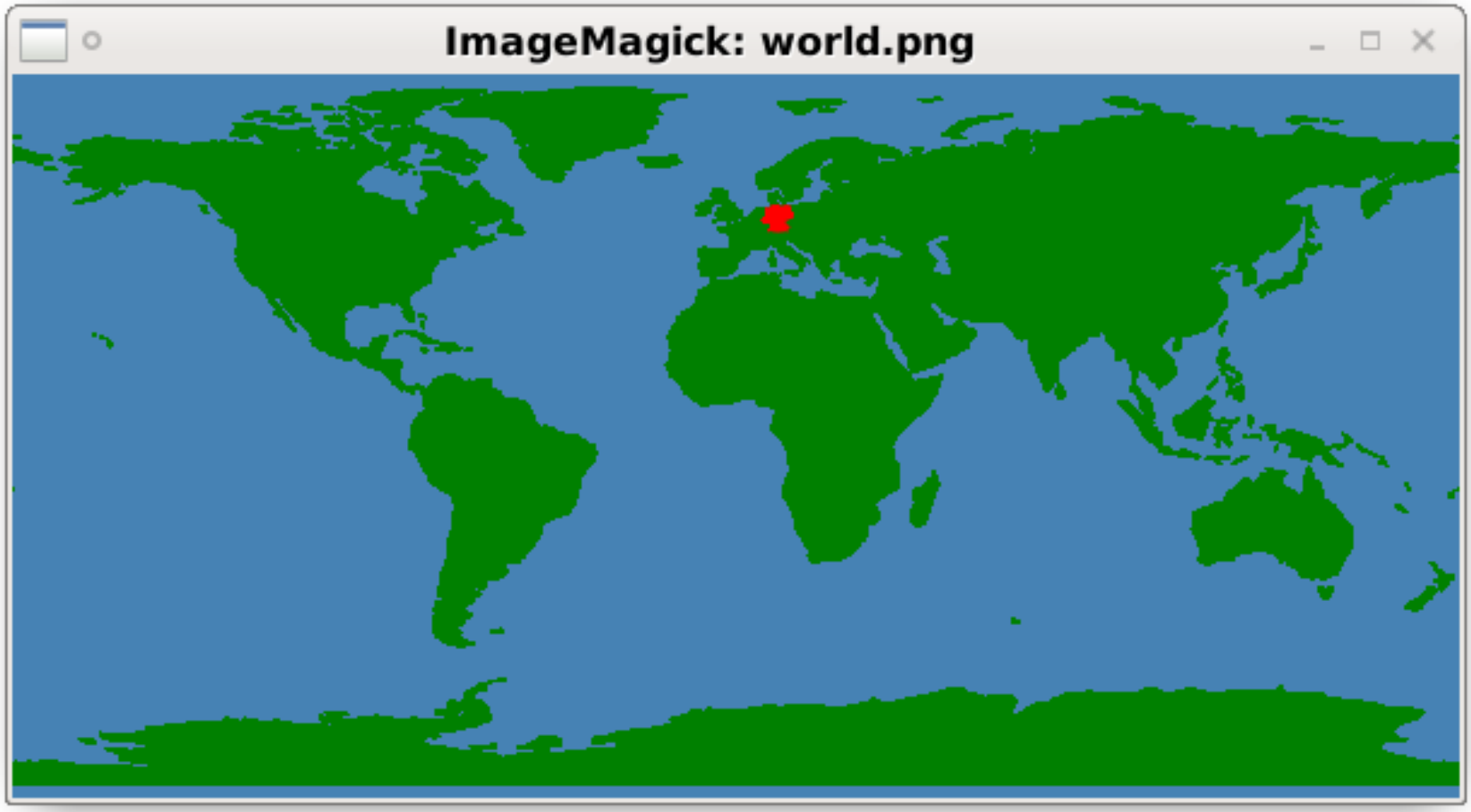
Finde Deutschland

```
[...]
polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('green')
polygons.gamma = 0.0

rules.symbols.append(polygons)
style.rules.append(rules)

highlight = mapnik.PolygonSymbolizer()
highlight.fill = mapnik.Color('red')

germany = mapnik.Rule()
germany.filter = mapnik.Expression("[NAME] = 'Germany'")
germany.symbols.append(highlight)
```

Finde Deutschland

```
[...]
polygons = mapnik.PolygonSymbolizer()
polygons.fill = mapnik.Color('green')
polygons.gamma = 0.0

rules.symbols.append(polygons)
style.rules.append(rules)

highlight = mapnik.PolygonSymbolizer()
highlight.fill = mapnik.Color('red')

germany = mapnik.Rule()
germany.filter = mapnik.Expression("[NAME] = 'Germany'")
germany.symbols.append(highlight)

style.rules.append(germany)
map.append_style('Countries', style)
[...]
```

... und das Ergebnis



Und jetzt mit XML

```
1 import mapnik
2
3 map = mapnik.Map(600,300)
```

Und jetzt mit XML

```
1 import mapnik
2
3 map = mapnik.Map(600,300)
4
5 mapnik.load_map(m, 'world.xml')
6
```

Und jetzt mit XML

```
1 import mapnik
2
3 map = mapnik.Map(600,300)
4
5 mapnik.load_map(m, 'world.xml')
6
7 map.zoom_all()
8
9 mapnik.render_to_file(map, 'world.png', 'png')
```

XML style definition

```
<?xml version="1.0" encoding="utf-8"?>
```

XML style definition

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <Map background-color='steelblue'>
```

XML style definition

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='steelblue'>
  <Style name="Countries">
    <Rule>
      <PolygonSymbolizer fill="green" gamma="0.0"/>
    </Rule>
```

XML style definition

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='steelblue'>
  <Style name="Countries">
    <Rule>
      <PolygonSymbolizer fill="green" gamma="0.0"/>
    </Rule>
    <Rule>
      <Filter>([NAME]='Germany')</Filter>
      <PolygonSymbolizer fill="red"/>
    </Rule>
  </Style>
```

XML style definition

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='steelblue'>
  <Style name="Countries">
    <Rule>
      <PolygonSymbolizer fill="green" gamma="0.0"/>
    </Rule>
    <Rule>
      <Filter>([NAME]='Germany')</Filter>
      <PolygonSymbolizer fill="red"/>
    </Rule>
  </Style>
  <Layer name="world">
```

XML style definition

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='steelblue'>
  <Style name="Countries">
    <Rule>
      <PolygonSymbolizer fill="green" gamma="0.0"/>
    </Rule>
    <Rule>
      <Filter>([NAME]='Germany')</Filter>
      <PolygonSymbolizer fill="red"/>
    </Rule>
  </Style>
  <Layer name="world">
    <StyleName>Borders</StyleName>
```

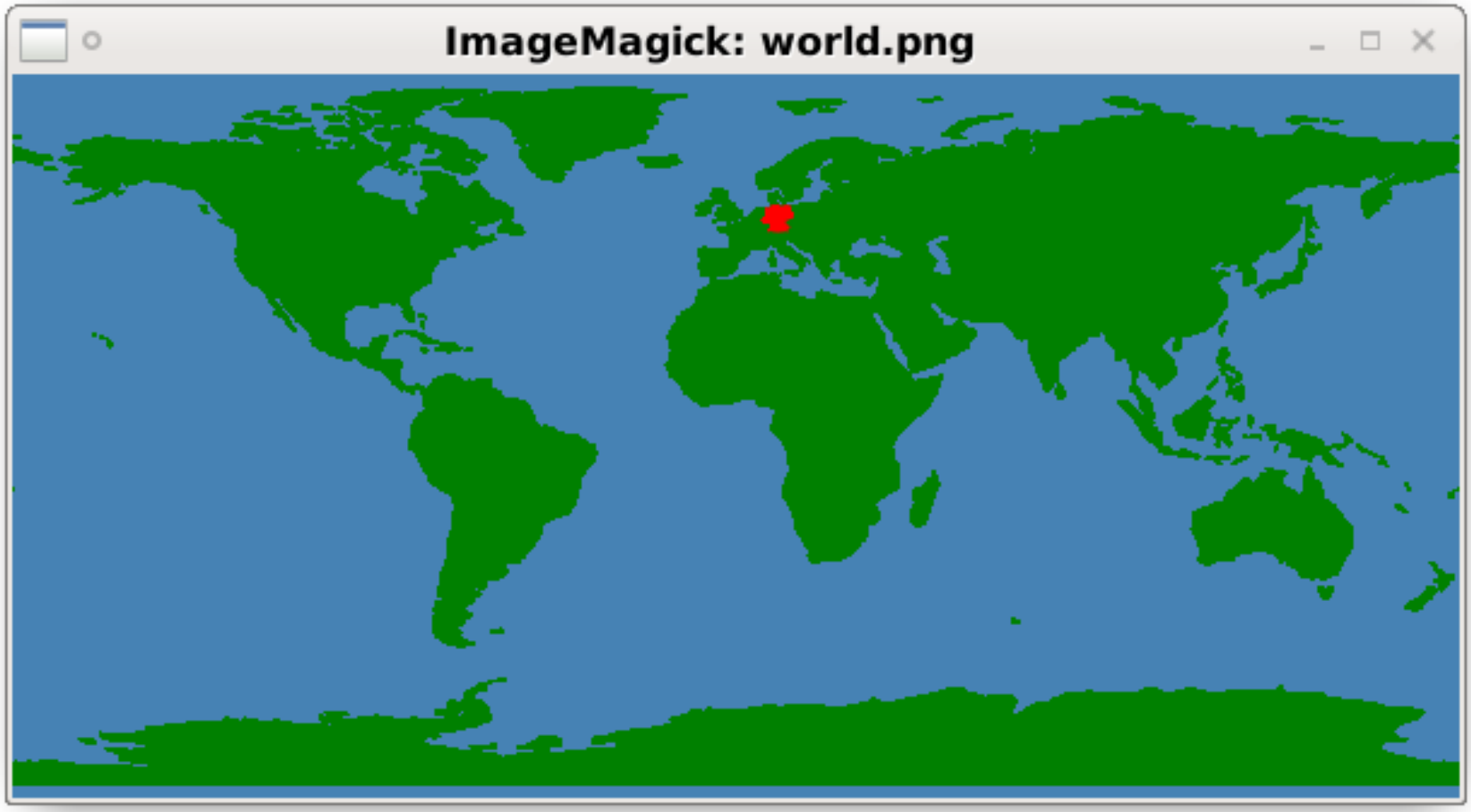
XML style definition

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='steelblue'>
  <Style name="Countries">
    <Rule>
      <PolygonSymbolizer fill="green" gamma="0.0"/>
    </Rule>
    <Rule>
      <Filter>([NAME]='Germany')</Filter>
      <PolygonSymbolizer fill="red"/>
    </Rule>
  </Style>
  <Layer name="world">
    <StyleName>Borders</StyleName>
    <Datasource>
      <Parameter name="file"countries.shp</Parameter>
      <Parameter name="type">shape</Parameter>
    </Datasource>
```

XML style definition

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='steelblue'>
  <Style name="Countries">
    <Rule>
      <PolygonSymbolizer fill="green" gamma="0.0"/>
    </Rule>
    <Rule>
      <Filter>([NAME]='Germany')</Filter>
      <PolygonSymbolizer fill="red"/>
    </Rule>
  </Style>
  <Layer name="world">
    <StyleName>Borders</StyleName>
    <Datasource>
      <Parameter name="file"countries.shp</Parameter>
      <Parameter name="type">shape</Parameter>
    </Datasource>
  </Layer>
</Map>
```

... und das Ergebnis



Symbolizer einsetzen

Ein neues Codegerüst

```
import mapnik

map = mapnik.Map(600,300)

mapnik.load_map(map, 'example.xml')

map.zoom_all() # zoom to fit
map.zoom(-1.1) # zoom out 10% more

mapnik.render_to_file(map, 'world.png', 'png')
```

Point Symbolizer

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='white'>
  <Style name='point'>
    <Rule>
      <PointSymbolizer file='point.png'/>
    </Rule>
  </Style>
```

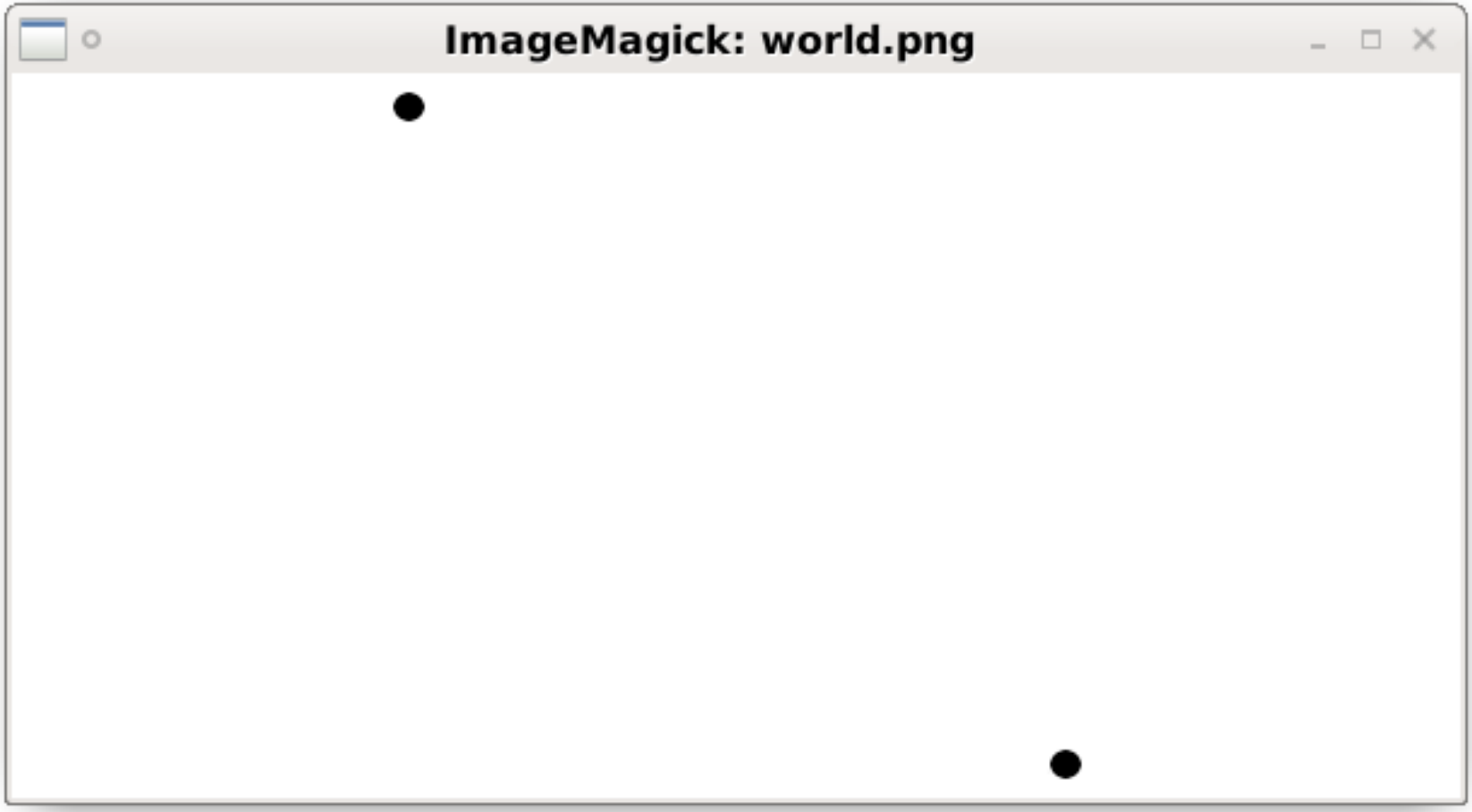
Point Symbolizer

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='white'>
  <Style name='point'>
    <Rule>
      <PointSymbolizer file='point.png'/>
    </Rule>
  </Style>

  <Layer name="test">
   <StyleName>point</StyleName>
   <Datasource>
    <Parameter name='type'>geojson</Parameter>
    <Parameter name='file'>ex1.geojson</Parameter>
   </Datasource>
  </Layer>
</Map>
```

Point Data

```
{
  "type": "FeatureCollection",
  "features": [
    { "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [ 12.54, 55.69 ]
      }
    },
    { "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [ 12.55, 55.68 ]
      }
    }
  ]
}
```

Ergebnis



Line und TextSymbolizer

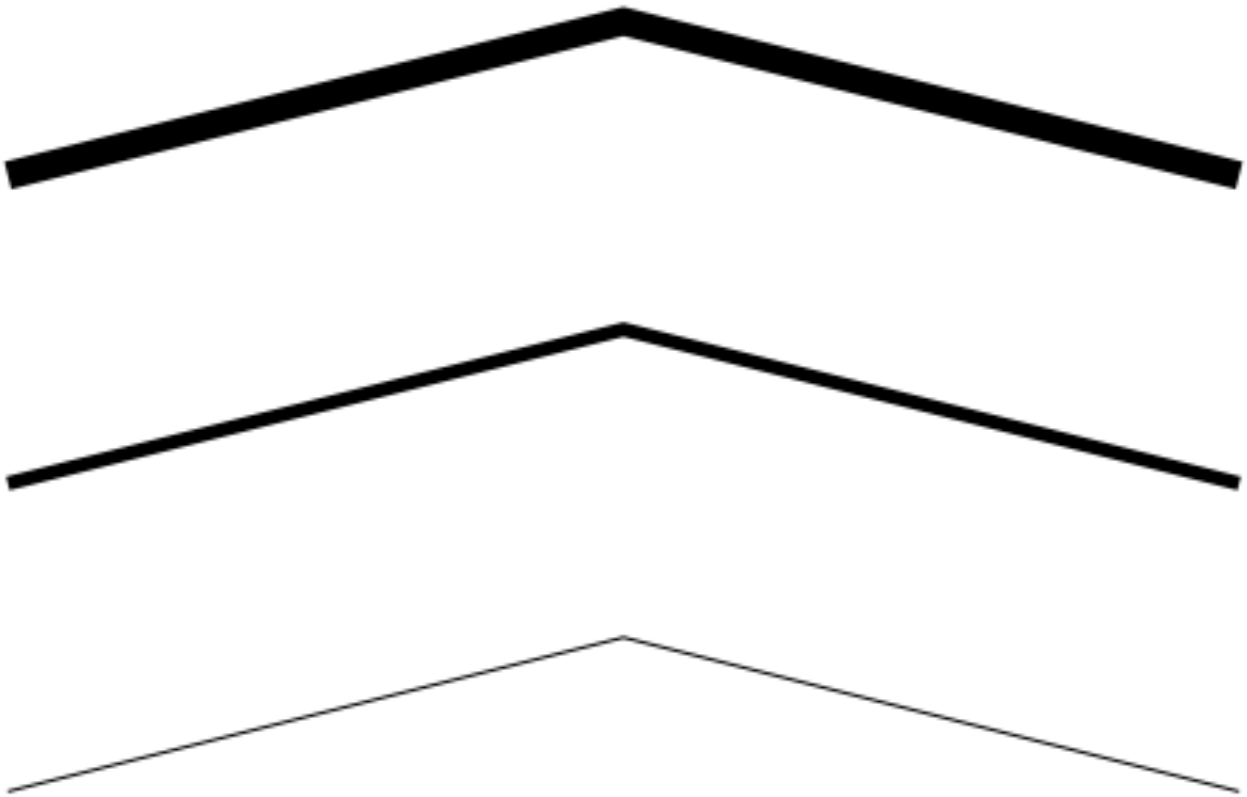
```
{
  "type": "FeatureCollection",
  "features": [
    { "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [10, 10], [20, 20], [30, 40]
        ]
      },
      "properties": {
        "name": "Teststreet"
      }
    }
  ]
}
```

Line und Text Symbolizer

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='white'>
  <Style name='line'>
    <Rule>
      <LineSymbolizer stroke='steelblue' stroke-width='30'>
      <TextSymbolizer placement="line" face-name="DejaVu␣Sans␣Book"
                      size="30" fill="black"
                      halo-fill="white" halo-radius="1"
      >[name]</TextSymbolizer>
    </Rule>
  </Style>

  <Layer name="test">
   <StyleName>line</StyleName>
   [...]
```

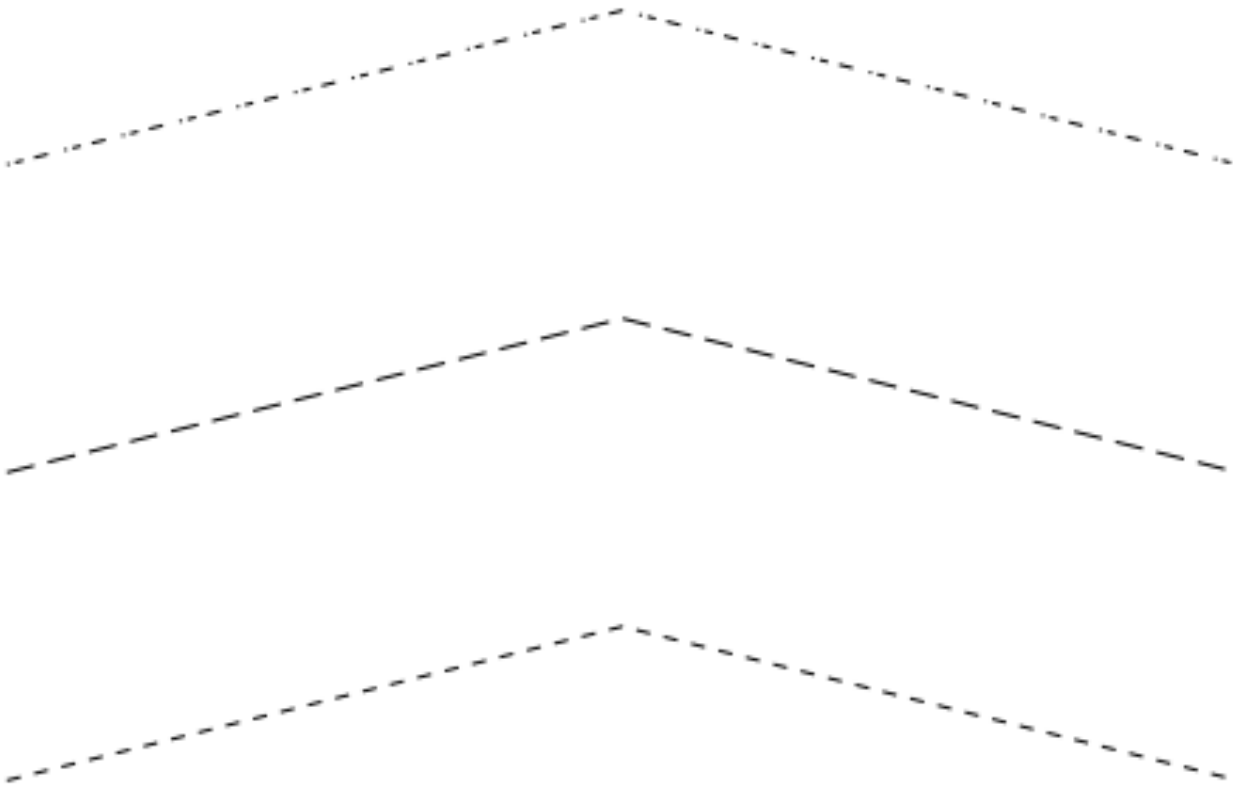
Result

LineSymbolier - width



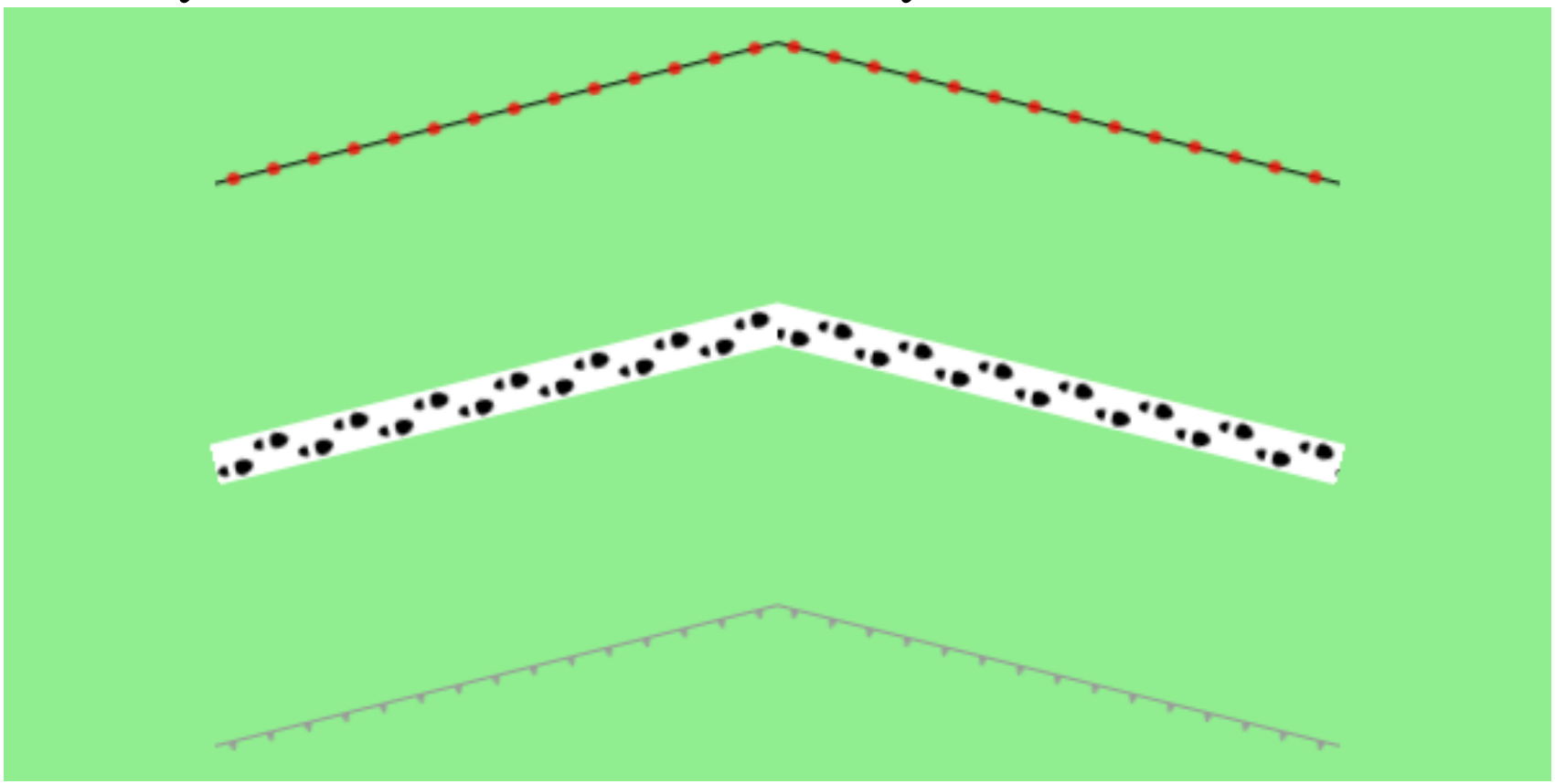
LineSymbolizer - dashes

```
<LineSymbolizer stroke-dasharray="1,2,3,4,5,6"/>
```

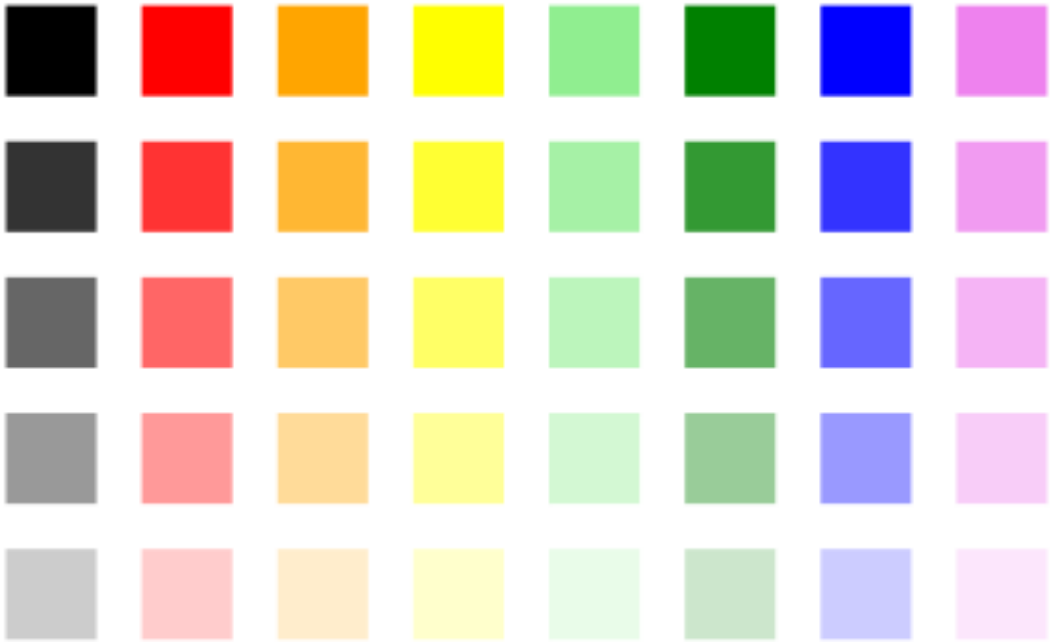


LinePaternSymbolizer

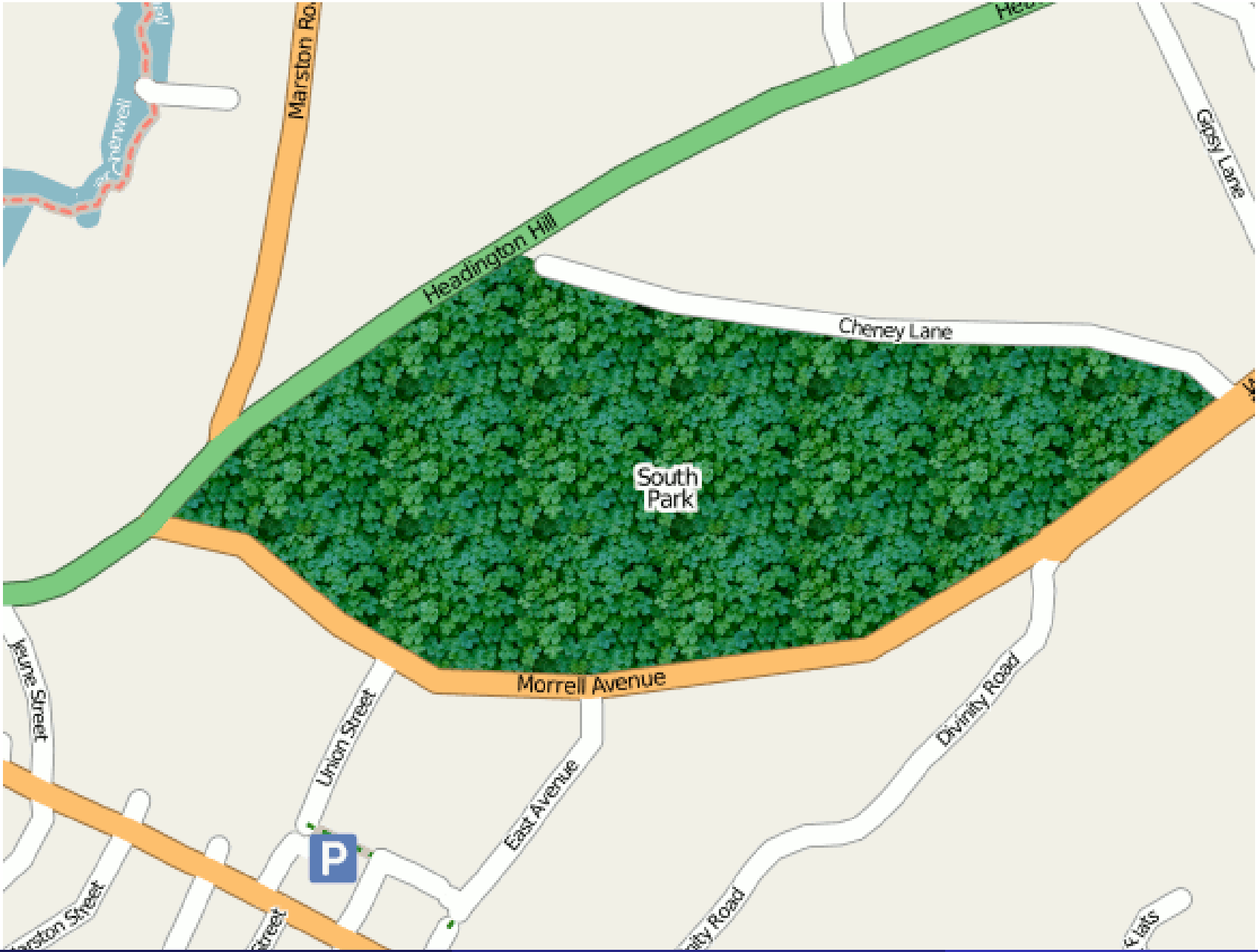
```
<LineSymbolizer stroke-dasharray="1,2,3,4,5,6"/>
```



PolygonSymbolizer - fill / opacity

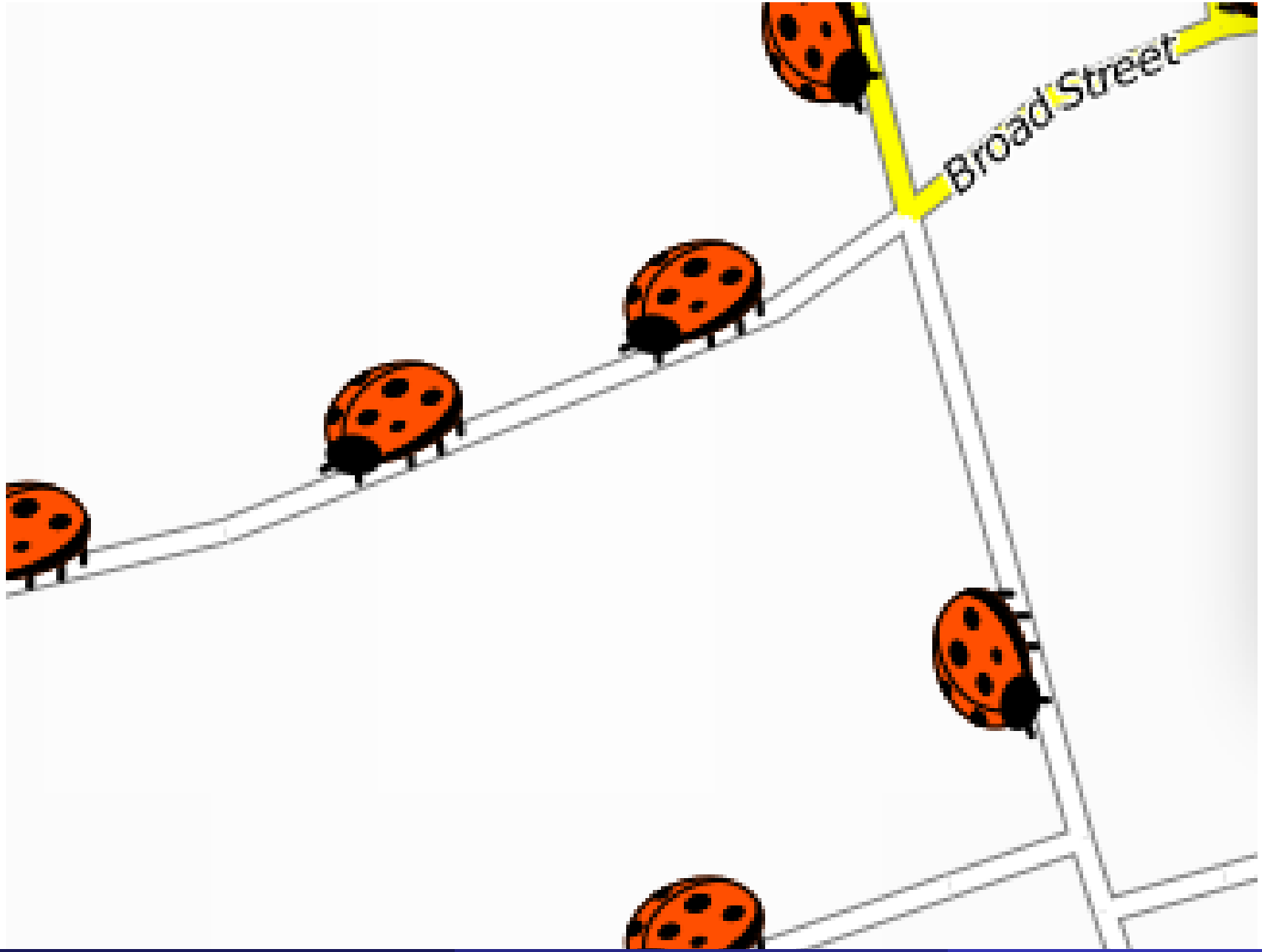


PolygonPatternSymbolizer

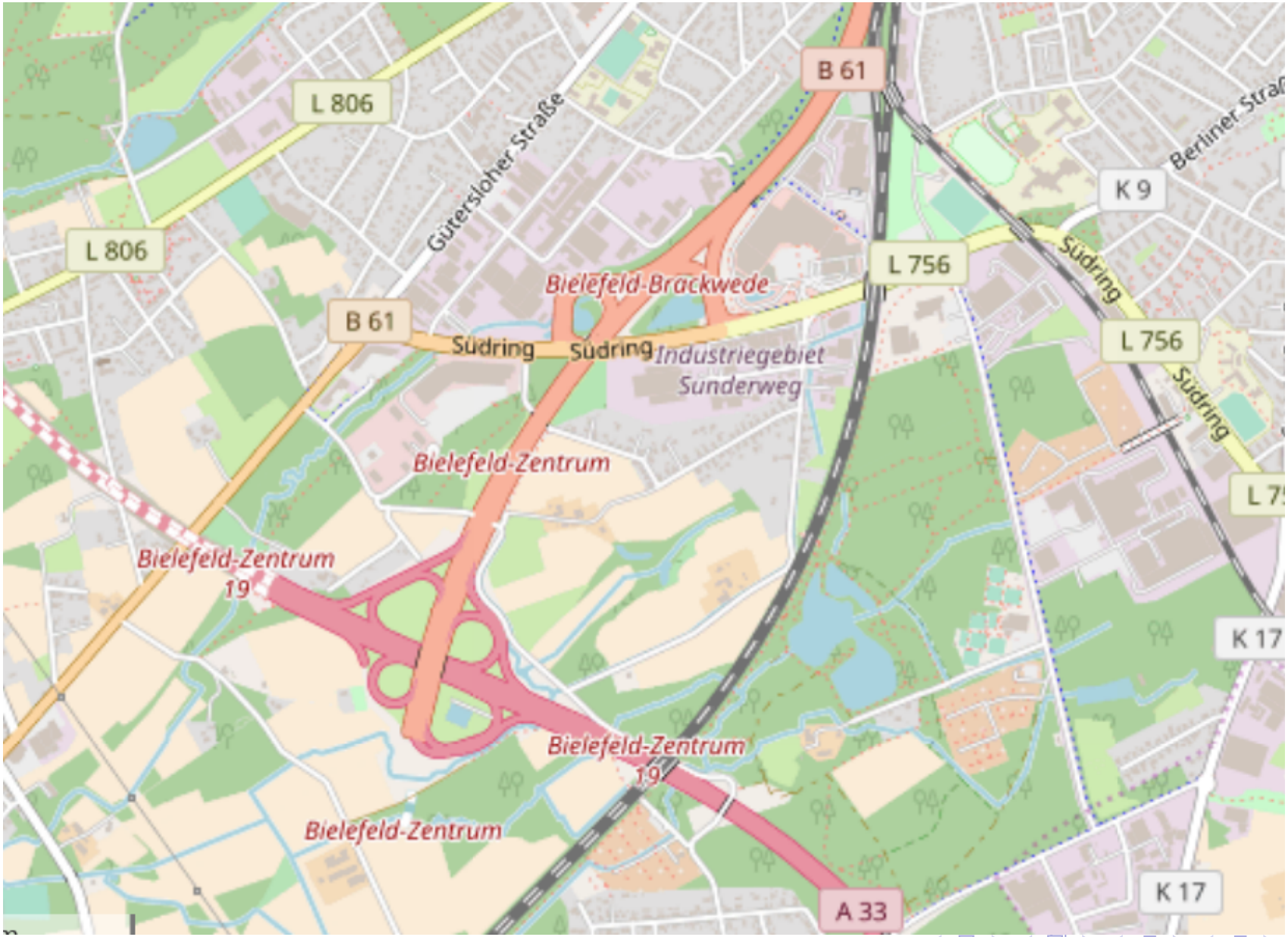


BuildingSymbolizer

MarkerSymbolizer



ShieldSymbolizer



Auf und neben der Karte malen

Zeichnen in Cairo-Kontexten

```
1 import mapnik
2 import cairo
3
4 surface = cairo.PDFSurface('world.pdf', 600, 300)
5 context = cairo.Context(surface)
6
```

Zeichnen in Cairo-Kontexten

```
import mapnik
import cairo

surface = cairo.PDFSurface('world.pdf', 600, 300)
context = cairo.Context(surface)

map = mapnik.Map(600,300)
mapnik.load_map(map, 'world.xml')
map.zoom_all()
map.zoom(-1.1)
```

Zeichnen in Cairo-Kontexten

```
import mapnik
import cairo

surface = cairo.PDFSurface('world.pdf', 600, 300)
context = cairo.Context(surface)

map = mapnik.Map(600,300)
mapnik.load_map(map, 'world.xml')
map.zoom_all()
map.zoom(-1.1)

mapnik.render(map, surface)
```

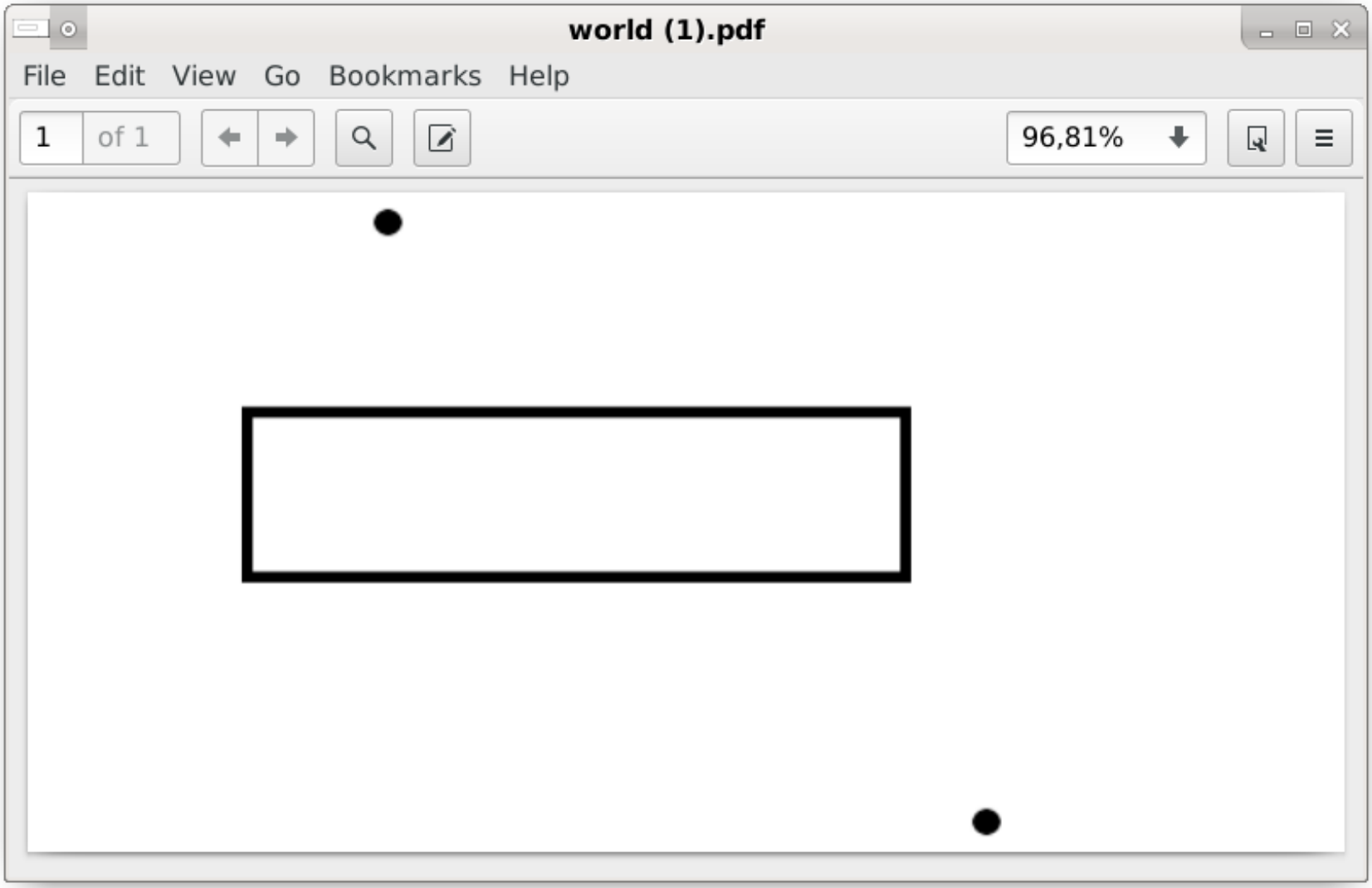
Zeichnen in Cairo-Kontexten

```
import mapnik
import cairo

surface = cairo.PDFSurface('world.pdf', 600, 300)
context = cairo.Context(surface)

map = mapnik.Map(600,300)
mapnik.load_map(map, 'world.xml')
map.zoom_all()
map.zoom(-1.1)

mapnik.render(map, surface)

context.set_source_rgb(0, 0, 0)
context.set_line_width(5)
context.rectangle(100,100,300,75)
context.stroke()
```

Zeichnen in Cairo-Kontexten

```
import mapnik
import cairo

surface = cairo.PDFSurface('world.pdf', 600, 300)
context = cairo.Context(surface)

map = mapnik.Map(600,300)
mapnik.load_map(map, 'world.xml')
map.zoom_all()
map.zoom(-1.1)

mapnik.render(map, surface)

context.set_source_rgb(0, 0, 0)
context.set_line_width(5)
context.rectangle(100,100,300,75)
context.stroke()

surface.finish()
```

Ergebnis



SVGs hinzufügen (v2)

```
1 import mapnik
2 import cairo
3 import rsvg
4
```

SVGs hinzufügen (v2)

```
1 import mapnik
2 import cairo
3 import rsvg
4
5 surface = cairo.PDFSurface('world.pdf', 600, 300)
6 context = cairo.Context(surface)
7
```

SVGs hinzufügen (v2)

```
import mapnik
import cairo
import rsvg

surface = cairo.PDFSurface('world.pdf', 600, 300)
context = cairo.Context(surface)

map = mapnik.Map(600,300)
[...]
mapnik.render(map, surface)
```

SVGs hinzufügen (v2)

```
import mapnik
import cairo
import rsvg

surface = cairo.PDFSurface('world.pdf', 600, 300)
context = cairo.Context(surface)

map = mapnik.Map(600,300)
[...]
mapnik.render(map, surface)

svg = rsvg.Handle('compass.svg')
context.move_to(10,10)
context.scale(0.5, 0.5)
svg.render_cairo(context)
```

SVGs hinzufügen (v2)

```
import mapnik
import cairo
import rsvg

surface = cairo.PDFSurface('world.pdf', 600, 300)
context = cairo.Context(surface)

map = mapnik.Map(600,300)
[...]
mapnik.render(map, surface)

svg = rsvg.Handle('compass.svg')
context.move_to(10,10)
context.scale(0.5, 0.5)
svg.render_cairo(context)

surface.finish()
```

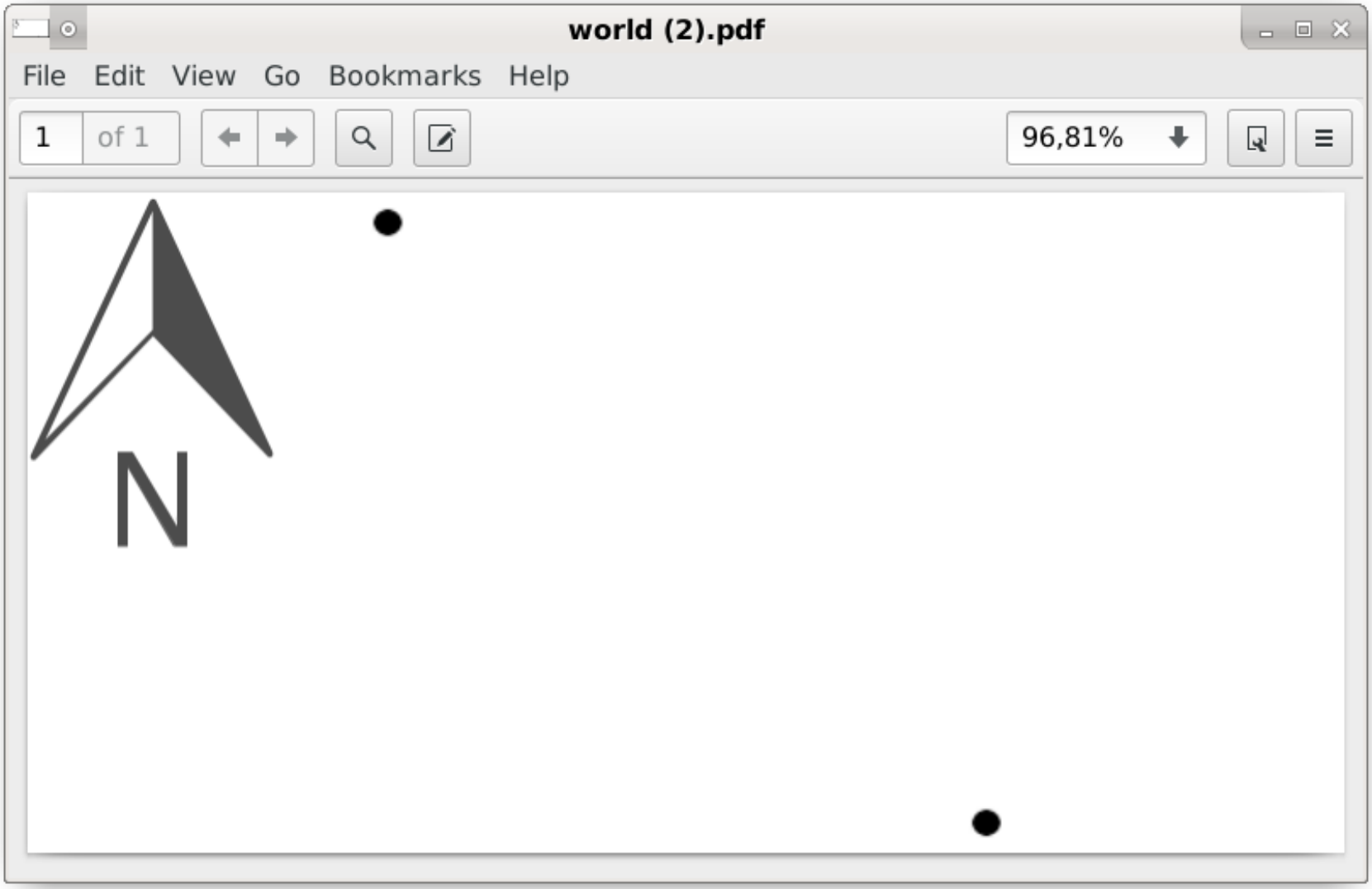
SVGs hinzufügen (v3)

```
1 import mapnik
2 import cairo
3 import gi
4 gi.require_version('Rsvg', '2.0')
5 from gi.repository import Rsvg
6
```

SVGs hinzufügen (v3)

```
import mapnik
import cairo
import gi
gi.require_version('Rsvg', '2.0')
from gi.repository import Rsvg

[...]
rsvg = rsvg.Handle()
svg = rsvg.new_from_file('compass.svg')
context.move_to(10,10)
context.scale(0.5, 0.5)
svg.render_cairo(context)

surface.finish()
```

Ergebnis



Text hinzufügen

```
context.select_font_face("Droid Sans Bold", cairo.FONT_S
context.set_font_size(48)
context.set_source_rgb(1, 1, 1)

text = 'Some text'

x_bearing, y_bearing, width, height = context.text_exten

context.move_to(100, 100)
context.show_text(text)
```

Ergebnis

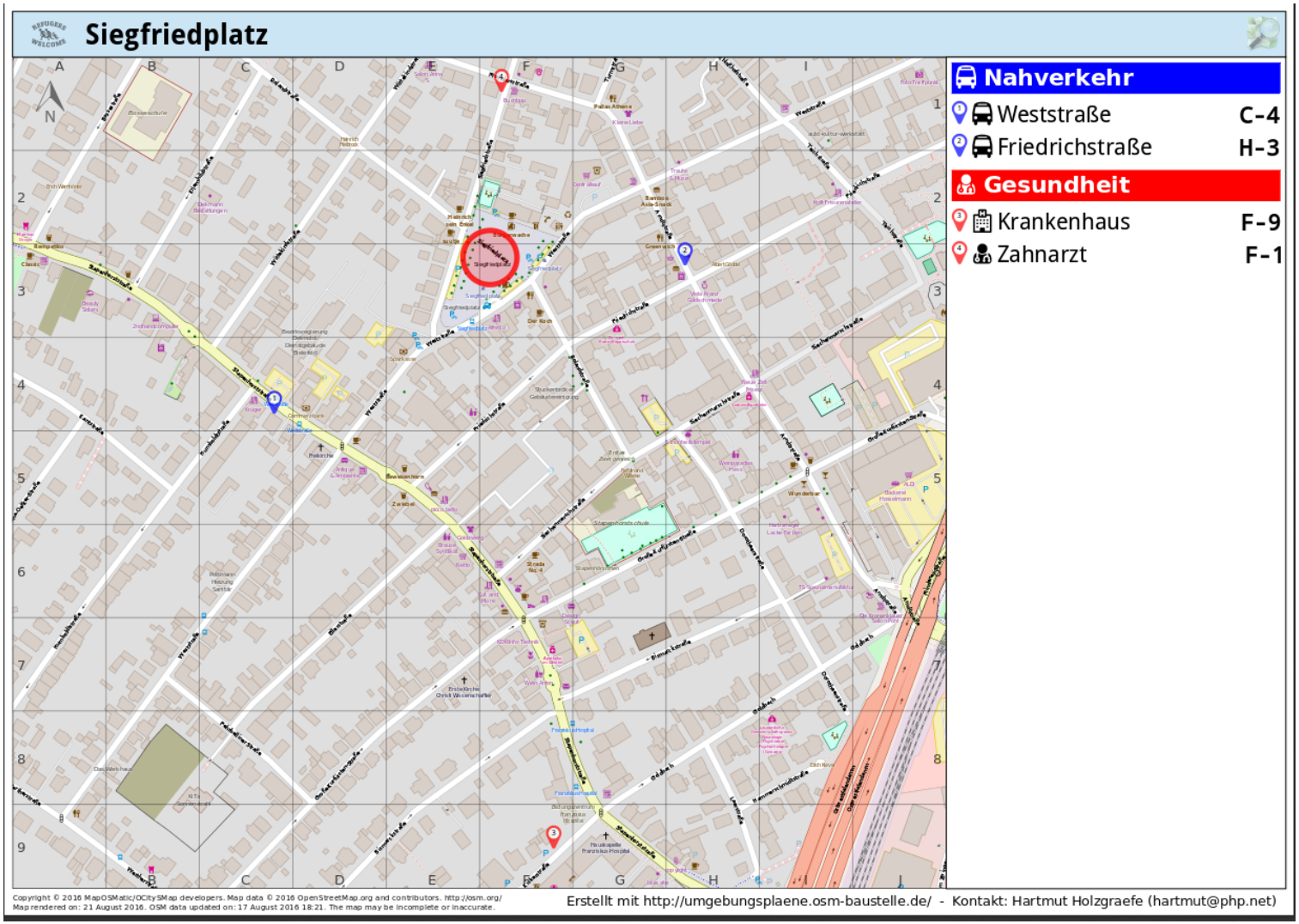
Zusammenfassung

Tatsächliche Anwendungen 1

`maposmatic.osm-baustelle.de`

Tatsächliche Anwendungen 2

`get-maps.org`



Was ich gelernt habe

Was ich gelernt habe

- Man muss nicht wirklich viel Python Code schreiben

Was ich gelernt habe

- Man muss nicht wirklich viel Python Code schreiben
- Die Arbeit steckt in den Stildefinitionen

Was ich gelernt habe

- Man muss nicht wirklich viel Python Code schreiben
- Die Arbeit steckt in den Stildefinitionen
- Flexible Lösung um gerenderte Kartendaten ...

Was ich gelernt habe

- Man muss nicht wirklich viel Python Code schreiben
- Die Arbeit steckt in den Stildefinitionen
- Flexible Lösung um gerenderte Kartendaten ...
- ... mit eigenen Dekorationen zu kombinieren

Was ich gelernt habe

- Man muss nicht wirklich viel Python Code schreiben
- Die Arbeit steckt in den Stildefinitionen
- Flexible Lösung um gerenderte Kartendaten ...
- ... mit eigenen Dekorationen zu kombinieren
- Die Mapik Dokumentation ist ausbaufähig :(

Fragen, Anmerkungen, Wünsche?

Quellen

Mapnik Wiki: `https://github.com/mapnik/mapnik/wiki`
Cairo Graphics: `https://cairographics.org/pycairo/`
RSVG: `https://developer.gnome.org/rsvg/stable/rsvg-Using-RSVG-with-cairo.html`
MapOSMatic: `https://maposmatic.osm-baustelle.de/`
Python/Mapnik Beispielsammlung (im Aufbau):
`https://github.com/hholzgra&python-mapnik-examples`