Rendering map data with Python and Mapnik

From Bits To Pictures

Hartmut Holzgraefe

hartmut@php.net

FOSDEM - Feb. 4th, 2018

Who am I?

- Hartmut Holzgraefe

Who am I?

- Hartmut Holzgraefe
- from Bielefeld, Germany

Who am I?

- Hartmut Holzgraefe
- from Bielefeld, Germany
- Studied electric engineering, computer science, and biology

Who am I?

- Hartmut Holzgraefe
- from Bielefeld, Germany
- Studied electric engineering, computer science, and biology
- OpenStreetMapper since 2007

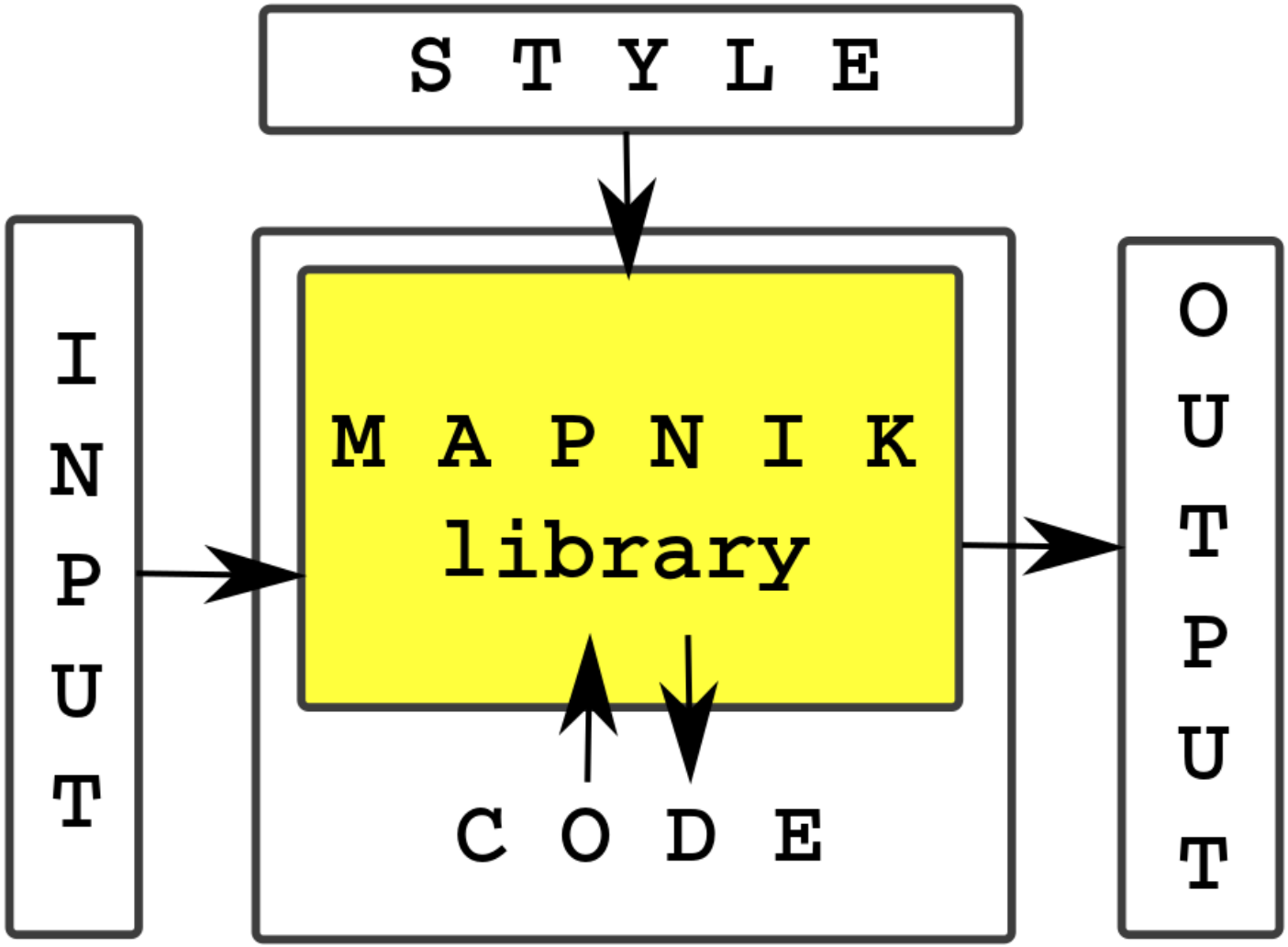
Who am I?

- Hartmut Holzgraefe
- from Bielefeld, Germany
- Studied electric engineering, computer science, and biology
- OpenStreetMapper since 2007
- Principal Database Support Engineer at MariaDB Corp.
  (and previously MySQL, Sun, Oracle, SkySQL)

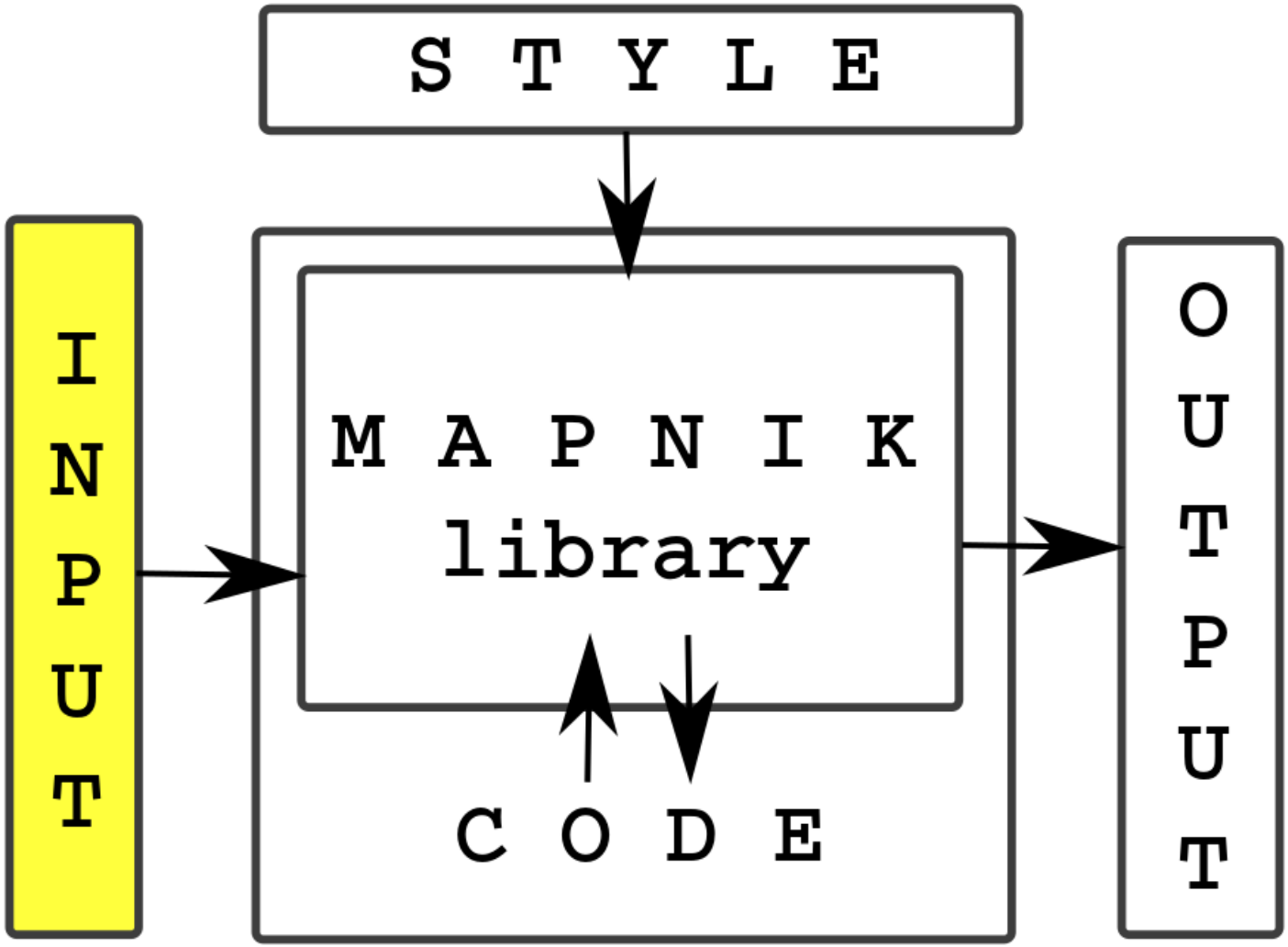
Mapnik Overview

1. Mapnik Overview
2. Prequisites
3. Points, Lines and Polygons
4. Layers, Styles and Symbolizers
5. Code basics
6. Using Symbolizers
7. Drawing on top
8. Summing it up ...

Mapnik Overview



Input



Data Sources

Mapnik can read map data from many different sources:

Data Sources

Mapnik can read map data from many different sources:

- Shapefiles

Multiple other formats via plugins:

Data Sources

Mapnik can read map data from many different sources:

- Shapefiles
- SQL database result sets

Multiple other formats via plugins:

Data Sources

Mapnik can read map data from many different sources:

- Shapefiles
- SQL database result sets
- GeoJson

Multiple other formats via plugins:

Data Sources

Mapnik can read map data from many different sources:

- Shapefiles
- SQL database result sets
- GeoJson

Multiple other formats via plugins:

Data Sources

Mapnik can read map data from many different sources:

- Shapefiles
- SQL database result sets
- GeoJson

Multiple other formats via plugins:

- ... OGR for various vector and raster formats, e.g. OSM XML and GPX

Data Sources

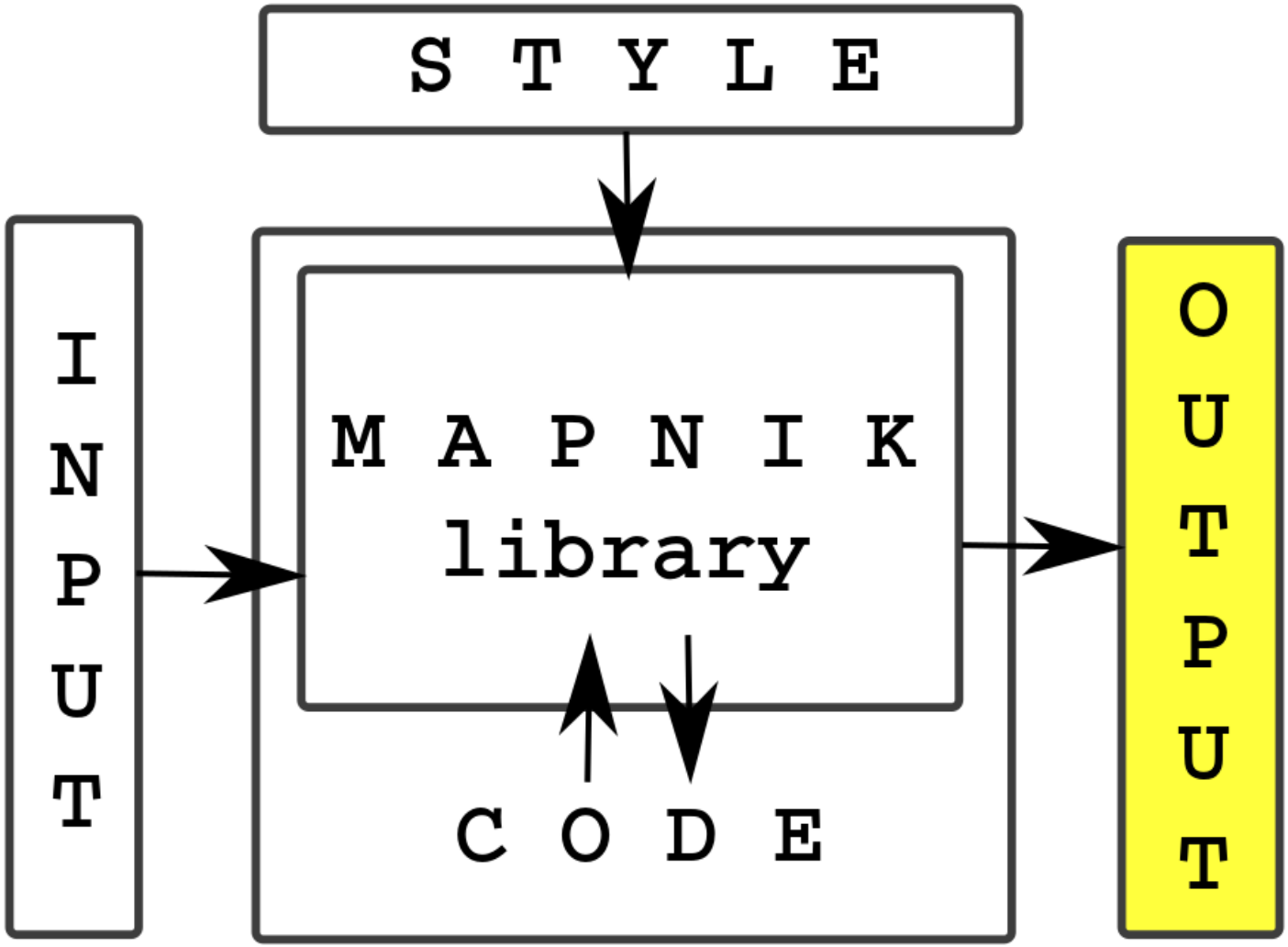
Mapnik can read map data from many different sources:

- Shapefiles
- SQL database result sets
- GeoJson

Multiple other formats via plugins:

- ... OGR for various vector and raster formats, e.g. OSM XML and GPX
- ... GDAL for various raster formats

Output



Output Formats

Mapnik can produce output in various formats

Output Formats

Mapnik can produce output in various formats

- PNG (32bit and 8bit)

Output Formats

Mapnik can produce output in various formats

- PNG (32bit and 8bit)
- JPG

Output Formats

Mapnik can produce output in various formats

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

Output Formats

Mapnik can produce output in various formats

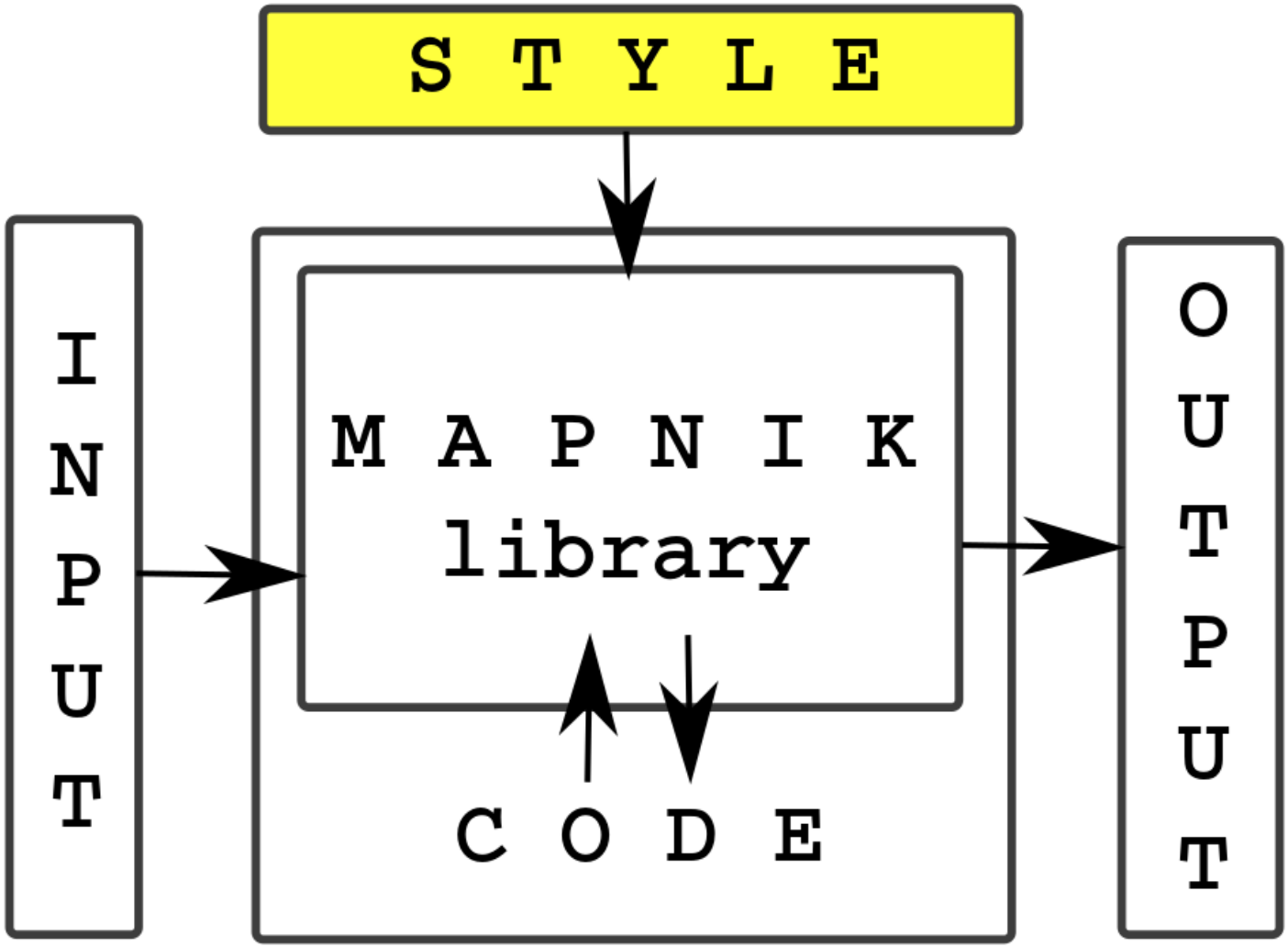
- PNG (32bit and 8bit)
- JPG
- SVG
- PDF

Output Formats

Mapnik can produce output in various formats

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

Style



Style

How data is rendered is defined by styles:

Style

How data is rendered is defined by styles:

- Styles can be defined in program code

Style

How data is rendered is defined by styles:

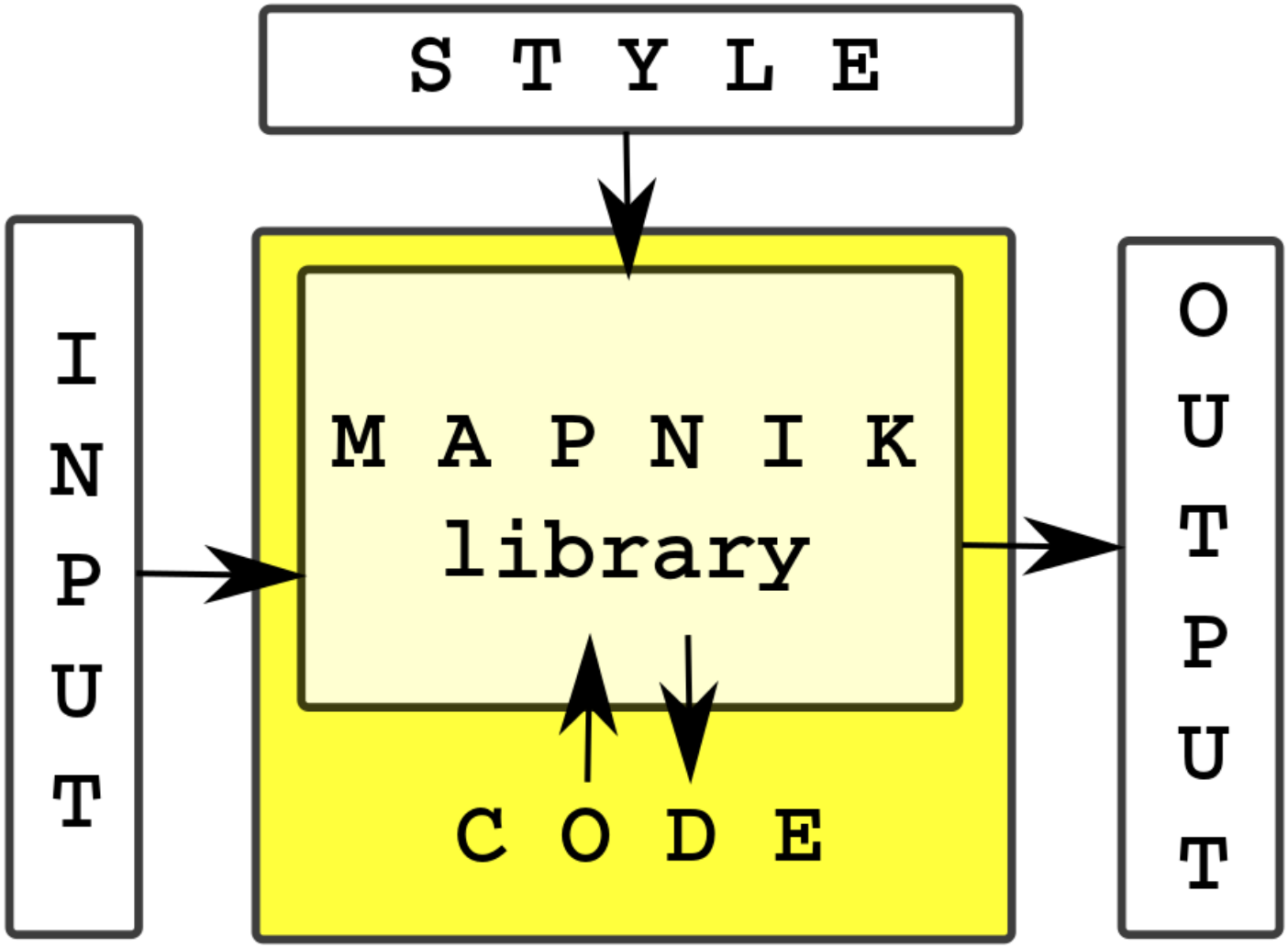
- Styles can be defined in program code
- or via XML style files

Style

How data is rendered is defined by styles:

- Styles can be defined in program code
- or via XML style files
- Some other style formats can be converted into Mapnik XML (mostly CartoCSS at this time)

Code



Control Code

Mapnik comes as a library written in C++, not a full application. So some extra code is needed to actually make it work.

Control Code

Mapnik comes as a library written in C++, not a full application. So some extra code is needed to actually make it work.

- native C++

Control Code

Mapnik comes as a library written in C++, not a full application. So some extra code is needed to actually make it work.

- native C++
- Python bindings

Control Code

Mapnik comes as a library written in C++, not a full application. So some extra code is needed to actually make it work.

- native C++
- Python bindings
- Experimental bindings for PHP 7

Prequisites

1 Mapnik Overview

2 Prequisites

3 Points, Lines and Polygons

4 Layers, Styles and Symbolizers

5 Code basics

6 Using Symbolizers

7 Drawing on top

8 Summing it up ...

We need the following components:

- Python (2 or 3)
- Mapnik 3 (2?)
- Python bindings for 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
```

Points, Lines and Polygons

1 Mapnik Overview

2 Prequisites

3 Points, Lines and Polygons

4 Layers, Styles and Symbolizers

5 Code basics

6 Using Symbolizers

7 Drawing on top

8 Summing it up ...

Points, Lines and Polygons:

All Mapnik data sources provide geo data as a collection of

- Points

Depending on the underlying data source some conversions may happen on the way.

All geo objects may have additional attributes that you can filter by, or use to decide how to display them (e.g. “name” text)

Points, Lines and Polygons:

All Mapnik data sources provide geo data as a collection of

- Points
- Lines

Depending on the underlying data source some conversions may happen on the way.

All geo objects may have additional attributes that you can filter by, or use to decide how to display them (e.g. “name” text)

Points, Lines and Polygons:

All Mapnik data sources provide geo data as a collection of

- Points
- Lines
- Polygons

Depending on the underlying data source some conversions may happen on the way.

All geo objects may have additional attributes that you can filter by, or use to decide how to display them (e.g. “name” text)

Points, Lines and Polygons:

All Mapnik data sources provide geo data as a collection of

- Points
- Lines
- Polygons
- Raster Images

Depending on the underlying data source some conversions may happen on the way.

All geo objects may have additional attributes that you can filter by, or use to decide how to display them (e.g. “name” text)

Layers, Styles and Symbolizers

1 Mapnik Overview

2 Prequisites

3 Points, Lines and Polygons

4 Layers, Styles and Symbolizers

5 Code basics

6 Using Symbolizers

7 Drawing on top

8 Summing it up ...

Layers

A Mapnik Layer is importing some data using one of the available data sources and binds it to one or more styles to present the imported data.

Styles

A Style can filter imported data and defines which symbolizer(s) to use to present the data.

Symbolizers

Symbolizers perform the actual rendering of data. There are four basic types:

- PointSymbolizer

Symbolizers

Symbolizers perform the actual rendering of data. There are four basic types:

- PointSymbolizer
- LineSymbolizer

Symbolizers

Symbolizers perform the actual rendering of data. There are four basic types:

- PointSymbolizer
- LineSymbolizer
- PolygonSymbolizer

Symbolizers

Symbolizers perform the actual rendering of data. There are four basic types:

- PointSymbolizer
- LineSymbolizer
- PolygonSymbolizer
- RasterSymbolizer

Symbolizers (cont.)

- MarkerSymbolizer

Symbolizers (cont.)

- MarkerSymbolizer
- LinePatterSymbolizer

Symbolizers (cont.)

- MarkerSymbolizer
- LinePatterSymbolizer
- TextSymbolizer

Symbolizers (cont.)

- MarkerSymbolizer
- LinePatterSymbolizer
- TextSymbolizer
- ShieldSymbolizer

Symbolizers (cont.)

- MarkerSymbolizer
- LinePatterSymbolizer
- TextSymbolizer
- ShieldSymbolizer
- PolygonPatternSymbolizer

Symbolizers (cont.)

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

Code basics

1 Mapnik Overview

2 Prequisites

3 Points, Lines and Polygons

4 Layers, Styles and Symbolizers

5 Code basics

6 Using Symbolizers

7 Drawing on top

8 Summing it up ...

A no-op example

```
import mapnik
```

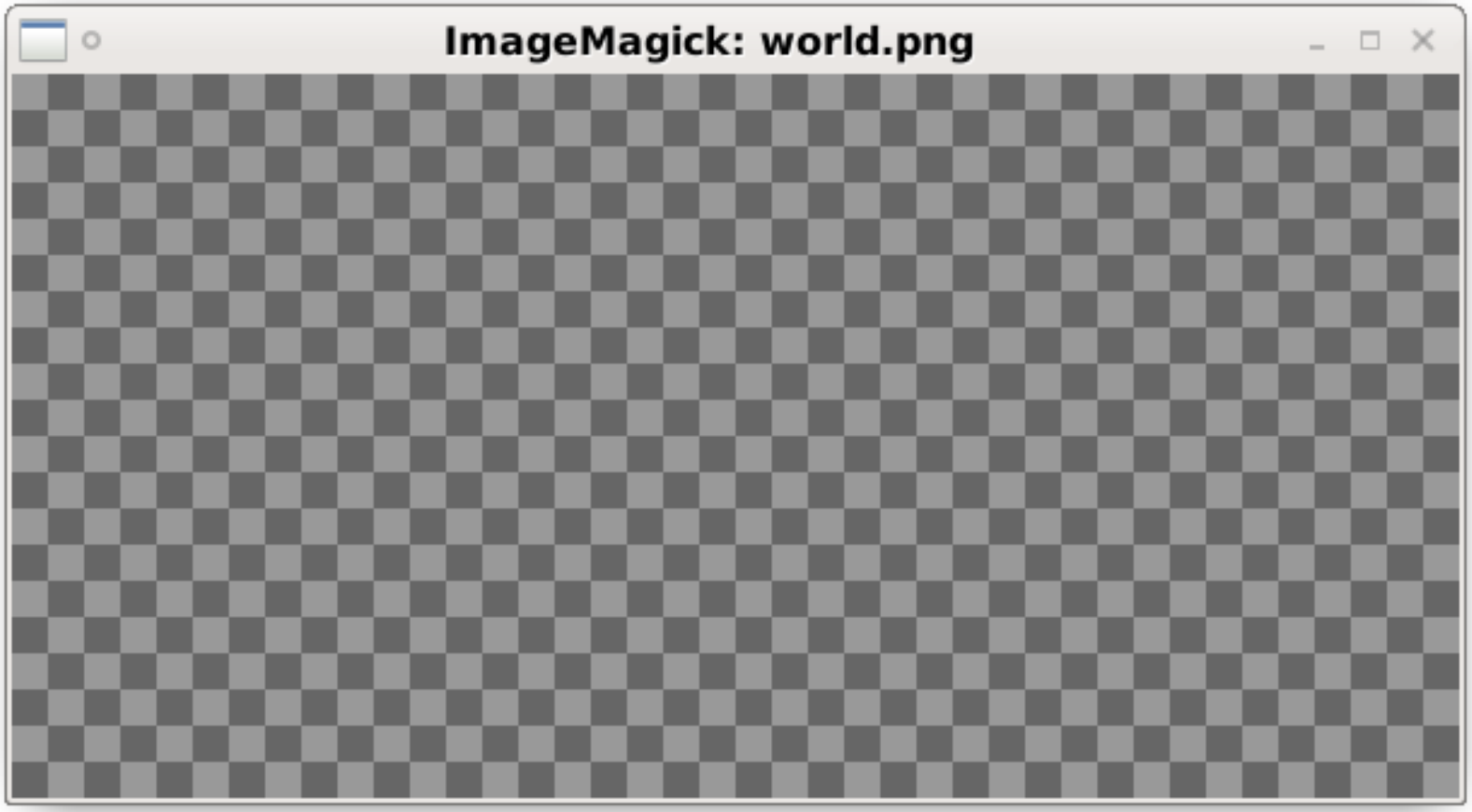
A no-op example

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

A no-op example

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

... and its result



A minimal example

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

A minimal example

```
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')
```

A minimal example

```
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')
8
9  rules = mapnik.Rule()
10 rules.symbols.append(polygons)
```

A minimal example

```
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)
```

A minimal example

```
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')
```

A minimal example

```
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)
```

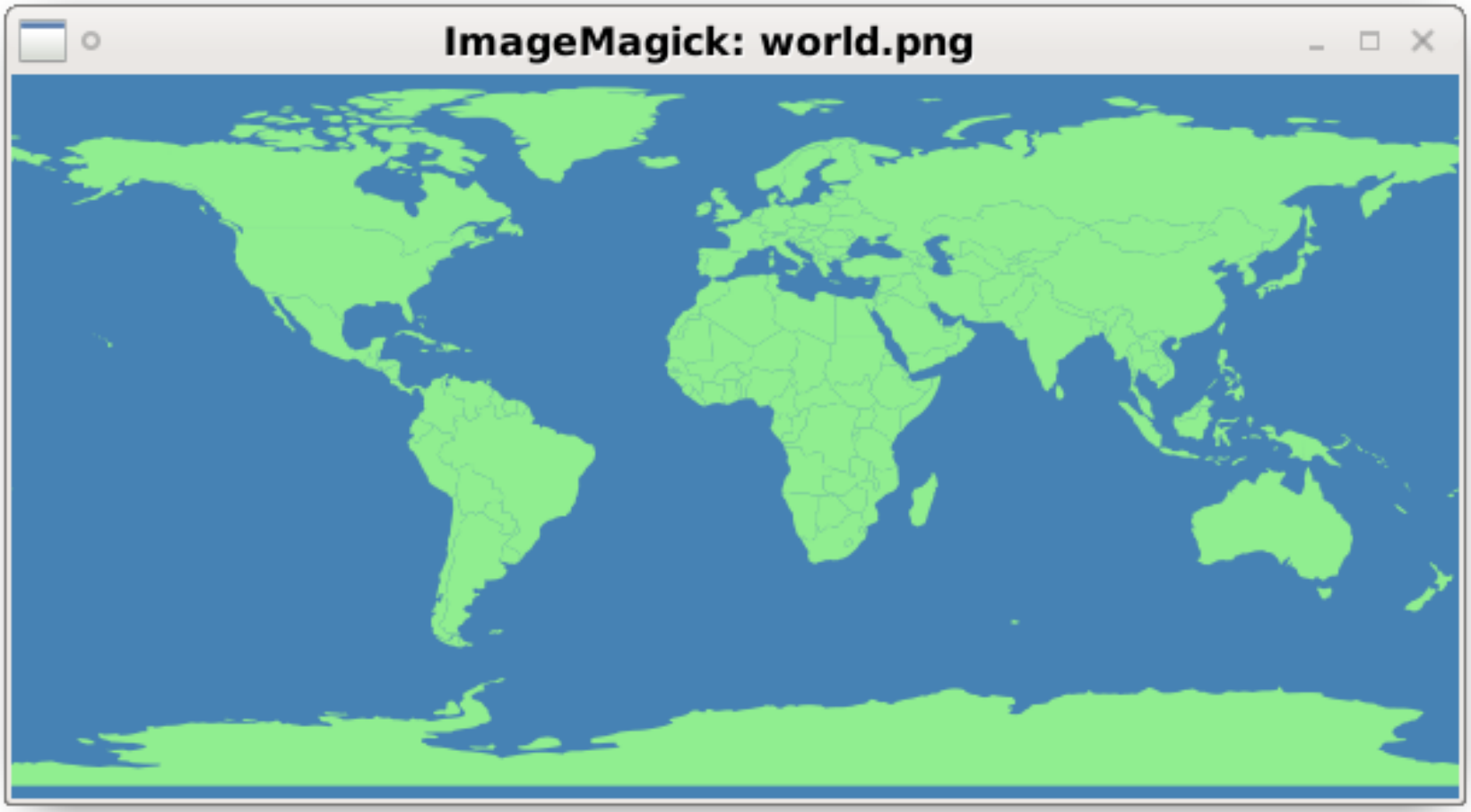
A minimal example

```
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()
```

A minimal example

```
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')
```

... and its result



Finding Germany

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

```

Finding Germany

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

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

```

Finding Germany

```
[...]
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')
```

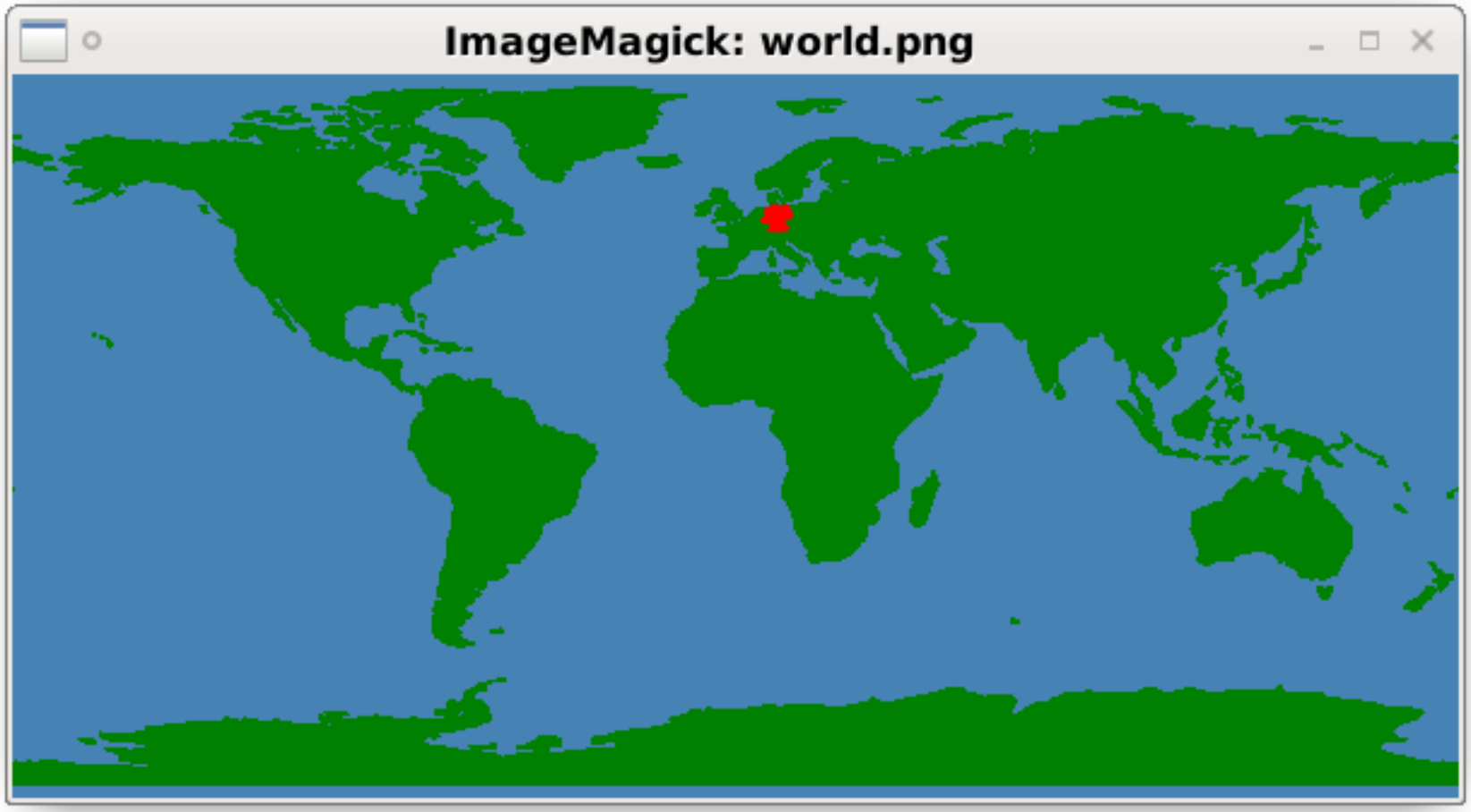
Finding Germany

```
[...]
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)
```

Finding Germany

```
[...]
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)
[...]
```

... and its result



Using XML

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

Using XML

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

Using 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

```
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="Borders">
    <Rule>
      <PolygonSymbolizer fill="green" gamma="0.0"/>
    </Rule>
```

XML style definition

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

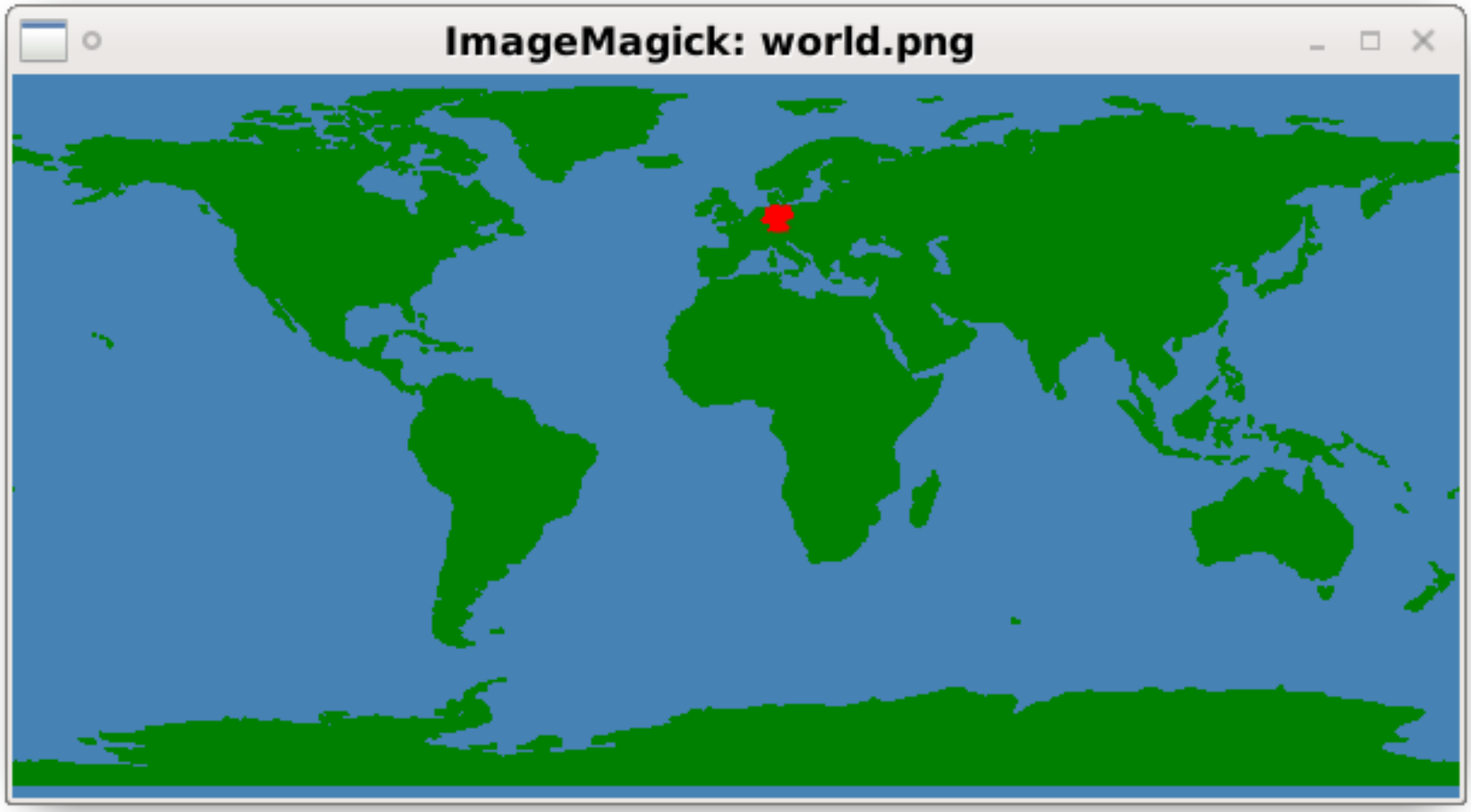
XML style definition

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='steelblue'>
  <Style name="Borders">
    <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">ne_110m_admin_0_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="Borders">
    <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">ne_110m_admin_0_countries.shp</Parameter>
      <Parameter name="type">shape</Parameter>
    </Datasource>
  </Layer>
</Map>
```

... and its result



Using Symbolizers

1 Mapnik Overview

2 Prequisites

3 Points, Lines and Polygons

4 Layers, Styles and Symbolizers

5 Code basics

6 Using Symbolizers

7 Drawing on top

8 Summing it up ...

New Skeleton

```
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')
```

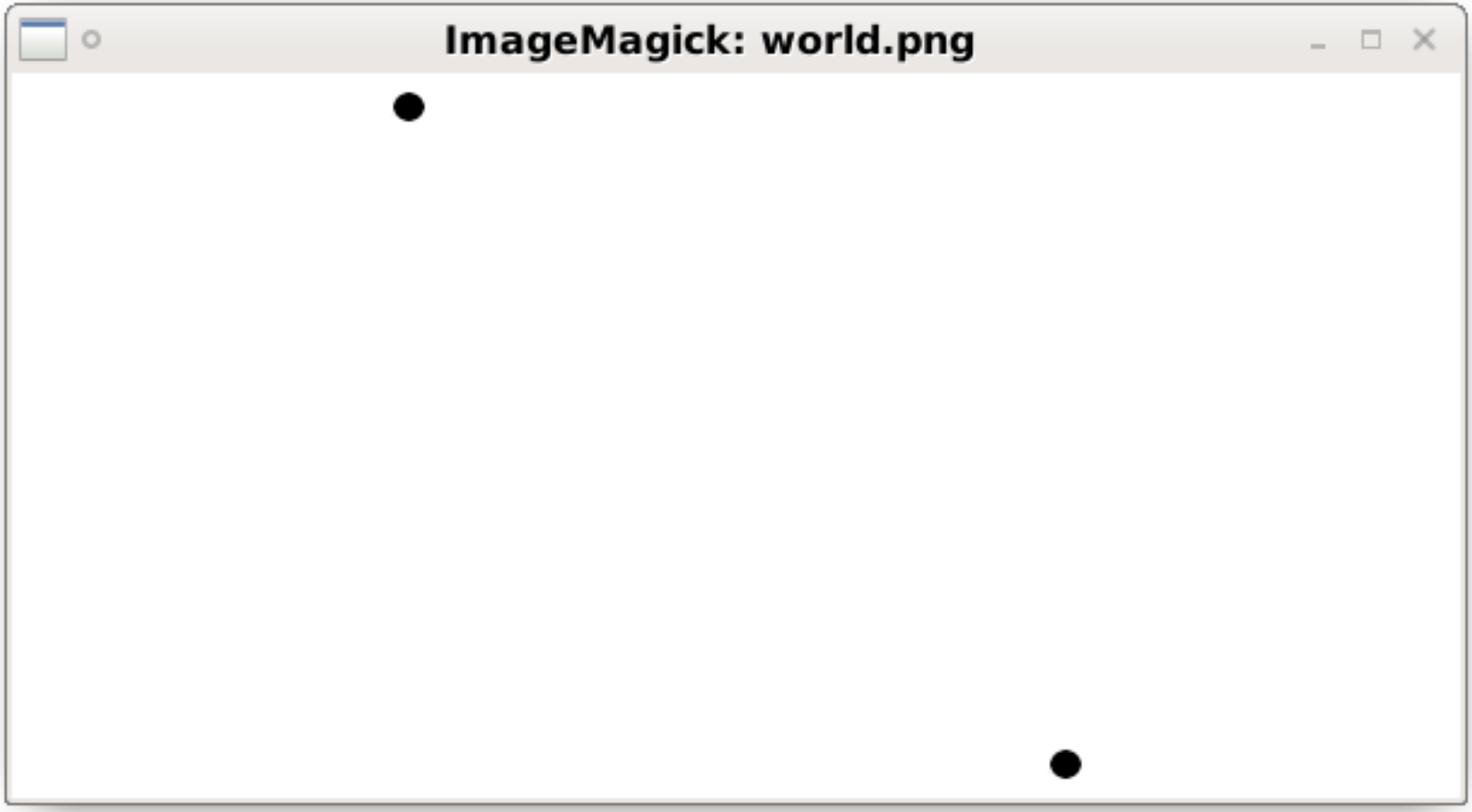
Point Symbolizer

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

  <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 ]
      }
    }
  ]
}
```

Result



Line Symbolizer

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

Line Symbolizer

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color='white'>
  <Style name='line'>
    <Rule>
      <LineSymbolizer stroke='steelblue' stroke-width='3
      <TextSymbolizer placement="line" face-name="DejaVu
                      fill="black" halo-fill="white" hal
      >[name]</TextSymbolizer>
    </Rule>
  </Style>

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

Result

The other symbolizers

Symbolizers not covered by examples yet

- PolygonSymbolizer (seen earlier)

The other symbolizers

Symbolizers not covered by examples yet

- PolygonSymbolizer (seen earlier)
- MarkerSymbolizer - repeated symbol along line

The other symbolizers

Symbolizers not covered by examples yet

- PolygonSymbolizer (seen earlier)
- MarkerSymbolizer - repeated symbol along line
- ShieldSymbolizer - flexible symbol with text along line

The other symbolizers

Symbolizers not covered by examples yet

- PolygonSymbolizer (seen earlier)
- MarkerSymbolizer - repeated symbol along line
- ShieldSymbolizer - flexible symbol with text along line
- LinePatternSymbolizer - line with attached symbols

The other symbolizers

Symbolizers not covered by examples yet

- PolygonSymbolizer (seen earlier)
- MarkerSymbolizer - repeated symbol along line
- ShieldSymbolizer - flexible symbol with text along line
- LinePatternSymbolizer - line with attached symbols
- PolygonPatternSymbolizer - fill polygon with repeated image

The other symbolizers

Symbolizers not covered by examples yet

- PolygonSymbolizer (seen earlier)
- MarkerSymbolizer - repeated symbol along line
- ShieldSymbolizer - flexible symbol with text along line
- LinePatternSymbolizer - line with attached symbols
- PolygonPatternSymbolizer - fill polygon with repeated image
- BuildingSymbolizer - draw pseudo-3D buildings

Drawing on top

1. Mapnik Overview
2. Prequisites
3. Points, Lines and Polygons
4. Layers, Styles and Symbolizers
5. Code basics
6. Using Symbolizers
7. Drawing on top
8. Summing it up ...

Drawing into a Cairo context

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

Drawing into a Cairo context

```
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)
```

Drawing into a Cairo context

```
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)
```

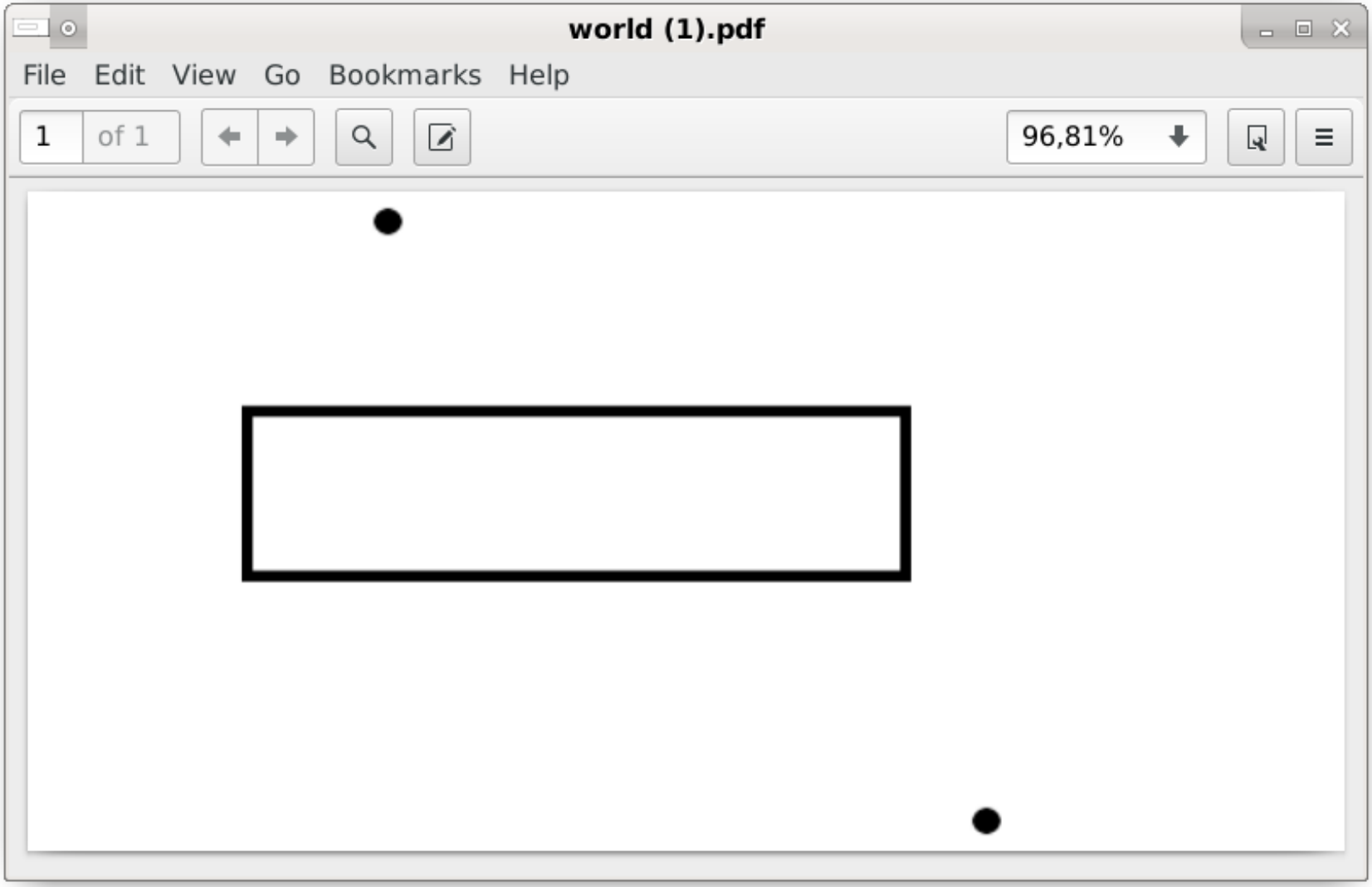
Drawing into a Cairo context

```
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()
```

Drawing into a Cairo context

```
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()
```

Result



Adding SVG images v2

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

Adding SVG images 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
```

Adding SVG images 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)
```

Adding SVG images 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)
```

Adding SVG images 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()
```

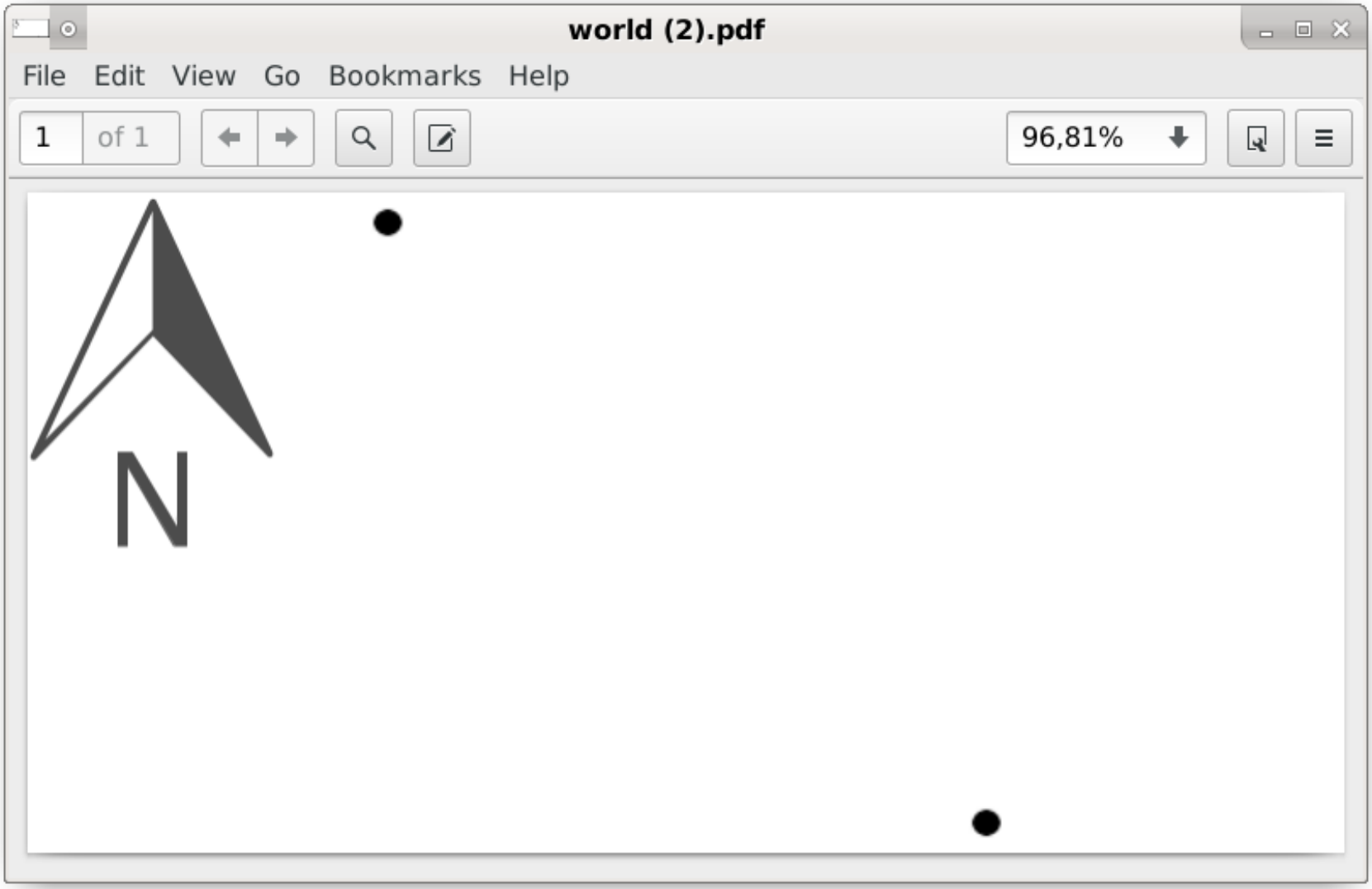
Adding SVG images v3

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

Adding SVG images 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()
```

Result



Adding Text

```
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)
```

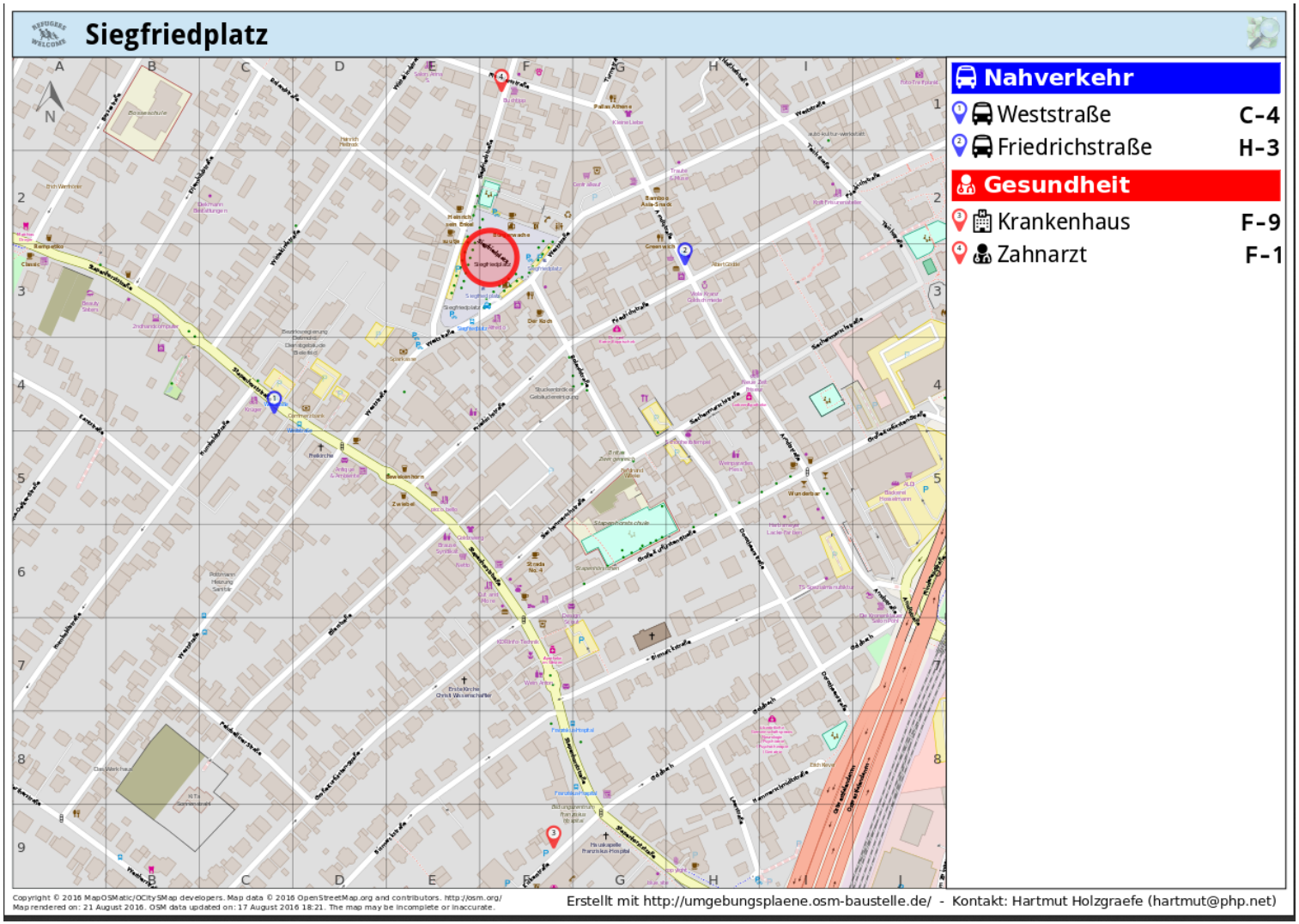
Result

Summing it up ...?

1. Mapnik Overview
2. Prequisites
3. Points, Lines and Polygons
4. Layers, Styles and Symbolizers
5. Code basics
6. Using Symbolizers
7. Drawing on top
8. Summing it up ...

A full featured Example

`get-maps.org`

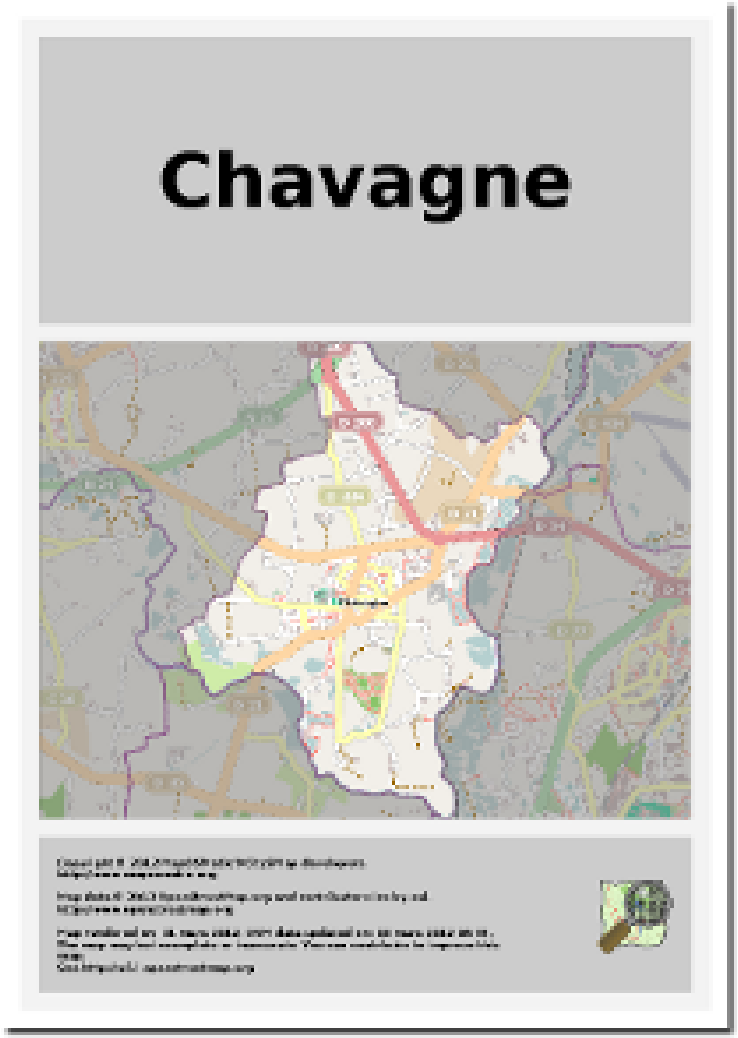


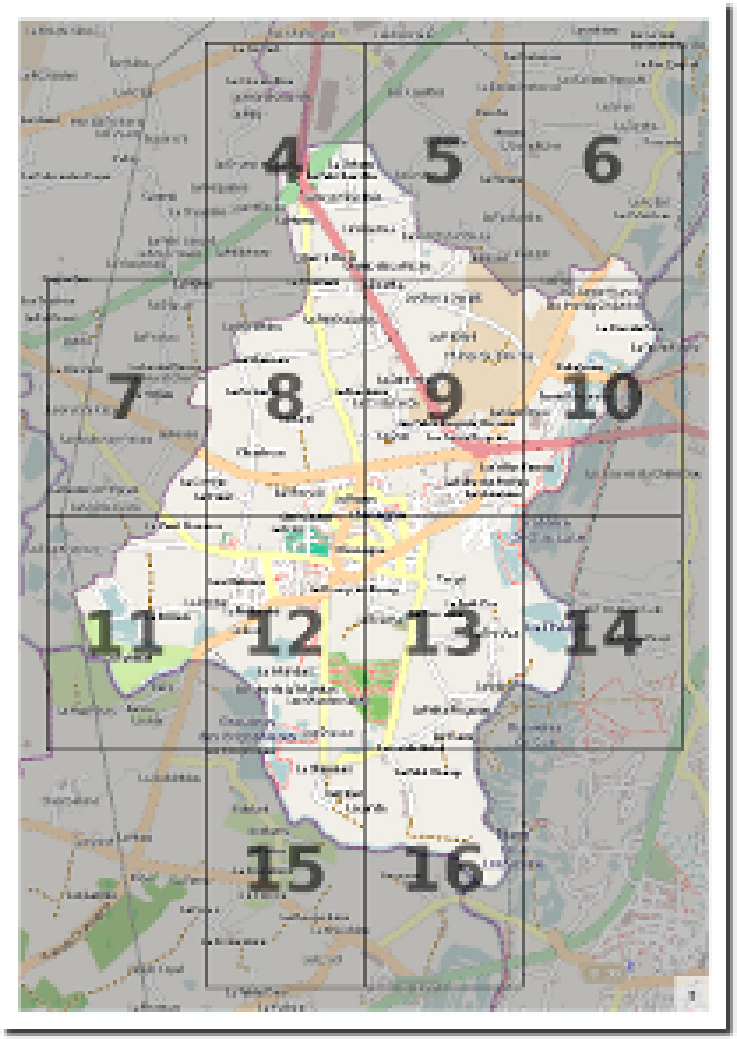
Another featured Example

`maposmatic.osm-baustelle.de`

And another one

`maposmatic.osm-baustelle.de`





What we learned

What we learned

- Code wise it is actually rather easy

What we learned

- Code wise it is actually rather easy
- The devil is in the styles (and details)

What we learned

- Code wise it is actually rather easy
- The devil is in the styles (and details)
- Flexible solution to mix map rendering ...

What we learned

- Code wise it is actually rather easy
- The devil is in the styles (and details)
- Flexible solution to mix map rendering ...
- ... and custom image decorations

What we learned

- Code wise it is actually rather easy
- The devil is in the styles (and details)
- Flexible solution to mix map rendering ...
- ... and custom image decorations
- Mapnik documentation is suboptimal :(

Questions, Remarks, Wishes?

Further reading