

Mapgen

by Gary68

User's manual

Version 1.00, April 2010

- residential
- motorway
- primary
- secondary
- tertiary
- track
- footway
- path
- cycleway
- park
- stadium
- cemetery
- military
- forest
- farmland
- residential
- industrial

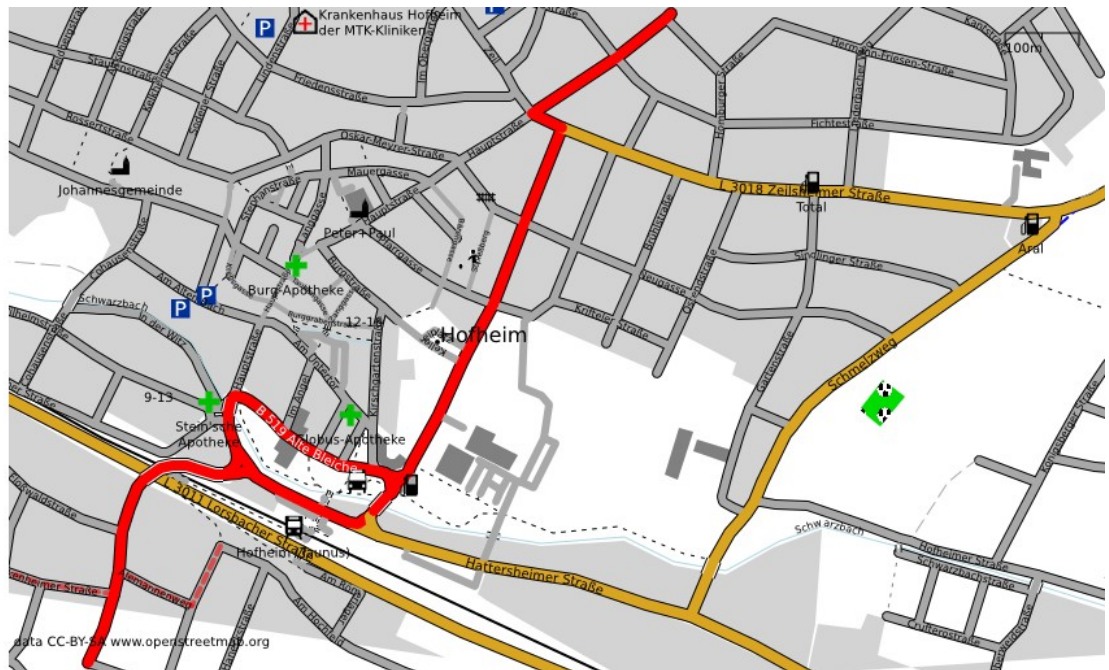


Table of Contents

Introduction.....	3
Hints	3
Installation.....	3
Basic parameters.....	4
Output.....	4
Usage examples.....	4
Map size and the like.....	5
Advanced parameters.....	5
Setting a scale for rules alone.....	7
Style file format.....	8
File.....	8
Nodes.....	8
Ways.....	9
Routes.....	11
Route icons.....	13
Colors.....	13
Fonts.....	14
Oneways.....	15
Extracts.....	15
Declutter.....	16
Grids, directory and stats.....	17
Street directory.....	17
Poi and place list.....	18
Statistics.....	18
Coordinates Grid	20
Debug.....	21

Introduction

Mapgen has its roots in osmdiff.pl and osmrender.pl. They were very basic render programs. Once Haiti was hit by the strong earthquake I wanted to provide large png maps for the local help. This proved to be hard since so many things couldn't be done with my programs. So I decided to improve the features of my renderer and give it a new name. So the basic goals became:

- Fast and easy map generation, different output formats
- Fast extraction of needed data out of *.osm files (place=*)
- Easy style file handling
- Street and place directories
- Keep it simple (easy invocation with only 2 mandatory parameters)
- Keep it powerful (by using more parameters)

Hints

The projection method used is not appropriate for very big areas.

Be aware that at the edges of the map items may be missing. This may occur when items are not completely contained in the osm file. You can try to prevent this by using the clip parameter with larger values.

Installation

- Put the mapgen.pl file in a folder
- Put *.pm files in a subfolder called OSM (can also be put into a directory contained in the @INC pathes)
- Get Math::Poygon from CPAN and create a subfolder Math (can also be put into a directory contained in the @INC pathes)
- (Install osmosis if desired; take care that it can be invoked from command line)
- (Install inkscape if desired; take care that it can be invoked from command line) **This is necessary when usein SVG files as tile patterns or when converting from SVG to PNG or PDF!**
- bzip2 installation: `sudo apt-get install libcompress-bzip2-perl`
- GD installation: `sudo apt-get install libgd-graph3d-perl`

Information about Osmosis can be found here: <http://wiki.openstreetmap.org/wiki/Osmosis>

Basic parameters

Obviously there are some things that can't be hidden from the user. So the user has to specify at least 2 basic parameters:

```
-in=file.osm  
-style=style.csv (original can be kept and maintained in OO sheet or MS Excel)
```

-in also supports *.osm.bz2 format.

Default output name is mapgen.svg.

Output

Basic and only output format from mapgen itself is SVG. That has the advantage that all further formats contain all elements that mapgen can produce. Disadvantage is a post-processing done by inkscape.

```
-out=file.svg (png and pdf names are automatic, DEFAULT=mapgen.svg)
```

There are two options specifying additional output formats:

```
-png (also produce png, inkscape must be installed, very big)  
-pdf (also produce pdf, inkscape must be installed)
```

The names are automatic and derived from the -out name. Inkscape must be installed and your system must be able to run it from command line in the current directory.

The PNG files are rather big. So maybe some post-processing is appropriate.

Usage examples

Most **simple** form:

```
perl mapgen.pl -in=file.osm -style=mapgenRules.csv
```

Also specify an **output** name, if you wish

```
perl mapgen.pl -in=file.osm -style=mapgenRules.csv -out=map.svg
```

Also specify **size** and that you want a **PDF** additionally:

```
perl mapgen.pl -in=file.osm -style=mapgenRules.csv -size=2048 -pdf
```

Now let's say you have a big osm file but only want a **map of a certain city**:

```
perl mapgen.pl -in=germany.osm -style=mapgenRules.csv -place=Frankfurt
```

This will probably not cover the whole of Frankfurt because the default **radiuses** are too small (2km each direction)

```
perl mapgen.pl -in=germany.osm -style=mapgenRules.csv -place=Frankfurt -lonrad=10  
-latrad=10
```

Now we want our map with **grid lines** and a **street directory**:

```
perl mapgen.pl -in=file.osm -style=mapgenRules.csv -grid=8 -dir
```

And let's turn off the **legend**:

```
perl mapgen.pl -in=file.osm -style=mapgenRules.csv -legend=0
```

Map size and the like

The background color can be set according to the given color set.

The size of the picture is specified by the width in pixels. Height is automatically calculated.

Clipping means that not all of the data given in the osm file will be presented. This is useful to clip incomplete data at the edges of the area in the osm file.

Padding means to pad the map with an empty border. Useful for islands...

Program will output some useful information about map and paper sizes as well as resolution and scale.

```
-bgcolor=TEXT (color for background)
-size=<integer> (in pixels for x axis, DEFAULT=1024)
-clip=<integer> (percent data to be clipped on each side, 0=no clipping, DEFAULT=0)
-pad=<INTEGER> (percent of white space around data in osm file, DEFAULT=0)
```

Advanced parameters

By default a legend is drawn in the upper left corner. This can be switched off.

```
-legend=INT (0=no legend; 1=legend in top left corner; 2 = legend in lower right
corner; DEFAULT=1)
```

You can also place the legend in the lower right corner by giving -legend=2

The legend will only show elements that would be drawn if present in the current map scale!

A ruler is drawn by default in the upper right corner. This can be switched off as well. Additionally a color can be specified.



```
-ruler=INT (0=no ruler; 1=draw ruler; DEFAULT=1)
-rulercolor=TEXT (DEFAULT=black)
```

Optionally a scale value can be calculated and added to the map. Of course the color for this text can be set.

```
-scale (print scale)
-scalecolor=TEXT (set scale color; DEFAULT = black)
```



A specific scale can be set, i.e. 1:25.000 by adding `-scaleset=25000` to the command line. To be able to work with this information you have to specify the resolution of the output device in dpi. By default this is set to 300dpi.

```
-scaleset=INTEGER (1:x preset for map scale; overrides -size=INTEGER! set correct
printer options!)
-scaledpi=INTEGER (print resolution; DEFAULT = 300 dpi)
```

Setting the scale overrides the -size parameter!

The program will in any case print information on how big the map will be and on what paper size it will fit.

Setting a scale for rules alone

If you specify a certain scale for rulescaleset then for selecting rules not the actual scale of the map is used but the specified one.

This is useful to show more or less details than initially intended by the given ruleset (minScale, maxScale).

A similar function can be found in the Garmin GPS receivers: Map details normal/more/most/less/least.

```
-rulescaleset=INTEGER (determines the scale used to select rules; DEFAULT=0,  
meaning actual map scale is used to select rules)
```

Style file format

In any case only one rule per object is used. And this is the first matching rule according to the style file. So it's possible i.e. first to look after all tracktype=XY and then having a “default rule” for all highway=track.

So it is important to have more specific rule higher above in the sheet/file!

The different sections of the file start with a SECTION line: This line contains the word “SECTION” in the first cell. These lines are mandatory even if no rules for a section are present.

You can add COMMENT lines for remarks if you enter “COMMENT” in the first cell. These lines will be omitted when style file is read.

File

Note that you can enter multiple key/value pairs for a rule. Valid for nodes and ways. Different entries must be separated by a “|”. If you for example want a rule for all tracks with tracktype=grade1, then the entries would be:

```
column key/tag: highway|tracktype
column value:   track|grade1
```

Nodes

Column #	Name	Values	Description
1	key	see wiki values can be separated by a “ ”. see above	
2	value	see wiki an asterisk (*) may be used as a wild card values can be separated by a “ ”. see above	
3	color	see separate table	the fill color
4	thickness	INTEGER	
5	label	key, where value will be the label text; entries can be separated by ! or #. !=AND. #=PRIO special keys: _lon and _lat	
6	label color	see list below	
7	label size	INTEGER	size of text
8	label offset	INTEGER	NOT USED RIGHT NOW
9	legend	0 or 1	

10	Icon	File name	
11	Icon size	In pixels	
12	FromScale	INTEGER	
13	ToScale	INTEGER	

A halo can be specified by using the option:

`-halo=<FLOAT>` (white halo width for point feature labels; DEFAULT=0)

However then a bold font is used so you also have to adjust `-ppc` to maybe 7.

The halo is white and opacity is 90%. Use large font sizes.

Ways

Column #	Name	Values	Description
1	key	see wiki values can be separated by a “ ”	
2	value	see wiki an asterisk (*) may be used as a wild card values can be separated by a “ ”	
3	color	see separate list	the fill color
4	thickness	INTEGER	thickness of line
5	dash style	1-4: for tracks 10-14: relation 1:1 dashes 20-23: for borders 30-35: for steps (drawn with linecap=BUTT)	determines the style of the dashes forming the way
6	Border color	TEXT	Border color for the way. Only valid if border thickness is > 0
7	Border thickness	INTEGER	Border thickness for the way
8	fill	0 or 1	0 = area will not be filled; 1 = area will be filled
9	label	key, where value will be used as label text. entries can be separated by ! or #. !=AND. #=PRIO special keys: _lon and _lat	
10	label color	see below	*

11	label size	INTEGER	font size
12	label font-family	see below	
13	label offset	INTEGER	offset for label text in y-direction (negative = up, positive = down)
14	legend	0 or 1	entry for automatic legend (0=no, 1=yes)
15	base layer	0 or 1	applies for areas (closed ways). areas tagged with 1 are drawn as "background" first. use for landuse, natural etc.
16	Icon	TEXT	Fill icon for area. SVG and PNG supported
17	FromScale	INTEGER	
16	ToScale	INTEGER	

If you want to use SVG files as area tile patterns inkscape is mandatory!

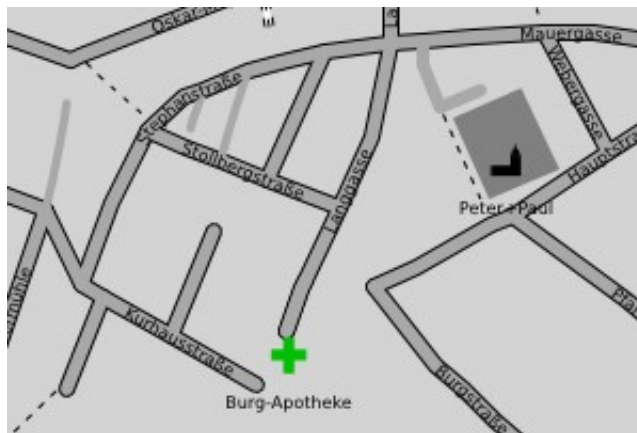
Using _lon and _lat you could nicely label a natural=peak, using a second rule for elevation i.e:



Using area icons...



Ways with borders...



Routes

Column #	Name	Values	Description
1	RouteType	bus, hiking - see wiki	
2	Color(s)	All valid colors	A set of colors can be given by using the ; as a delimiter
3	Thickness	INTEGER	...of the line
4	DashType	See above	...of the line
5	Opacity	0..100	...opacity in percent
6	Label	Valid keys	Key to get the label from
7	StopNodeThicknness	INTEGER	
8	FromScale	INTEGER	
9	ToScale	INTEGER	

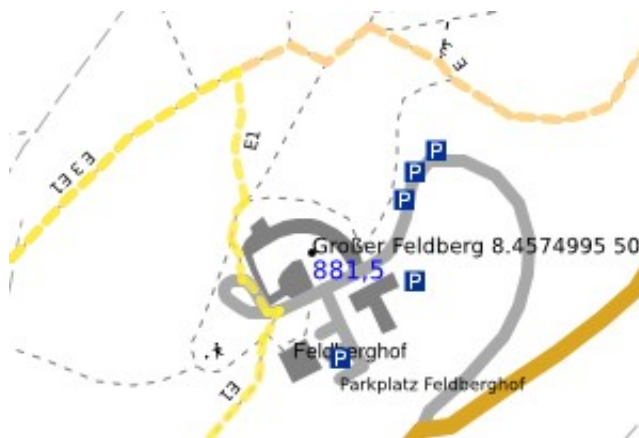
The colors will be assigned to each found route in order of occurrence in style file if no k/v for the colors are present in the osm file.

Additionally the labels can be configured by command line parameters:

```
-routelabelcolor=TEXT (color for labels of routes)
-routelabelsize=INTEGER (DEFAULT=8)
-routelabelfont=TEXT (DEFAULT=sans-serif)
-routelabeloffset=INTEGER (DEFAULT=10)
```



Bus routes in Frankfurt



Hiking routes in the Taunus



Some bus stops in Frankfurt

Route icons

Additionally icons for routes can be added to the map if icons are provided as follows. Create folder “routeicons” and save icons there. As the file name choose the ref or the name of the route preceeded by the route type and a dash. **If the route has a ref the ref has to be used!**

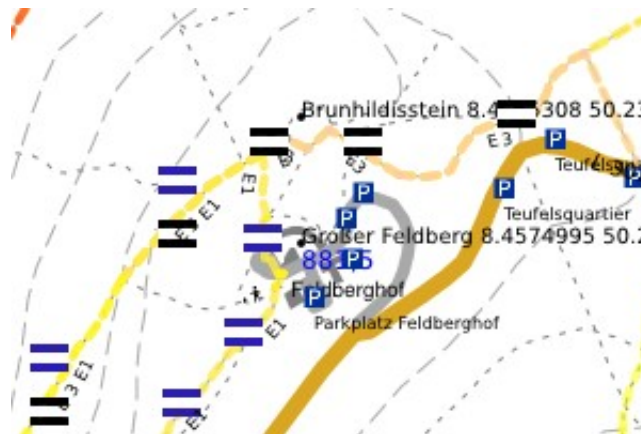
Example: ./routeicons/hiking-E3.png OR ./routeicons/hiking-E3.svg

The default icon directory can be changed.

```
-icondir=TEXT (dir for icons for routes; ./icondir/ i.e.; DEFAULT=./routeicons/ )
```

You can also change the default distance of 25 pixels between multiple icons for same way.

```
-routeicondist=INTEGER (dist in y direction for route icons on same route;  
DEFAULT=25)
```



Colors

aliceblue	blueviolet	cyan	darkorange
antiquewhite	brown	darkblue	darkorchid
aqua	burlywood	darkcyan	darkred
aquamarine	cadetblue	darkgoldenrod	darksalmon
azure	chartreuse	darkgray	darkseagreen
beige	chocolate	darkgreen	darkslateblue
bisque	coral	darkgrey	darkslategray
black	cornflowerblue	darkkhaki	darkslategrey
blanchedalmond	cornsilk	darkmagenta	darkturquoise
blue	crimson	darkolivegreen	darkviolet

deeppink	lightblue	mediumturquoise	royalblue
deepskyblue	lightcoral	mediumvioletred	saddlebrown
dimgray	lightcyan	midnightblue	salmon
dimgrey	lightgoldenrodyellow	mintcream	sandybrown
dodgerblue	lightgray	mistyrose	seagreen
firebrick	lightgreen	moccasin	seashell
floralwhite	lightgrey	navajowhite	sienna
forestgreen	lightpink	navy	silver
fuchsia	lightsalmon	oldlace	skyblue
gainsboro	lightseagreen	olive	slateblue
ghostwhite	lightskyblue	olivedrab	slategray
gold	lightslategray	orange	slategrey
goldenrod	lightslategrey	orangered	snow
gray	lightsteelblue	orchid	springgreen
green	lightyellow	palegoldenrod	steelblue
greenyellow	lime	palegreen	tan
grey	limegreen	paleturquoise	teal
honeydew	linen	palevioletred	thistle
hotpink	magenta	papayawhip	tomato
indianred	maroon	peachpuff	turquoise
indigo	mediumaquamarine	peru	violet
ivory	mediumblue	pink	wheat
khaki	mediumorchid	plum	white
lavender	mediumpurple	powderblue	whitesmoke
lavenderblush	mediumseagreen	purple	yellow
lawngreen	mediumslateblue	red	yellowgreen
lemonchiffon	mediumspringgreen	rosybrown	

Fonts

- serif
- sans-serif
- cursive
- fantasy

- monospace
- Times
- Baskerville
- Verdena
- Symbol

Oneways

```
-oneways (add oneway arrows)
-onewaycolor=TEXT (color for oneway arrows)
```



Extracts

If you don't want the whole osm data to be printed that is contained in the file – no problem. As long as you have installed osmosis and this can be invoked from the current directory by command line.

Just specify the name of a place or a node id and mapgen will look for such a place. Upon success it will invoke osmosis to extract the needed data. By default a width and height of 4km (2*2km radius) is set. But of course it can be overridden.

```
-place=TEXT (Place to draw automatically; quotation marks can be used if necessary;
node id can be given alternatively; OSMOSIS REQUIRED!)
-lonrad=FLOAT (radius for place width in km, DEFAULT=2)
-latrad=FLOAT (radius for place width in km, DEFAULT=2)
```

Program will print used OSMOSIS string to STDOUT for later use...

Declutter

Usually when drawing maps (especially with lots of details) clutter may occur.

- mapgen will register an used area for each drawn label (except street labels) and won't use this area again.
- mapgen will register an used area for each drawn icon and won't use this area again.
- mapgen registers a polygon for each drawn way label and will prevent collision

Icons will be placed at the given location.

Text labels for nodes and areas will be placed on a default position and orientation. If this is not possible other positions and alignments will be tried. So labels might be moved and/or omitted at all.

At the end of map drawing program will show how many labels and icons were omitted or moved.

To prevent a special form of clutter caused by certain parallel ways the following option may be specified:

```
-declutter (declutter text; WARNING: some labels might be omitted; motorway and trunk will only be labeled in one direction)
```

- Motorways and trunks will be labeled only in one direction

mapgen tries not to put labels on ways that are too short. To do so it calculates the length of the way and the length of the text. This is done by assuming a need of 6 pixels per character for a 10pt font. Other font sizes are calculated automatically. If you find out that 6 ppc doesn't suit your used font, please reconfigure the value using the following option.

```
-ppc=<float> (pixels needed per label char using font size 10, DEFAULT=5.5)
```

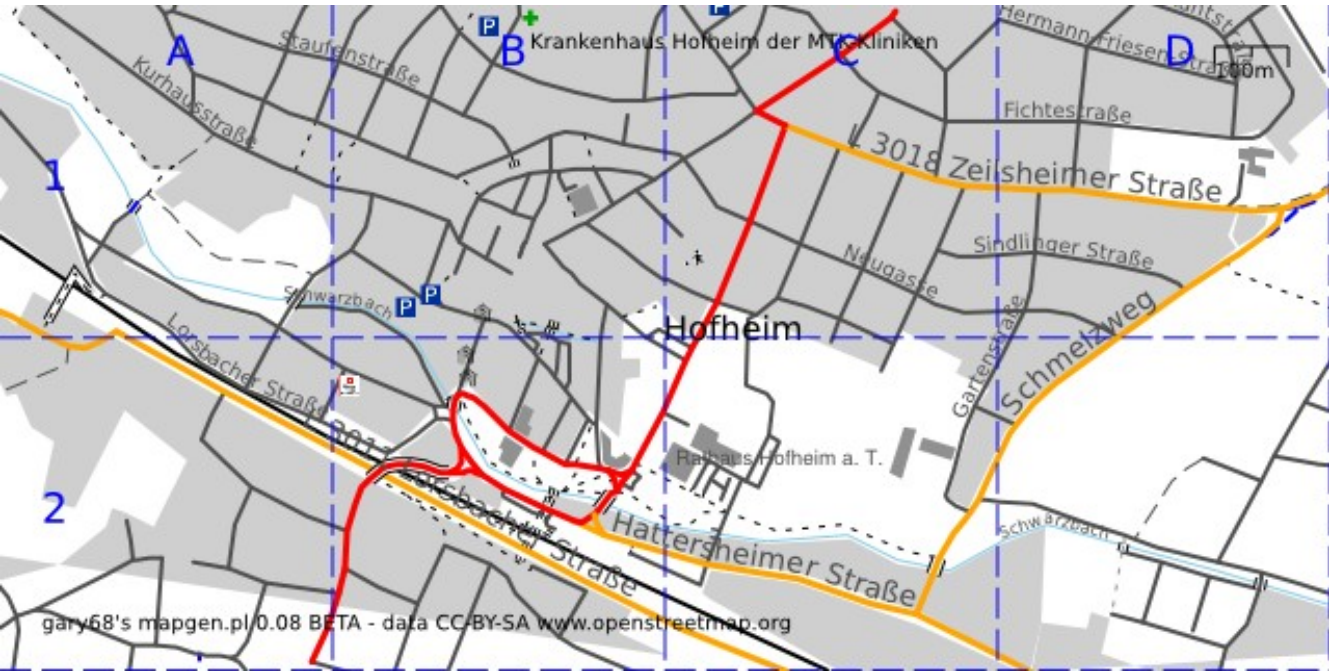
Using option -allowiconmove you can permit the program to slightly move icons if they don't fit the original position. Icons might be moved by 10 pixels in each direction. Labels are drawn afterwards and will be placed accordingly. Using this option will significantly increase the number of drawn icons in cluttered areas! (In Frankfurt using a scale of 1:50.000 150 more icons could be drawn.)

```
-allowiconmove (allows icons to be moved if they don't fit the exact position)
```


Grids, directory and stats

A grid can be laid over the map. Just specify the number of squares you want in longitude direction. The other dimension is automatic. Of course you can specify the grid color. The grid squares are labeled numerically and alphabetically.

```
-grid=<integer> (number parts for grid, 0=no grid, DEFAULT=0)
-gridcolor=TEXT (color for grid lines and labels (DEFAULT=black))
```



Street directory

mapgen can even create a street directory. It will do so including the grid squares where the street is located if the grid is turned on. Output is an unformatted street list to be further processed. The grid squares are separated by a tab. File name is name_streets.txt.

```
-dir (create street directory in separate file. if grid is enabled, grid squares will be added)
```

In the next table you can see a section of the street directory. On the left without grids, on the right with grid squares:

Burgstraße	Burgstraße.....B1
Cohausenstraße	Cohausenstraße.....A1 A2 B1
Crufterostraße	Crufterostraße.....D2
Elisabethenstraße	Elisabethenstraße...B2 C1 C2 D1 E1
Eschborner Weg	Eschborner Weg.....C1
Feldbergstraße	Feldbergstraße.....A1 B1

Poi and place list

You can print a poi and place list like the street directory. Grid option applies as well.

-poi (create list of pois)

```
Delkenheim.....C5
Diedenbergen.....F4
Eddersheim.....G5
Edeka Kundenparkplatz.....G7
Falkenberg (Keramag).....E7
Flörsheim.....F6
Geldautomat.....C1
```

Statistics

You can print a tag statistic about the usage of the keys and values. To keep the list short unimportant keys are omitted. This must be adapted in the code if desired.

The idea is to see what keys are used mostly. So you can decide for which features rules are needed.

Mapgen will print an alphabetical list of keys and values as well as a list of the most used k/v combinations. At the end of each line the program prints if it knows a rule for that k/v.

Output is separated for nodes and ways and will be written in a file called name_tagstat.txt

-tagstat (lists keys and values used in osm file; program filters list to keep them short!!! see code array noListTags)

TOP 20 LIST:

highway	residential	123	RULE
highway	footway	51	RULE
oneway	yes	38	-
highway	service	21	RULE
highway	primary	19	RULE
highway	steps	14	-
foot	yes	12	-
highway	secondary	11	RULE
building	yes	10	RULE
bicycle	yes	10	-
amenity	parking	10	RULE
surface	cobblestone	9	-
service	parking_aisle	8	-
highway	pedestrian	8	-
landuse	residential	6	RULE
highway	track	6	RULE
highway	path	5	RULE
highway	living_street	5	-
amenity	restaurant	5	-
amenity	pharmacy	5	-

Obviously we should maybe implement a rule for oneway=yes. And we can see that by far the most used tag here is highway=residential.

Coordinates Grid

You can add a grid of coordinates to your map by specifying the `-coords` option. By default the grid distance is 0.01 degrees.

This can be changed by using the `-coordsexp` parameter. The value given here is the power of 10. (-2 means 0.01 degrees, 2 means 10 degrees).

And of course the color can be changed.

```
-coords (turn on coordinates grid)
-coordsexp=INTEGER (degrees to the power of ten for grid distance; DEFAULT=-2
equals 0.01 degrees)
-coordscolor=TEXT (set color of coordinates grid)
```



Debug

Verbose will turn on lots of information to be printed while program executes. This is mostly done for debug purposes.

`-verbose`

If you want to print a map only containing multipolygons you can specify so. This is also a debug function, although a graphical one.

`-multionly (draws only areas of multipolygons; for test purposes)`