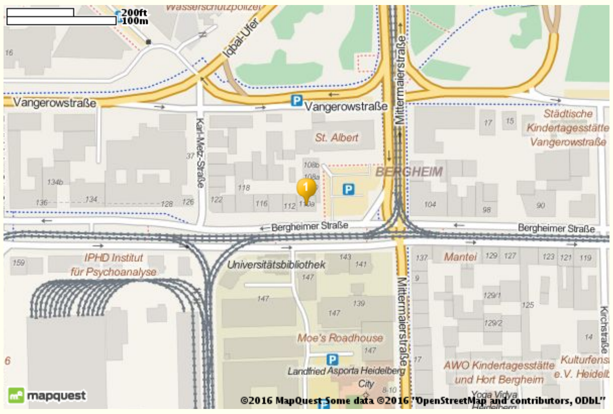
getmap.sty

v1.9

Downloading maps from OpenStreetMap, Google Maps or Google Street View



2016/06/18

Package author:
Josef Kleber

getmap@jklatex.de
http://getmap.jklatex.de/

Abstract

The goal of this package is the simplest possible provision of map images (OpenStreetMap, Google Maps and Google Street View are supported). In the simplest case, it is is sufficient to specify an address. The package loads the map using the `\write18` feature, which you must activate to use this package. The image will be downloaded by an external Lua script. You can use this script also from the command line.

Acknowledgment

I want to thank Norbert Preining, who did most of the recoding of `osmimage` (`Bash` $\rightarrow$ `Lua`). Moreover many thanks to Taco Hoekwater, Reinhard Kotucha and Heiko Oberdiek for their valuable contributions. Finally, I want to thank Doug Currie for helping me to implement the algorithm for encoded polylines in Lua.

1 Options

The following options can be used as package options with global scope, as well as options for the `\getmap` command with local scope!

1.1 General options

1.1.1 `mode (osm|gm|gsv)`

This option sets the mode, that is the source of the images. OpenStreetMap, Google Maps or Google Street View! Please note that – if used as local option (mixed modes) – the default values of the `scale`, `zoom`, `type` and `color` options for the respective mode are reset to guarantee correct download URLs!

1.1.2 `inputencoding`

This option specifies the input encoding of your file. The download script requires the strings encoded in utf8. For the safe conversion the input encoding of the file is required. Normally, you don't have to specify an encoding. The package tries to evaluate the encoding given to `inputenc` or assumes utf8. Usually that should work.

1.1.3 `overwrite (false|true)`

With this option, you can specify whether the image should be downloaded in any case. By default, the option is set to `false` in order to save bandwidth and compilation time. Nevertheless a check is performed on the existence of the image and the image will be downloaded, if it is not present. In the case of `true`, the image will be downloaded anyway! BTW, `overwrite` is equivalent to `overwrite=true`.

Note

changed default value to `getmap` in version 1.2!

1.1.4 `file (getmap)`

This option allows you to specify the name of the image (without extension).

1.2 `osm` mode

1.2.1 `key (Fmjtd|luur20u22d,75=o5-9aylh6)`

In `osm` mode, the download script requires a key in order to use the service of MapQuest. By default, it uses a key, which is registered for `getmap`. But you can register and use your own key with this option. The default key is stored in `getmap.cfg`. You can copy this file to your local TEX tree and store your own key there[1]! This file will be found after running `texhash`!

1.2.2 `scale` (`3385`)

This option allows you to specify a display scale for the map image in the range of 1692 – 221871572. You will not necessarily see a difference between 5000 and 5500. A scale value of 3385 corresponds to a zoom level of 17.

1.2.3 `zoom`

This option allows you to specify a zoom level in the range of 1 – 18. This option overwrites a possibly given scale.

Note

changed default value to `600` in version 1.2!

1.2.4 `xsize` (`600`)

This option specifies the width of the map in pixels. If you only want to slightly increase or decrease the map extract, you should adjust the size of the map. You still have full control over the size of the map in the document with the options of `\includegraphics`. (max: 3840)

1.2.5 `ysize` (`400`)

This option specifies the height of the map in pixels. (max: 3840)

1.2.6 `imagetype` (`png|jpeg|jpg|gif`)

This option allows you to specify the type of the image.

1.2.7 `type` (`map|sat|hyb`)

This option specifies the type of the map. It seems as if there would be only a few regions of Mother Earth, for which satellite and hybrid images are available.

1.2.8 `color` (`yellow_1`)

This option specifies the color of the marker. Possible colors:

http://open.mapquestapi.com/staticmap/icons.html

[1] Mapquest will deliver an url-encoded key, which must be decoded to ASCII, e.g. by Url decode

1.2.9 number (1)

This option specifies the number of the marker.

1.3 gm mode

1.3.1 scale (1)

For the free version of Google Maps the image size is limited to 640x640. You can set `scale` to a value of 2, to get exactly the same map in doubled size in pixels.

1.3.2 zoom (17)

This option allows you to specify a zoom level in the range of 0 – 21.

1.3.3 xsize (600)

This option specifies the width of the map in pixels. If you only want to slightly increase or decrease the map extract, you should adjust the size of the map. You still have full control over the size of the map in the document with the options of `\includegraphics`. (max: 640)

1.3.4 ysize (400)

This option specifies the height of the map in pixels. (max: 640)

1.3.5 imagetype (png|png8| png32|gif|jpg (progressive)|jpg-baseline (flat))

This option allows you to specify the type of the image.

1.3.6 type (roadmap|satellite|hybrid|terrain)

This option specifies the type of the map.

1.3.7 color (blue)

This option specifies the color of the marker. Possible colors:

black, brown, green, purple, yellow, blue, gray, orange, red, white or in hex format `0x3399FF`

1.3.8 number (1)

This option specifies the number of the marker. Google Maps also allows uppercase letters: [A-Z]!

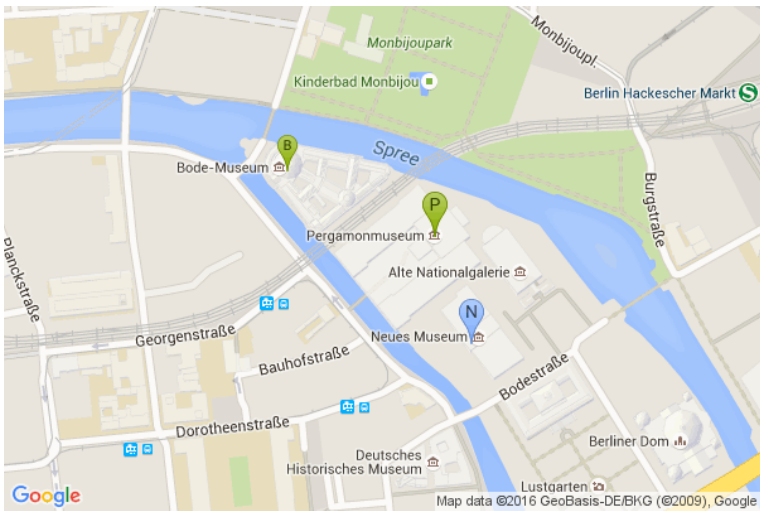
1.3.9 language (en)

This option specifies the language of the map labels. Of course, not all languages are supported for all countries. At least, english and one of the national languages should be supported. Possible option values: en, de, fr, es, it, fi, ...

1.3.10 markers

This option allows you to set more than just the standard marker, which will no longer be used! You don't have to specify an address, as Google Maps will deliver an image with all markers on the map. Nevertheless, you can specify an address, which will define the center of the map. This option expects one or more URL parameters like:

`&markers=size:mid|color:blue|label:S|loc1|loc2|...`



```
\getmap[
file=bmus1, mode=gm,
markers={&markers=size:mid|label:B|color:green|52.521847,13.394398%
         &markers=label:P|color:green|Pergamonmuseum, Berlin%
         &markers=label:N|color:blue|52.520063,13.397525}%
]{}
\includegraphics[width=10cm]{bmus1}
```

Earlier versions of this document used POIs for all museums. After an update from Google Maps, the quality of the geo codings deteriorated, at least in this example from Berlin. Using addresses or geographical coordinates usually solves this problem. See [7] for more information.

The parameters size, color and label are optional!

size tiny, mid, small

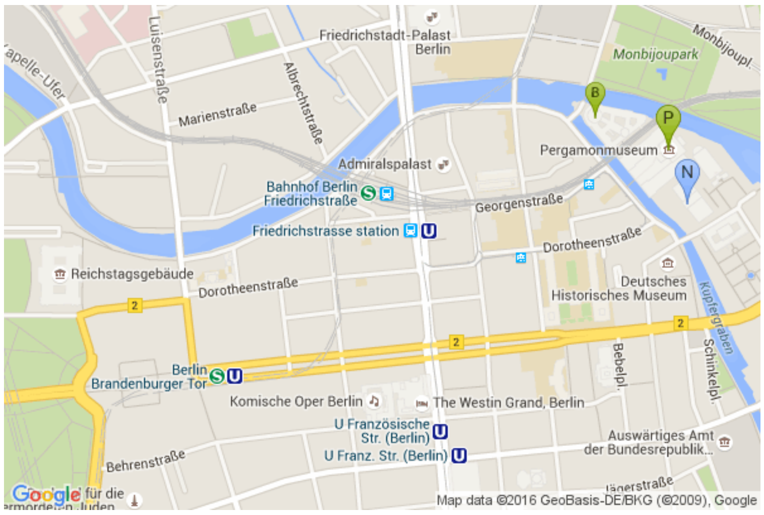
color red, black, brown, green, purple, yellow, blue, gray, orange, white, 0x1188FF

label [0-9][A-Z] (only in mid size!)

The default is a mid-sized red bubble with a black point!

1.3.11 `visible`

With this option you can specify a list of locations (separated by a pipe), which must be on the map!



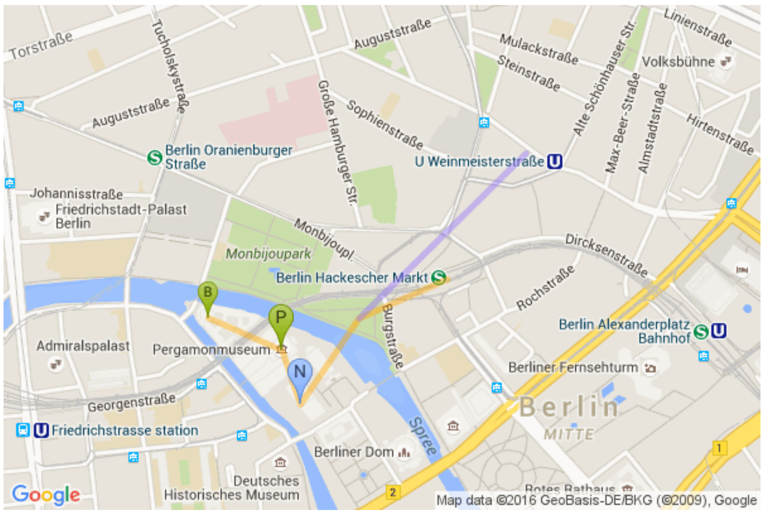
```latex
\getmap[
file=bmus2, mode=gm
markers={&markers=size:mid|label:B|color:green|52.521847,13.394398%
         &markers=label:P|color:green|Pergamonmuseum, Berlin%
         &markers=label:N|color:blue|52.520063,13.397525},%
visible={Brandenburger Tor, Berlin|Reichstagsufer 1, Berlin}]{}
\includegraphics[width=10cm]{bmus2}
```

1.3.12 path

With this option you can define one or more paths! It expects one or more URL parameters like:

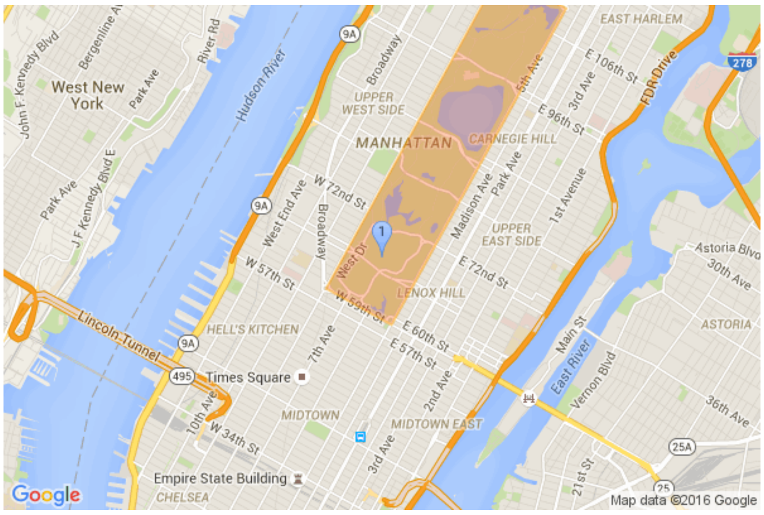
&path=weight:5|color:orange|loc1|loc2|...

```
\getmap[file=bmus3, mode=gm, language=de,
markers={&markers=size:mid|label:B|color:green|52.521847,13.394398%
         &markers=label:P|color:green|Pergamonmuseum, Berlin%
         &markers=label:N|color:blue|52.520063,13.397525},%
path={&path=weight:5|color:orange|52.521847,13.394398|%
      Pergamonmuseum, Berlin|52.520063,13.397525|%
      James-Simon-Park,Berlin|52.522649,13.402523%
      &path=weight:5|color:purple|James-Simon-Park, Berlin|%
      Weinmeisterstraße 6, Berlin}]{}
\includegraphics[width=10cm]{bmus3}
```



You can also use `fillcolor` to mark areas! In paths, you can also specify RGB32 colors, in which the last byte defines opacity, e.g. 55 (33%).

```
\getmap[file=cpny, mode=gm, zoom=13,
       path={&path=weight:2|color:orange|fillcolor:0xff641A55|
             40.764302, -73.973004|40.768044, -73.981903|%
             40.800642, -73.958193|40.796887, -73.949226|%
             40.764302, -73.973004}]{Central Park, New York}
\includegraphics[width=10cm]{cpny}
```



With small enough spaces between way points you can also defines routes!

1.3.13 **pathfile**

This option specifies the file holding the path specification. It will be loaded by the `Lua` script. You can use the `filecontents*` environment to keep the definition in your document. It should be a one line utf8-encoded file!

1.4 **gsv mode**

1.4.1 **xsize (600)**

This option specifies the width of the map in pixels. (max: 640)

1.4.2 **ysize (400)**

This option specifies the height of the map in pixels. (max: 640)

1.4.3 **heading (0)**

This option specifies the heading (direction) in degrees in the range of 0 – 360. (0: north, 90: east, ...)

1.4.4 **pitch (0)**

This option specifies the pitch (angle) of the camera view in degrees in the range of -90 – 90.

1.4.5 fov (90)

This option specifies the field of horizontal view (kind of zoom) in degrees in the range of 0 – 120.

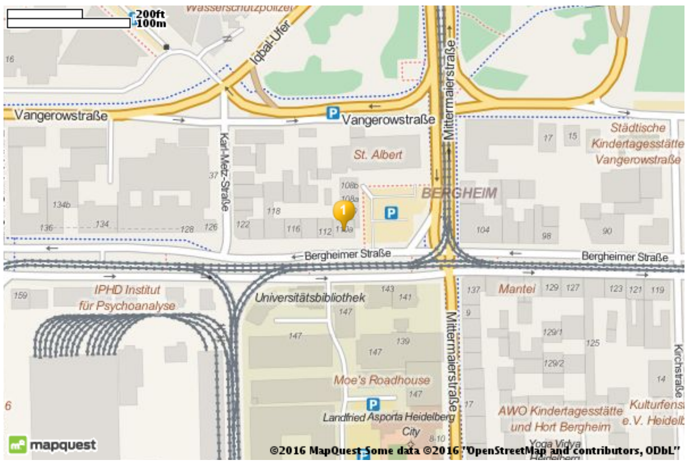
2 Command(s)

2.1 \getmap

\getmap[⟨*options*⟩]{⟨*address*⟩}

With the `\getmap` command you can download a map, if you enable `\write18` (TeXLive: `-shell-escape`, MiKTeX: `--enable-write18`). This is only necessary if you actually download an image. You can use the options described above to specify the properties of the downloaded image. After executing the command, the image is available in the current working directory!

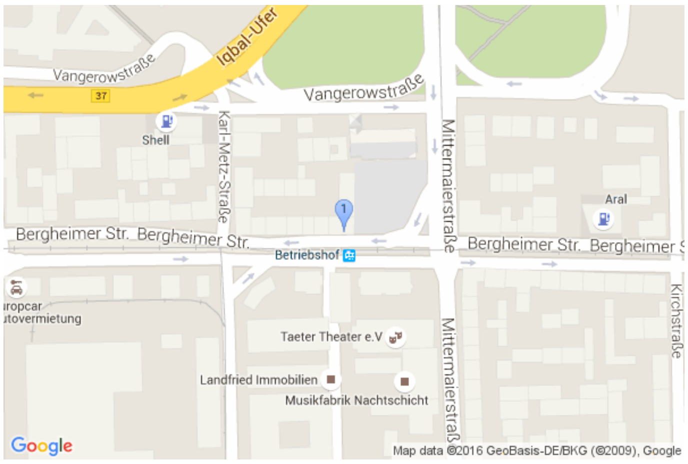
In the simplest case, you only need an address, a POI or geographic coordinates (`latitude,longitude`) to download the map. {⟨*address*⟩} must be fully expanded and must not contain macros! By default, the image is saved under the name `getmap.png`! If you need only one map (e.g. the office of Dante e.V.) in your document, it can be as simple as:


200ft
100m
Vangerowstraße
Mittermaierstraße
Karl-Metz-Straße
St. Albert
BERGHEIM
Bergheimer Straße
IPHD Institut für Psychoanalyse
Universitätsbibliothek
Moe's Roadhouse
Städtische Kindertagesstätte Vangerowstraße
Mantei
AWO Kindertagesstätte und Hort Bergheim
Kirchstraße
mapquest
©2016 MapQuest Some data ©2016 "OpenStreetMap and contributors, ODbL"

```
\getmap{Bergheimer Straße 110A, 69115 Heidelberg, Germany}
\includegraphics[width=9cm]{getmap}
```

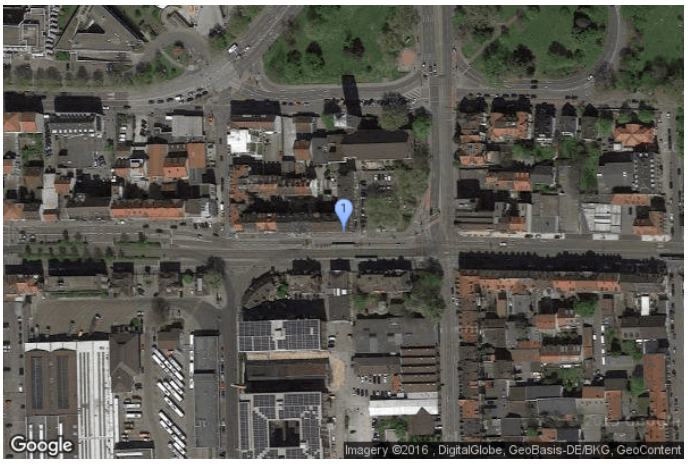
3 Examples

The same map as before from Google Maps:



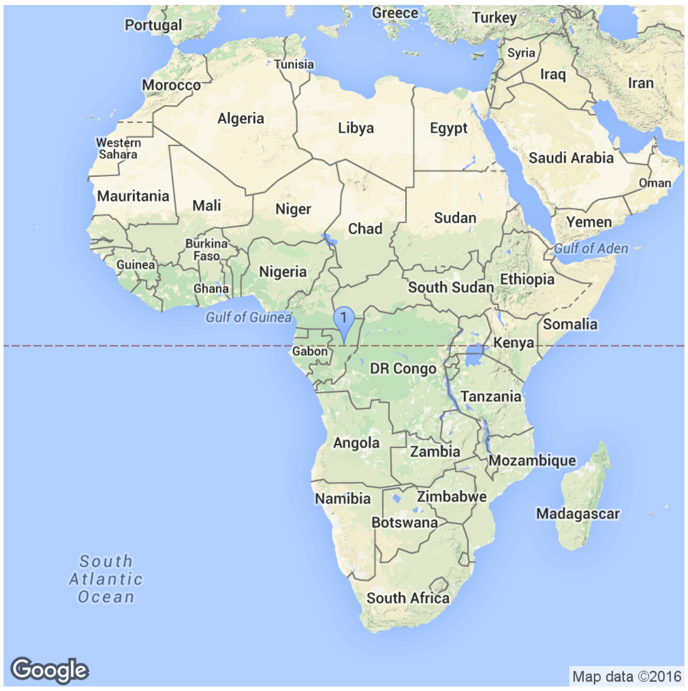
```
\getmap[file=dantegm,mode=gm]{Bergheimer Straße 110A,%
                              69115 Heidelberg, Germany}
\includegraphics[width=9cm]{dantegm}
```

The same map as satellite image:



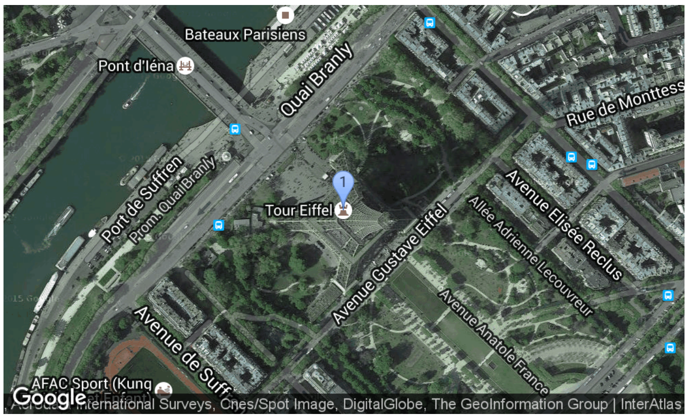
```
\getmap[file=dantegmsat,mode=gm,type=satellite]
       {Bergheimer Straße 110A, 69115 Heidelberg, Germany}
\includegraphics[width=9cm]{dantegmsat}
```

L'afrique, mon amour!



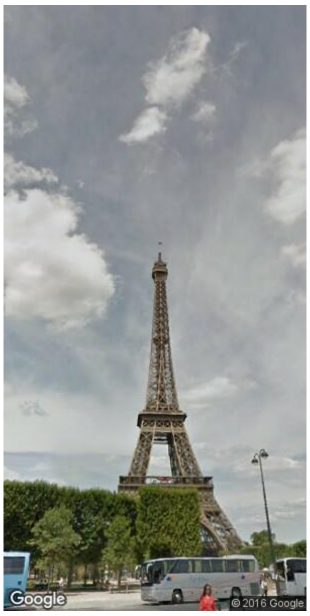
```
\getmap[file=africa,mode=gm,type=terrain,xsize=500,ysize=500,%
        scale=2,zoom=3]{0,16}
\includegraphics[width=9cm]{africa}
```

L'amour, ...



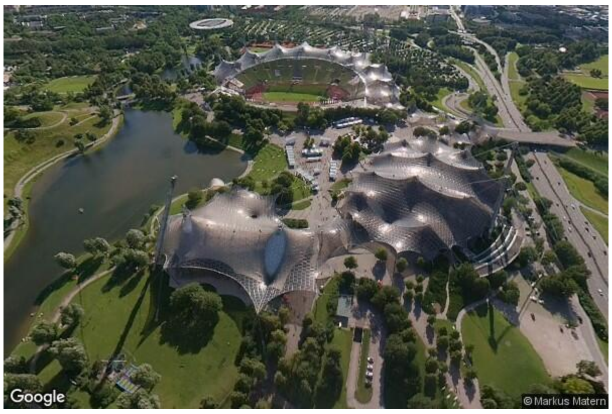
```
\getmap[file=paris,mode=gm,type=hybrid,xsize=500,ysize=300,%
        scale=2,zoom=16]{Tour Eiffel, Paris}
\includegraphics[width=9cm]{paris}
```

Street View now:



```
\getmap[file=parisgsv,mode=gsv,heading=320,pitch=30,fov=40,%
        xsize=300,ysize=600]{Avenue Piere-Loti, Paris}
\includegraphics[width=4cm]{parisgsv}
```

View from Olympic Tower Munich (Olympic Stadium and Park):



```
\getmap[file=mucoly,mode=gsv,heading=260,pitch=-40,fov=90]%
       {Olympiaturm}
\includegraphics[width=8cm]{mucoly}
```

4 The getmapdl Lua script

Basically, the `getmapdl` Lua script downloads static map images depending on command line options and allows to parse `kml`, `gpx` and `gps` (a plain list of geographical coordinate pairs `(latitude,longitude)` on each line) files and outputs `gps` or encoded polylines (epl). The script offers the following modes (`-m`):

osm downloads a static map image based on OpenStreetMap data

gm downloads a static map image based on Google Maps data

gsv downloads an image based on Google Street View data

kml2epl parses a `kml` file and outputs geographical coordinates of places and encoded polylines (`epl`) for routes and lines to `STDOUT`

kml2gps parses a `kml` file and outputs geographical coordinates

gpx2epl parses a `gpx` file and outputs encoded polylines

gpx2gps parses a `gpx` file and outputs a list of geographic coordinate pairs (`gps`)

gps2epl parses a `gps` file and outputs `epl`

gps2gps parses a `gps` file and outputs – based on a given bound – a reduced list of `gps` coordinates

The first three modes are used by `\getmap`. You may use the script also from the command line! `getmapdl -h` will give you a list of available commad line options.

The other modes are usefull for creating encoded polylines (`epl`), which is the route format of Google Maps. You can parse the following example from Google Maps in `gpx` format

```
<trkseg>
      <trkpt lon="-120.2" lat="38.5"/>
      <trkpt lon="-120.95" lat="40.7"/>
      <trkpt lon="-126.453" lat="43.252"/>
</trkseg>
```

with

```
$ getmapdl -m gpx2epl -G test.gpx
_p~iF~ps|U_ulLnnqClqNvxq‘@
```

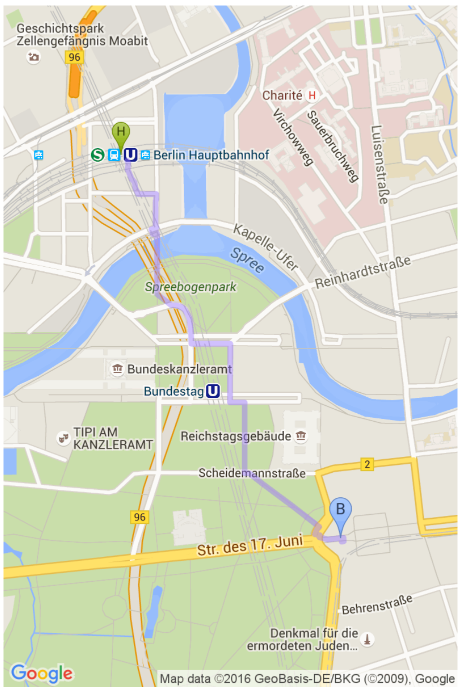
This encoded polyline can be used for the `path` or `pathfile` option of `\getmap`.

5 How to define routes

Routes are described by so called encoded polylines and can be used with `enc:polyline_data` as location specifier in a `path`. This string can contain all sorts of troublesome characters for LaTeX. `\getmap` can deal with them, with the exception of curly braces! These will break your LaTeX document. As a work-around, use the `pathfile` option. Please note that the length of the URL is limited to 2048 bytes. So, there's no way to support extreme long paths!

5.1 OpenStreetMap

OpenStreetMap does not offer routing service directly, but you can use an OpenStreetMap based route service[2] to create your route and export it to a `gpx` file[3]. It's basically a xml-packaged list of geographical coordinates. You can use the `getmapdl` script to convert a route to encoded polylines, e.g. a pedestrian route from Berlin Central Station to Brandenburg Gate:



```
\begin{filecontents*}{berlin.epl}
&path=weight:5|color:purple|enc:_xq_IcgrpA?AFE@?^BFE@A^U@CLQXEZU?
gCR?B?DBF@@?vA?D?D?BAHE@JBN@JLGFCG[DC~C?@?F?R?vA?p@iB@i@Fe@JWRSTO
f@Gh@C^A?e@?gE?w@r@?lB@hA?‘@??M?aA?]dI??O?O?Cn@cBfBeF|AeEHNVNBc@H
```

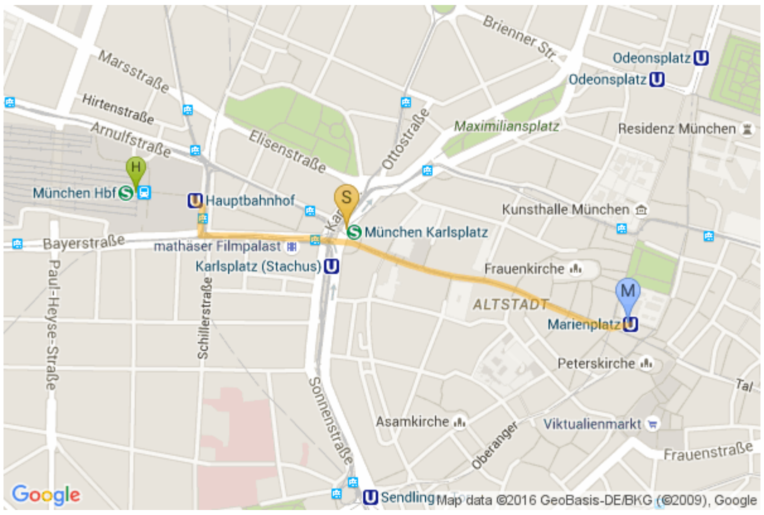
[2] http://openrouteservice.org

[3] This also means that you can visualize your own routes tracked with hardware or a software app!

```
@H_AEwA?OAMNA@N
\end{filecontents*}
\getmap[file=berlin, mode=gm, language=de, scale=2,
        xsize=400, ysize=600,
markers={&markers=size:mid|label:H|color:green|Berlin, Hbf
         &markers=label:B|color:blue|Brandenburger Tor, Berlin},
pathfile={berlin.epl}]{}
\includegraphics[width=6cm]{berlin}
```

5.2 Google Maps

One possible way is to use Google Maps' online interactive map tool[4]!



```
\begin{filecontents*}{muc.epl}
&path=weight:5|color:orange|enc:okydHa}peAXOHi@jANv@A@kJJmFAiDJuA
HmDXcBb@cC‘@gC^_DV}CJaCF{APaBVkBdAyEf@qC‘@{B\aDSwB
\end{filecontents*}
\getmap[file=muc, mode=gm,
markers={&markers=size:mid|label:H|color:green|München, Hbf
         &markers=label:S|color:yellow|Stachus, München
         &markers=label:M|color:blue|Marienplatz, München},
pathfile={muc.epl}]{}
\includegraphics[width=10cm]{muc}
```

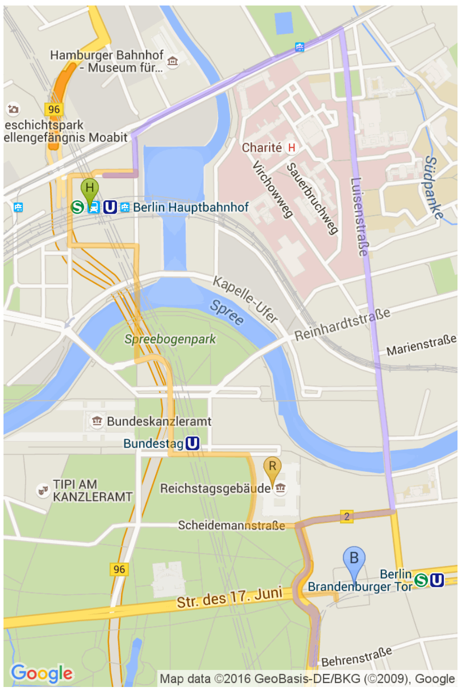
You can also use the **new** version of *My Maps*. It allows you to define markers, routes and arbitrary lines on different layers and to export these into a `kml` file, e.g berlin.kml[5]:

[4] https://developers.google.com/maps/documentation/utilities/polylineutility
[5] https://bitbucket.org/kleberj/getmap/downloads/Berlin.kml

```
$ getmapdl -m kml2epl -K Berlin.kml
Route: Route von Berlin Hbf, Moabit nach Brandenburger Tor, Paris
er Platz, Berlin
k}q_IufrpA?iFQ?gCDQBMHSm@Wq@GQIK]{AUaA}AaICIMm@Kg@EUiByIi@cDSqA_@
uBIa@a@gBpB[fAMhCS|@Gd@Ev@Ep@GfAIpC[bAMr@IbAMp@G\GtBUdCSp@MrAK~AQ
pAMhAKx@IjDc@VAB?D@@?B?dCU‘AKjAMRCPlGNtEBhABh@BVF‘@D‘@D\rB{@ZKDAH
CLAHAHBHBFHFFDNHRHJHHDDHDHBF?F?VCJCJILQJUJg@HQJMJKHAFAJ?X?bCRH@R?
AWQ?K@iBKq@GYAK@EBIFEFEFEFCFAD


Point: Berlin Hbf, Moabit  [Europaplatz 1, 10557 Berlin, Deutschl
and]
52.52581820000001,13.3695451


Point: Brandenburger Tor, Pariser Platz, Berlin  [Ebertstraße 21,
 10117 Berlin, Deutschland]
52.5159099,13.3773202


Point: Berlin Hbf  [arrive with train]
52.5249948,13.368988


Point: Reichstag  [nice view from the roof]
52.5185973,13.3758974


Point: Brandenburger Tor  [once behind the wall]
52.5163514,13.3789873


Route: Route von Berlin Hbf, Moabit nach Pariser Platz, Berlin
k}q_IufrpA?iFQ?@hH@H@F@B@B@?DB?\?|B@F@DB@B@~FCB?X@V??O?O?O?wF@W?o
@?k@?Y?eAvAD‘@?‘@ARCFAP?hAAZ?B?D?J?B?DAJB^@T?~B?MsAAIAEAM?S@OBM?A
@IDKBKDKFKFIJEFCJAJATAL?XAN?P?VAlBCbB?L@RB?kH@}Gp@]|@ObA?BAz@JTF^
LZN?wF@aCE]~Am@\KREJCNANAH@HBJHHFLPP\DBFFDB@@@?D@F?JAJCKiEAQ?K@I?
KBSXC


Point: Berlin Hbf, Moabit  [Europaplatz 1, 10557 Berlin, Deutschl
and]
52.52581820000001,13.3695451


Point: Deutscher Bundestag Redaktion Das Parlament, Berlin  [Plat
```

```
z der Republik 1, 10557 Berlin, Deutschland]
52.518502000000005,13.3751849


Point: Pariser Platz, Berlin  [Pariser Platz 1, 10117 Berlin, Deu
tschland]
52.5160749,13.3783013
```

Now, you can take these data for your map:



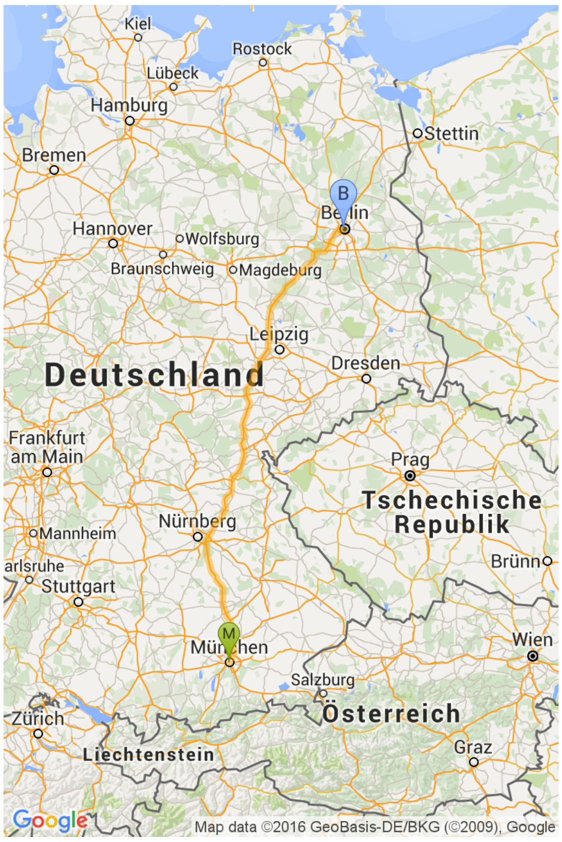
```
\begin{filecontents*}{berlin2.epl}
&path=weight:5|color:orange|enc:k}q_IufrpA?iFQ?@hH@H@F@B@B@?DB?\?
|B@F@DB@B@~FCB?X@V??O?O?O?wF@W?o@?k@?Y?eAvAD‘@?‘@ARCFAP?hAAZ?B?D?
J?B?DAJB^@T?~B?MsAAIAEAM?S@OBM?A@IDKBKDKFKFIJEFCJAJATAL?XAN?P?VAl
BCbB?L@RB?kH@}Gp@]|@ObA?BAz@JTF^LZN?wF@aCE]~Am@\KREJCNANAH@HBJHHF
LPP\DBFFDB@@@?D@F?JAJCKiEAQ?K@I?KBSXC&path=weight:5|color:purple|
enc:k}q_IufrpA?iFQ?gCDQBMHSm@Wq@GQIK]{AUaA}AaICIMm@Kg@EUiByIi@cDS
qA_@uBIa@a@gBpB[fAMhCS|@Gd@Ev@Ep@GfAIpC[bAMr@IbAMp@G\GtBUdCSp@MrA
K~AQpAMhAKx@IjDc@VAB?D@@?B?dCU‘AKjAMRCPlGNtEBhABh@BVF‘@D‘@D\rB{@Z
KDAHCLAHAHBHBFHFFDNHRHJHHDDHDHBF?F?VCJCJILQJUJg@HQJMJKHAFAJ?X?bCR
H@R?AWQ?K@iBKq@GYAK@EBIFEFEFEFCFAD
\end{filecontents*}
\getmap[file=berlin2, language=de, xsize=400, ysize=600,
        scale=2, mode=gm,
markers={&markers=size:mid|label:H|color:green|52.5249948,13.3689
```

```
88
         &markers=size:mid|label:R|color:yellow|52.5185973,13.375
8974
         &markers=label:B|color:blue|52.5163514,13.3789873},
pathfile={berlin2.epl}]{}%
\includegraphics[width=6cm]{berlin2}%
```

5.2.1 Long routes

Long routes are defined by a huge number of way points, but the URL length is limited to 2048 bytes. The following example[6] (Stachus, Munich → Brandenburg Gate, Berlin) consists of more than 6000 way points. A created polyline would hugely exceed the URL length limit!

After transfering the `kml` file into a `gps` file, you can use the `gps2gps` mode of `getmapdl` to reduce the number of way points by a given bound. It takes a new pair of `gps` coordinates only if the difference of latitude or longitude is larger than the given bound! Finally, you can use the new `gps` file to create an encoded polyline.



[6] https://bitbucket.org/kleberj/getmap/downloads/MucBer.kml

```
$ getmapdl -m kml2gps -K MucBer.kml >MucBer.gps

$ cat MucBer.gps
Route: Route von Stachus, München nach Brandenburger Tor, Pariser
 Platz, Berlin
48.13903,11.56556
48.1392,11.56562
[ ... many, many way points ...]
52.5159,13.37735
52.51591,13.37732


Point: Stachus, München  [Karlsplatz 10, 80335 München, Deutschla
nd]
48.13903,11.56556


Point: Brandenburger Tor, Pariser Platz, Berlin  [Ebertstraße 21,
 10117 Berlin, Deutschland]
52.51591,13.37732

$ getmapdl -m gps2gps -G MucBer.gps -B 0.025 >MucBerR.gps

Route 1: reduced gps coordinates (Bound = 0.025): 6119 -> 193

$ getmapdl -m gps2epl -G MucBerR.gps

Route: Route von Stachus, München nach Brandenburger Tor, Pariser
 Platz, Berlin

}cydHw{qeAc|CmbCcoCq{CswDk‘Be~C_oAqcDe_@wdD~eAs_Dt~AsbDn{A{aEhk@m
aD‘i@kcDhAk}CoHcnD|Hg}CvVk{Cmx@i|Cpe@c|Cp|@kaClxDk~CzxCqnDzkAk‘Df
tAc}CrrAonDtqBu|C~m@e~CoGc}Csd@wnD{EyjDr@s}C‘@}iDoBskCn|CgbCp_D_p
B~aD_cB~aDc}CgOssBr}Ci‘Cj{CwzAl_Dc|Cnp@w{C~dAagDnk@o{Cnu@o{ChcCcl
EhdBalDni@u{CpUagFfRa}CfLq|CxQu{Cr\m{C{sBwqC}|Cm}CucDuhD}jDa}GiuD
{|CkuBm~CeqCkoAwdDkyAkmDc‘CkiDmaEm‘Ck|CeCakBwfD}_D{gBo|CbUw}C~yAw
|Cqc@{}C}‘BumC_|Cqr@kvDmSk}Cu{CiaAkbDbGi_DoGw_DyLk‘DxH{_Dmb@q}Cyr
@{cAkaDcs@{_DarAy{C_|By_Dy~Ck‘Ci}CeMm|CqiCi|CmzAmgDkUm‘D_wBg}CiVu
~C|aBg|Cva@i{Cc\}zBv~Cw‘DyyBq_D_gB{|CoEu{ChLacDfSy|CeBk{CjsC}~Ca~
@_}Cil@w|Coz@e_DidCecDivBe|Cqc@{{CemAa_DcyBq|C}|A_|CtuAsfE|dDwzDr
z@e~CvXo~C|c@s{CicCirB}jDkq@q}C{dDiuBkcDsnAm}CsnAk}DuI{|CkMmvFmsE
y|CmuBc~Cw{C_‘Do~CqhEujHekBc|CkvCmaDufDqgC_}Csh@e|Cnq@kuDemAm}Ccz
A{bEgg@cvDlyAiaDbUi{Cjw@i}CsLy}F}I_cDc{@gjDi_@o_Ds]{|D_c@o|Cy}A}q
BubDo}CowCoyBe|Cy_Du_CotAg}C}}@obD{sCu{CkrB_jDqoFcmCsmBq_D_l@w}Ca
z@mdD}eBa‘DmuBonE_nA_|Cw‘BaaDalDqfCu_CwyDenBqwDqzA{~C}mDkgEcdAmcD
si@{‘DabBq{Ccx@s{CsfDwsDueDvR_hD_WcuAu_Dsj@g~DkF{kFf_@waDcOa‘DkDc
```

```
|C}_DceCiu@{aDu|Ckv@m{CcpAiz@k|CmeD}eBeaEedCg‘CeaDe‘Cy{CwdBipEgJ_
gDwOy{Fq@at@
```

Taking a look into the log file, we find:

```
getmapdl.lua:
url = http://maps.googleapis.com/maps/api/staticmap? ...
url length = 1866 bytes
output = mucber.png
```

With 193 way points we almost reached the URL length limit of 2048 bytes. The accuracy of the encoded polyline is obviously good enough. So, about 200 way points seem to be a good choice. But the length of an encoded pair of `gps` coordinates depends on the space between two points and may vary between 2 and 8 bytes!

6 Implementation

```
1 ⟨*package⟩
```

First, we provide the LaTeX package `getmap`.

```
2 \NeedsTeXFormat{LaTeX2e}%
3 \ProvidesPackage{getmap}[2016/06/18 v1.9 getmap.sty - Josef Kleber (C) 2014,2016]%
```

We need a few packages!

```
4 \RequirePackage{xkeyval}%
5 \RequirePackage{stringenc}%
6 \RequirePackage{ifthen}%
```

Newer versions of LuaTeX v0.85+ no longer supports `\write18`! Therefore, we use `shellesc` instead.

```
7 \RequirePackage{shellesc}%
```

We provide a macro `\GM@JK@define@key`, which defines package options with global scope and options for `\getmap` with local scope. It takes four arguments {⟨*prefix*⟩}, {⟨*package*⟩},{⟨*option*⟩} and {⟨*default*⟩}.

```
8 \newcommand*\GM@JK@define@key[4]%
9 {%
10   \expandafter\gdef\csname#1@#3\endcsname{#4}%
11   \define@key{#2.sty}{#3}[#4]%
12   {%
13     \expandafter\gdef\csname#1@#3\endcsname{##1}%
14   }%
15   \define@key{#2}{#3}%
16   {%
17     \expandafter\def\csname#1@#3\endcsname{##1}%
```

```
  }%
}%
\newcommand*\GM@JK@define@key@detok[4]%
{%
  \expandafter\gdef\csname#1@#3\endcsname{#4}%
  \define@key{#2.sty}{#3}[#4]%
  {%
    \expandafter\gdef\csname#1@#3\endcsname{\detokenize{##1}}%
  }%
  \define@key{#2}{#3}%
  {%
    \expandafter\def\csname#1@#3\endcsname{\detokenize{##1}}%
  }%
}%
```

Now, we can use this macro to define our options.

```
\GM@JK@define@key{GM@JK}{getmap}{mode}{osm}%
\GM@JK@define@key{GM@JK}{getmap}{key}{}%
\GM@JK@define@key{GM@JK}{getmap}{xsize}{600}%
\GM@JK@define@key{GM@JK}{getmap}{ysize}{400}%
\GM@JK@define@key{GM@JK}{getmap}{scale}{3385}%
\GM@JK@define@key{GM@JK}{getmap}{zoom}{}%
\GM@JK@define@key{GM@JK}{getmap}{type}{map}%
\GM@JK@define@key{GM@JK}{getmap}{imagetype}{png}%
\GM@JK@define@key{GM@JK}{getmap}{color}{yellow_1}%
\GM@JK@define@key{GM@JK}{getmap}{number}{1}%
\GM@JK@define@key{GM@JK}{getmap}{heading}{0}%
\GM@JK@define@key{GM@JK}{getmap}{fov}{90}%
\GM@JK@define@key{GM@JK}{getmap}{pitch}{0}%
\GM@JK@define@key{GM@JK}{getmap}{language}{en}%
\GM@JK@define@key@detok{GM@JK}{getmap}{markers}{}%
\GM@JK@define@key@detok{GM@JK}{getmap}{path}{}%
\GM@JK@define@key@detok{GM@JK}{getmap}{visible}{}%
\GM@JK@define@key{GM@JK}{getmap}{pathfile}{}%
\GM@JK@define@key{GM@JK}{getmap}{file}{getmap}%
\GM@JK@define@key{GM@JK}{getmap}{inputencoding}{}%
\GM@JK@define@key{GM@JK}{getmap}{overwrite}{true}%
```

For options without default value, we define reasonable default values! We overwrite the default for `overwrite`, because we don't want `overwrite` to be `true` by default, but that `overwrite` is equivalent to `overwrite=true`!

Moreover, we load `getmap.cfg` to set the default key. You can copy this file to your local TEX tree and replace the key with your own!

We try to use the input encoding specified for `inputenc` or `utf8` instead.

```
\gdef\GM@JK@overwrite{false}%
\gdef\GM@JK@key{}%
%
\IfFileExists{getmap.cfg}%
{%
```

```
  \input{getmap.cfg}%
}%
{%
  \gdef\GM@JK@key{Fmjtd|luur20u22d,75=o5-9aylh6}%
}%
%
\@ifpackageloaded{inputenc}%
{%
  \gdef\GM@JK@inputencoding{\inputencodingname}%
}%
{%
  \gdef\GM@JK@inputencoding{utf8}%
}%
%
```

Later, we will need a switch, if `\write18` is enabled.

```
\newif\ifGM@JK@writexviii\GM@JK@writexviiifalse%
%
```

We execute the package options to define and set the option macros.

```
\ExecuteOptionsX{mode,xsize,ysize,scale,zoom,type,imagetype,color,number,file,heading,fov,
%
\ProcessOptionsX\relax%
%
```

We need to reset some defaults in `gm` mode.

```
%
\ifthenelse{\equal{\GM@JK@mode}{gm}}%
{%
  \gdef\GM@JK@scale{1}%
  \gdef\GM@JK@zoom{17}%
  \gdef\GM@JK@type{roadmap}%
  \gdef\GM@JK@color{blue}%
}%
{}%
%
```

We check if `\pdf@shellescape` is available to test if `\write18` is enabled.

If `false`, we assume `\write18` is available and hope for the best.

If `true`, we set the switch \GM@JK@writexviii accordingly!

```
%
\ltx@IfUndefined{pdf@shellescape}%
{%
  \PackageInfo{getmap}{\pdf@shellescape is undefined}%
  \PackageInfo{getmap}{can not test if \write18 is available}%
  \GM@JK@writexviiitrue%
}%
{%
  \PackageInfo{getmap}{\pdf@shellescape is available}%
  \ifnum\pdf@shellescape=1\relax%
    \PackageInfo{getmap}{\write18 enabled}%
    \GM@JK@writexviiitrue%
  \else%
    \GM@JK@writexviiifalse%
  \fi%
}%
%
```

We define a macro that is executed as `\ShellEscape` call. First, we test if `\write18` is enabled and issue a package error if not! Otherwise we execute `\ShellEscape` depending on the mode

```
\newcommand*\GM@JK@shellescape%
{%
  \ifGM@JK@writexviii\relax%
    \ifthenelse{\equal{\GM@JK@mode}{osm}}%
    {%
      \ShellEscape{getmapdl \space-l\space "\GM@JK@location@string"%
                            \space-m\space osm%
                            \space-k\space "\GM@JK@key@string"%
                            \space-x\space \GM@JK@xsize%
                            \space-y\space \GM@JK@ysize%
                            \space-z\space "\GM@JK@zoom"%
                            \space-s\space \GM@JK@scale%
                            \space-t\space \GM@JK@type%
                            \space-i\space \GM@JK@imagetype%
                            \space-c\space "\GM@JK@color"%
                            \space-n\space \GM@JK@number%
                            \space-o\space \GM@JK@file}%
    }%
    {%
      \ifthenelse{\equal{\GM@JK@mode}{gm}}%
      {%
        \ShellEscape{getmapdl \space-l\space "\GM@JK@location@string"%
                              \space-m\space gm%
                              \space-x\space \GM@JK@xsize%
                              \space-y\space \GM@JK@ysize%
                              \space-z\space \GM@JK@zoom%
                              \space-s\space \GM@JK@scale%
                              \space-t\space \GM@JK@type%
                              \space-i\space \GM@JK@imagetype%
                              \space-c\space "\GM@JK@color"%
                              \space-n\space \GM@JK@number%
                              \space-L\space "\GM@JK@language"%
                              \space-M\space "\GM@JK@markers@string"%
                              \space-C\space "\GM@JK@location@string"%
                              \space-P\space "\GM@JK@path@string"%
                              \space-p\space "\GM@JK@pathfile"%
                              \space-V\space "\GM@JK@visible@string"%
                              \space-o\space \GM@JK@file}%
      }%
      {%
        \ifthenelse{\equal{\GM@JK@mode}{gsv}}%
        {%
          \ShellEscape{getmapdl \space-l\space "\GM@JK@location@string"%
                                \space-m\space gsv%
                                \space-x\space \GM@JK@xsize%
                                \space-y\space \GM@JK@ysize%
                                \space-H\space \GM@JK@heading%
                                \space-F\space \GM@JK@fov%
                                \space-T\space \GM@JK@pitch%
                                \space-o\space \GM@JK@file}%
        }%
        {%
          \PackageError{getmap}{invalid mode}{invalid mode! Use osm, gm or gsv!}%
        }%
      }%
    }%
  \else%
    \PackageError{getmap}{\write18 disabled}%
                         {\write18 disabled\MessageBreak%
                          Use -shell-escape (TeXLive)\MessageBreak%
                          or\space\space--enable-write18 (MiKTeX)}%
  \fi%
}%
```

\getmap Here, we define the user command to download the map.

\getmap[⟨*options*⟩]{⟨*address*⟩}

```
\newcommand*\getmap[2][]%
{%
```

We start a group to keep the setting of options local. Then we test the mode to reset some defaults! Finally, we set the local options again to override defaults if necessary!

```
  \begingroup%
    \setkeys{getmap}{#1}%
    \ifthenelse{\equal{\GM@JK@mode}{gm}}%
    {%
      \def\GM@JK@scale{1}%
      \def\GM@JK@zoom{17}%
      \def\GM@JK@type{roadmap}%
      \def\GM@JK@color{blue}%
    }%
    {}%
    \ifthenelse{\equal{\GM@JK@mode}{osm}}%
    {%
      \def\GM@JK@scale{3385}%
      \def\GM@JK@zoom{}%
      \def\GM@JK@type{map}%
      \def\GM@JK@color{yellow␣1}%
    }%
```

```
{}%
\setkeys{getmap}{#1}%
```

In `gsv` mode, we have an implicit `imagetype=jpg`. Therefore, we have to set it to allow the later test on the existence of the image file!

```
\ifthenelse{\equal{\GM@JK@mode}{gsv}}%
{\def\GM@JK@imagetype{jpg}}{}%
\PackageInfo{getmap}{using \GM@JK@inputencoding\space encoding}%
\def\GM@JK@location{#2}%
```

`texlua` expects its arguments encoded in `utf8`!

```
\StringEncodingConvert%
  {\GM@JK@location@string}%
  {\detokenize\expandafter{\GM@JK@location}}%
  {\GM@JK@inputencoding}{utf-8}%
\StringEncodingSuccessFailure%
{%
  %success
}%
{% failure
  \errmessage{Converting to UTF-8 failed}%
}%
\StringEncodingConvert%
  {\GM@JK@key@string}%
  {\detokenize\expandafter{\GM@JK@key}}%
  {\GM@JK@inputencoding}{utf-8}%
\StringEncodingSuccessFailure%
{%
  %success
}%
{% failure
  \errmessage{Converting to UTF-8 failed}%
}%
\StringEncodingConvert%
  {\GM@JK@markers@string}%
  {\GM@JK@markers}%
  {\GM@JK@inputencoding}{utf-8}%
\StringEncodingSuccessFailure%
{%
  %success
}%
{% failure
  \errmessage{Converting to UTF-8 failed}%
}%
\StringEncodingConvert%
  {\GM@JK@path@string}%
  {\GM@JK@path}%
  {\GM@JK@inputencoding}{utf-8}%
\StringEncodingSuccessFailure%
{%
```

```
      %success
    }%
    {% failure
      \errmessage{Converting to UTF-8 failed}%
    }%
    \StringEncodingConvert%
      {\GM@JK@visible@string}%
      {\GM@JK@visible}%
      {\GM@JK@inputencoding}{utf-8}%
    \StringEncodingSuccessFailure%
    {%
      %success
    }%
    {% failure
      \errmessage{Converting to UTF-8 failed}%
    }%
```

We check, if `overwrite` is `true` and download the map. If not, we check if the image is already in the working directory. If not, we download the image!

```
    \ifthenelse{\equal{\GM@JK@overwrite}{true}}%
    {%
      \GM@JK@shellescape%
    }%
    {%
      \IfFileExists{\GM@JK@file.\GM@JK@imagetype}%
      {%
        \PackageInfo{getmap}{overwrite=false; (\GM@JK@file.\GM@JK@imagetype)%
                             using existing file!}%
      }%
      {%
        \PackageInfo{getmap}{overwrite=false; (\GM@JK@file.\GM@JK@imagetype)%
                             file does not exist! downloading ...}%
        \GM@JK@shellescape%
      }%
    }%
  \endgroup%
}%
```

⟨/package⟩

7 References

[1] Google, Inc. Encoded Polyline Algorithm Format, 2014. https://developers.google.com/maps/documentation/utilities/polylinealgorithm.

[2] Google, Inc. Google Street View Image API, 2014. https://developers.google.com/maps/documentation/streetview/index.

[3] Google, Inc. Interactive Polyline Encoder Utility, 2014. https://developers.google.com/maps/documentation/utilities/polylineutility.

[4] Google, Inc. Static Maps API V2 Developer Guide, 2014. https://developers.google.com/maps/documentation/staticmaps/.

[5] Josef Kleber. Berlin: Hbf Berlin - Brandenburger Tor (getmap Test), 2014. https://bitbucket.org/kleberj/getmap/downloads/Berlin.kml.

[6] Josef Kleber. MucBer: München Stachus -> Berlin Brandenburger Tor, 2014. https://bitbucket.org/kleberj/getmap/downloads/MucBer.kml.

[7] Josef Kleber. Google Maps 'Implicit Positioning of the Map' broken?, 2016. https://stackoverflow.com/questions/34653500/google-maps-implicit-positioning-of-the-map-broken.

[8] MapQuest, Inc. Compressed Lat/Lng Encoding/Decoding, 2014. http://open.mapquestapi.com/common/encodedecode.html.

[9] MapQuest, Inc. Introducing the Data Manager API Web Service, 2014. http://developer.mapquest.com.

[10] MapQuest, Inc. MapQuest Open Platform Web Services, 2014. http://open.mapquestapi.com/.

[11] MapQuest, Inc. Static Map Service: Standard Icons, 2014. http://open.mapquestapi.com/staticmap/icons.html.

[12] MapQuest, Inc. Zoom To Scale Mapping, 2014. http://open.mapquestapi.com/staticmap/zoomToScale.html.

[13] OpenRouteService.org. Routing with user-generated, collaboratively collected free geodata., 2014. http://openrouteservice.org.

8 Change History

v1.0
- General: CTAN upload 23

v1.1
- \getmap: Bugfix: problem in URL when using \usepackage[utf8]{inputenc} 27

v1.2
- General: added getmap.cfg to store default key (FR by Ulrike Fischer) 24
  - added support for Google Maps 23
  - changed default values of xsize (600) and file (getmap) 24
  - renamed osmimage.lua to getmapdl.lua 23

v1.3
- General: added support for Google Street View 23

v1.4
- General: getmapdl.lua: added options language, markers, visible, path and pathfile in gm mode 23

v1.5
- General: added gpx2gps bash script 23

v1.6
- General: getmapdl.lua: added gpx2epl, gps2epl and gpx2gps modes 23
  - removed gpx2gps bash script . 23

v1.7
- General: getmapdl.lua: added kml2gps and kml2epl modes . 23
  - getmapdl.lua: added multi track support for gpx2gps and gpx2epl mode 23
  - getmapdl.lua: revised gps2epl mode 23

v1.8
- General: getmapdl.lua: added gps2gps mode to reduce way points with a given bound (default: 0.1) 23
  - getmapdl.lua: added check of URL length and output to log 23
  - getmapdl.lua: added multi route support for gps2epl mode 23
  - getmapdl.lua: added rounding of gps coordinates to kml2gps and gpx2gps modes 23

v1.9
- \getmap: Bugfix: restore mode defaults for scale, zoom, type and color options to guarantee correct download URLs 27
- General: LuaTEX v0.85+ fix: use shellesc's \ShellEscape instead of \write18 26