

OS Vectormap to Garmin: HowTo

Introduction

The OS Vectormap project provides tools and instructions to convert freely-available Ordnance Survey data into maps for your Garmin device or PC. Because these are vector maps they have the advantage of staying clear and sharp when you zoom right in.

This document relates to version 1.3 of the project. Version 1.3 contains several important updates to allow the maps to build properly on 32 bit and lower memory systems. **These instructions have also changed in some places compared to the previous version** and should help you to avoid installation issues experienced by some users. Please read and follow them carefully.

This document aims to give as non-technical as possible a description of how to get vector maps working on your Garmin GPS device using freely available open source Windows software. The process requires a few different stages but it's just a matter of working through these one by one.

The time taken to build the map depends on how fast your PC is and especially how much memory it's got. A fast machine with 16 GB of memory can compile a 100 km square in about fifteen minutes while on a smaller system it can take two to three hours.

Overview

The Ordnance Survey releases its Vectormap District mapping of the UK for free¹ but deliberately limits it by excluding data about footpaths and bridleways. The project overcomes this by combining the OS data with rights-of-way information that we can extract from the OpenStreetMap project.

The steps we shall be following are as follows:

1. install the processing tools
2. download the OS Vectormap District data for the areas required
3. download the OpenStreetMap footpath data
4. configure the settings file
5. build the map
6. install the map on your GPS

The second part of the document describes how to customise your map, and includes examples of how to add new features and change the styling.

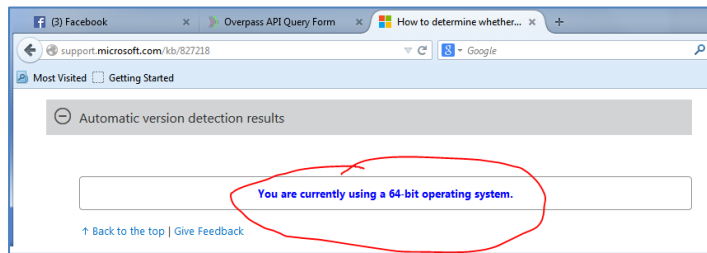
Step 1: install the processing tools

a) verify whether you are running 32 or 64 bit Windows

Some of the tools have a different version for 64 bit windows so it is important to be sure what you are running and it's worthwhile to check even if you have a pretty good idea already.

¹ released under the OS Open Data licence (<http://www.ordnancesurvey.co.uk/docs/licences/os-opendata-licence.pdf>)

Browse the following page: <http://support.microsoft.com/kb/827218>. This should automatically detect what you are running. Scroll down the page to the 'Automatic version detection results' section.



If for some reason the automatic version detection method doesn't work then follow the instructions on the page for 'manual detection'.

b) Check your Java installation

Several of the tools that we will be using during the build rely on Java and you need to have at least version 7 installed.

To check your Java installation open a windows command prompt (*start->all programs->accessories->command prompt*), type `java -version` and press return. The results should look something like the screenshot on the right.

```
Command Prompt
E:\Users\Martin>java -version
java version "1.7.0_60"
Java(TM) SE Runtime Environment (build 1.7.0_60-b19)
Java HotSpot(TM) 64-Bit Server VM (build 24.60-b09, mixed mode)
E:\Users\Martin>
```

If Java reports a version lower than 1.7 or your system can't find Java at all, then you will need to install an updated version. This is easily done by following the straightforward instructions at http://www.java.com/en/download/help/windows_manual_download.xml. You should install the latest version (currently Java 8). You only need the standard installation, not the one for developers.

Occasionally you may find that although you have an up-to-date installation of Java your system can't find it from the command line or may insist on using an older version. If this happens to you then it is safest to completely remove all current and older versions using the Java un-installation tool: http://www.java.com/en/download/faq/uninstaller_toolinfo.xml and then re-install it from scratch.

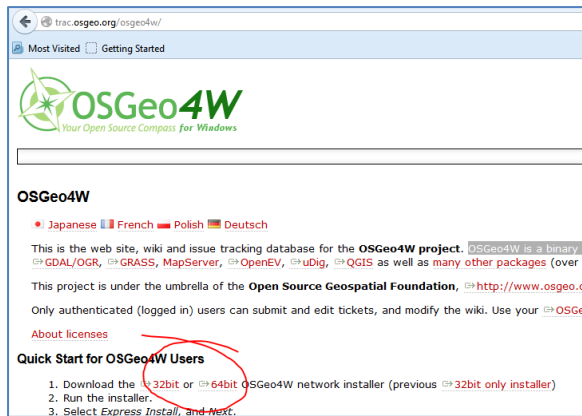
c) install OSGeo4W

Note: this section has been updated with some important information about Java

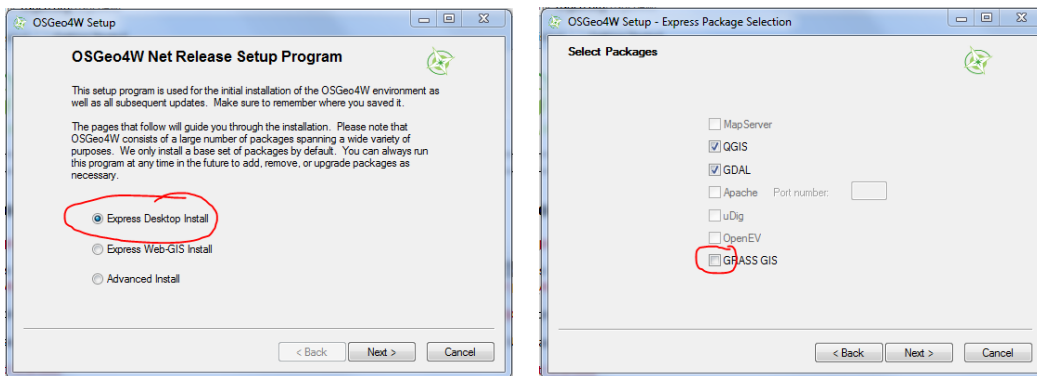
OSGeo4W is like a workbench that we will use for running many of the other tools that we will be using. Installing it sets up a lot of things in the background that would be very technical and time consuming to configure manually².

OSGeo4W is very straightforward to install. Just go to its website at <http://trac.osgeo.org/osgeo4w/> and click the link for either the 32 bit or the 64 bit version **to match your version of Windows**.

² If you prefer you can nonetheless avoid using OSGeo4W by manually installing GDAL with the python bindings, for example by following <http://pythongisandstuff.wordpress.com/2011/07/07/installing-gdal-and-ogr-for-python-on-windows/> - however I would recommend the OSGeo4W method if you plan to use the `convert_python` action (see later).

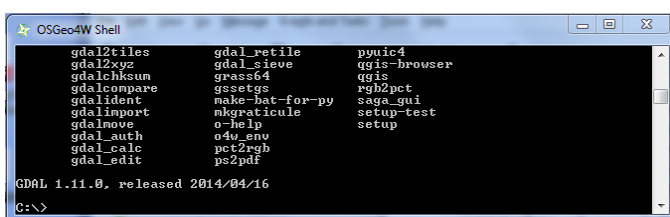


Run the installer. This will pop up a window as shown below left. Change the selection to *Express Desktop Install* then click next. On the following screen (see right) un-tick the box marked *GRASS GIS*. If you want a faster download then you can un-tick the box for QGIS too.



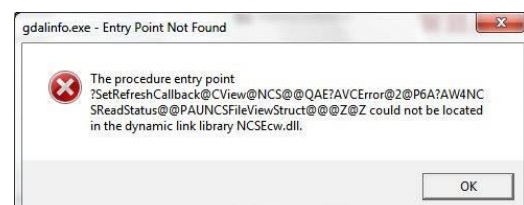
Then go ahead and install the package accepting the default values for everything else.

To check the installation find the new start menu folder for OSGeo4W (under *all Programs*) and click the item called *OSGEO4W Shell*. That should open up a command window like the one below:



You should see a line at the end that reports the presence of GDAL.

If instead you get an error message that looks like this then don't panic. Just use windows to open up the C:\OSGeo4W64\bin\gdalplugins folder (or your local equivalent) and delete the file called 'gdal_ECW_JP2ECW.dll'. That should fix it.



Important note: some users have had problems with OSGeo4W installing (and then insisting on using) an older version of Java that is invisible to the official Java uninstaller tool because it's in a nonstandard location. After installing, use windows explorer to check for the file 'java.exe' in your C:\OSGeo4W\bin directory. If that file exists please delete it or rename it to something else.

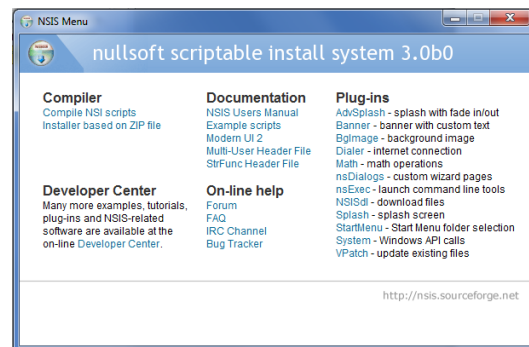
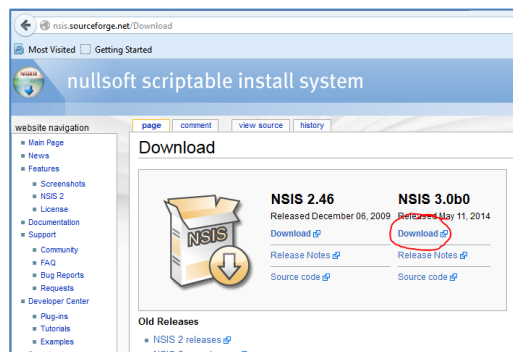
Although the OSGeo4W Shell looks like the normal Windows command prompt, it in fact has several special environment settings. It is the 'workbench' that we shall be using to run all the rest of the tools.

For now we don't need to run any commands so just click into the window and type **exit** then hit return.

d) install the NSIS scriptable installer

NSIS is an excellent, free, tool for creating Windows installers. The map creation process automatically uses NSIS to create an installer that sets up the map on your PC so that it's accessible to Garmin *Basecamp* even when your GPS isn't connected. You can also use the installer to set up the map on other people's PCs even if they don't have a Garmin (they just have to download Basecamp).

Go to the NSIS download page at <http://nsis.sourceforge.net/Download>. Click the *Download* link for the latest version (see below left) then run the installer accepting all the default values. Make a note of the installation location (probably C:\Program Files (x86)\NSIS or C:\Program Files\NSIS) as you will need this information later.



To check the installation, use Windows explorer to open the folder at the installation location. Click on *NSIS.exe* – this should open a window like the one on the right, which you can then simply close.

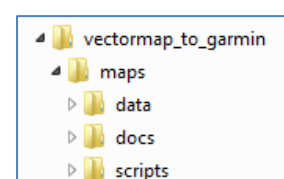
e) install the OpenStreetMap tools and vmd2garmin

The OpenStreetMap project provides the core tools to process and convert the Ordnance Survey map data into a form suitable for a Garmin GPS. This process is automated and controlled by my 'vmd2garmin' script supported by several other files that control the styling of the map.

To make things simple I have bundled all the necessary files up into a single archive that also creates a directory structure that organises the different types of map data into a consistent structure.

Browse to <https://sourceforge.net/projects/vectormap2garmin/files/> and open the folder for the latest version. Then download the file *vectormap_to_garmin.zip*, extract it and copy the 'maps' folder to a convenient location such as C:\Users\<username>\Documents .

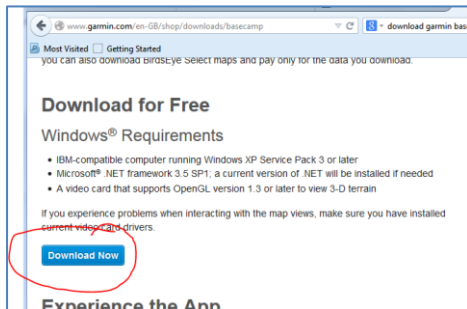
You should now have a *vectormap_to_garmin* folder and a number of subfolders that contain the various scripts and configuration files and also provide the location in which to store the source data for the map.



f) install Garmin Basecamp

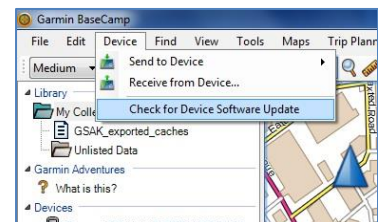
Garmin Basecamp is a very useful Windows application to view and interact with maps installed on your GPS device or on your PC. It also loads software updates for your device and is able to upload maps to certain Garmin GPS devices that don't have a USB interface.

You can even use Basecamp to use the maps we will create if you don't have a GPS device at all.



It is not necessary to re-install Basecamp if you already have it but if you don't then browse to <http://www.garmin.com/en-GB/shop/downloads/basecamp> and scroll down to the 'Download for Free' section as shown. Click the download button and follow the instructions.

To check the installation, simply launch Basecamp from its item in the start menu. If you have a GPS device then you should connect it and then make sure it is running the most up-to-date software by using the option under the 'device' menu.



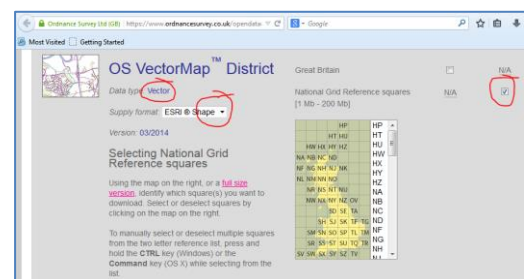
Step 2: download the Ordnance Survey map data

The OS Vectormap District data is organised into 100km squares, each identified by a two letter code. I recommend that you start with just one square and fetch more later when you've checked that you can successfully build the map. The download process is very straightforward.

Go to the overview map at <http://www.ordnancesurvey.co.uk/docs/maps/national-grid-map.pdf>, decide which square(s) you want and then go to the OS open data download page at <https://www.ordnancesurvey.co.uk/opendatadownload/products.html>.

Scroll down until you see the OS Vectormap District section for the *vector* data type and that the Supply Format is *ESRI shape* (see right).

Tick the box for *National Grid Reference Squares* then while holding down the Ctrl key click on the square(s) that you want.

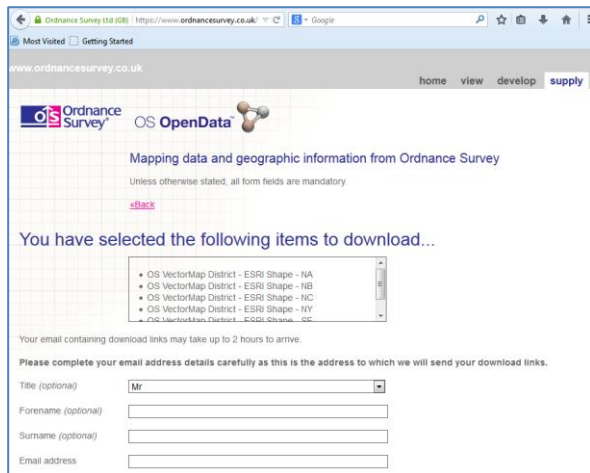


Now scroll down to the bottom of the page and click on the large blue *next* button. This takes you to a validation page that confirms what you want to download and where you have to complete some information about yourself.

You don't have to give your name but be sure to provide a valid email address because this is used to send you a one-time link to download the data.

For company name enter 'personal'.

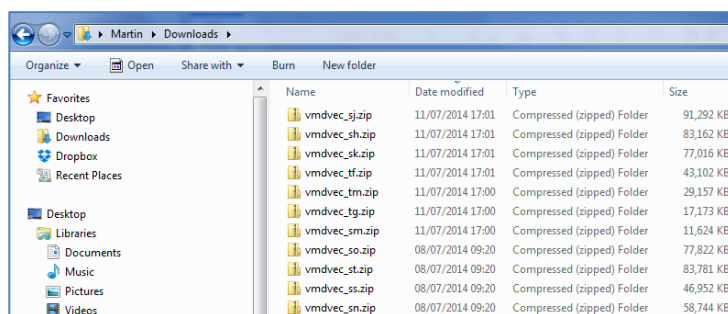
Scroll down to the bottom of the page and complete the *Captcha* control in the red box. This is just there to prevent automated abuse of the site. It says you should type the two words shown but I have only ever seen it display a three digit number.



The screenshot shows the Ordnance Survey OpenData website. It displays a list of selected items for download: OS VectorMap District - ESRI Shape - NA, OS VectorMap District - ESRI Shape - NB, OS VectorMap District - ESRI Shape - NC, OS VectorMap District - ESRI Shape - NY, and OS VectorMap District - ESRI Shape - SE. Below the list, there is a form to enter contact details. The form includes fields for Title (optional), Forename (optional), Surname (optional), and Email address. The Title field is currently set to 'Mr'.

Now scroll down to the bottom of the page and click *continue*. This should almost immediately send an email to the address you specified - although the site does say it could take up to 2hrs.

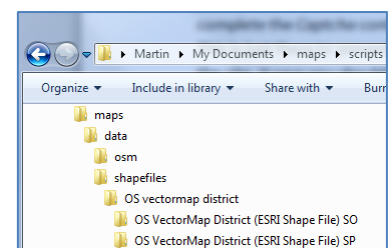
When the email arrives it will contain a link for each of the 100km squares that you requested. The links are valid for a period of three days.



Name	Date modified	Type	Size
vmdvec_sj.zip	11/07/2014 17:01	Compressed (zipped) Folder	91,292 KB
vmdvec_sh.zip	11/07/2014 17:01	Compressed (zipped) Folder	83,162 KB
vmdvec_sk.zip	11/07/2014 17:01	Compressed (zipped) Folder	77,016 KB
vmdvec_tf.zip	11/07/2014 17:01	Compressed (zipped) Folder	43,102 KB
vmdvec_tm.zip	11/07/2014 17:00	Compressed (zipped) Folder	29,157 KB
vmdvec_tg.zip	11/07/2014 17:00	Compressed (zipped) Folder	17,173 KB
vmdvec_sm.zip	11/07/2014 17:00	Compressed (zipped) Folder	11,624 KB
vmdvec_so.zip	08/07/2014 09:20	Compressed (zipped) Folder	77,822 KB
vmdvec_st.zip	08/07/2014 09:20	Compressed (zipped) Folder	83,781 KB
vmdvec_ss.zip	08/07/2014 09:20	Compressed (zipped) Folder	46,952 KB
vmdvec_sn.zip	08/07/2014 09:20	Compressed (zipped) Folder	58,744 KB

Go ahead and download the squares. Each one will arrive in the form of a zip file of around 50 to 100MB so you might want to do this somewhere with a good Internet connection.

Extract each zip file into the *OS vectormap district* folder in the *shapefile* area of the data directory that you created in Step 1(d). This should create a separate subfolder for each square, as shown.



Step 3: download the OpenStreetMap footpath data

As discussed above the Ordnance Survey Vectormap District series doesn't include footpath data so we need to fetch this from somewhere else. Several country councils now publish an online version of their rights of way maps but unfortunately this is not yet universal. The best solution for the time being is to extract this information from the OpenStreetMap project, which usually incorporates all the data that councils have published - although it's not yet quite as good as the full data shown on OS maps.

Fortunately the process to extract and download OpenStreetMap data is pretty straightforward.

First of all you need to know the extent of your map in degrees of latitude and longitude. This enables us to extract/download the footpath data for only the area you need, which is much quicker and more efficient than downloading the data for the whole of the UK.

Square	South	West	North	East
HP	60.68	-2.00	61.57	-0.11
HT	59.77	-3.79	60.69	-2.00
HU	59.78	-2.00	60.68	-0.16
HW	58.78	-7.20	59.74	-5.55
HX	58.84	-5.47	59.78	-3.78
HY	58.87	-3.74	59.79	-2.00
HZ	58.88	-2.00	59.78	-0.21
NA	57.80	-8.74	58.79	-7.19
NB	57.88	-7.07	58.85	-5.46
NC	57.94	-5.39	58.88	-3.73
ND	57.97	-3.70	58.89	-2.00
NF	56.91	-8.58	57.89	-7.06
NG	56.99	-6.94	57.95	-5.38
NH	57.04	-5.30	57.98	-3.69
NJ	57.08	-3.66	57.99	-2.00
NK	57.09	-2.01	57.98	-0.30
NL	56.02	-8.43	57.00	-6.93
NM	56.09	-6.83	57.05	-5.29
NN	56.15	-5.23	57.09	-3.65
NO	56.18	-3.62	57.10	-2.00
NR	55.20	-6.72	56.16	-5.22
NS	55.25	-5.15	56.19	-3.61
NT	55.28	-3.58	56.20	-2.00
NU	55.29	-2.01	56.19	-0.38
NW	54.30	-6.62	55.26	-5.14
NX	54.35	-5.08	55.29	-3.57
NY	54.38	-3.55	55.30	-2.00
NZ	54.39	-2.01	55.29	-0.42
OV	54.38	-0.47	55.26	1.15
SC	53.45	-5.02	54.39	-3.54
SD	53.48	-3.51	54.40	-2.00
SE	53.49	-2.01	54.39	-0.46
SH	52.56	-4.96	53.49	-3.50
SJ	52.58	-3.48	53.50	-2.00
SK	52.59	-2.01	53.49	-0.49
SM	51.61	-6.34	52.57	-4.95
SN	51.66	-4.90	52.59	-3.47
SO	51.68	-3.45	52.60	-2.00
SP	51.69	-2.01	52.59	-0.52
SR	50.72	-6.26	51.67	-4.89
SS	50.76	-4.84	51.69	-3.44
ST	50.79	-3.43	51.70	-2.00
SU	50.79	-2.01	51.69	-0.55
SV	49.76	-7.56	50.73	-6.25
SW	49.82	-6.18	50.77	-4.83
SX	49.86	-4.79	50.80	-3.42
SY	49.89	-3.40	50.80	-2.00
SZ	49.89	-2.00	50.80	-0.58
TA	53.48	-0.50	54.36	1.08
TF	52.58	-0.53	53.46	1.02
TG	52.56	0.94	53.42	2.52
TL	51.68	-0.56	52.57	0.95
TM	51.66	0.89	52.52	2.43
TQ	50.79	-0.59	51.67	0.90
TR	50.76	0.83	51.62	2.34
TV	49.89	-0.61	50.77	0.84

You can find the overall min and max coordinates of your map very easily by using the table on the left³.

Simply look up the values for each 100 km square in your map in turn; and note down the overall lowest (most negative) South and West coordinates, and the overall highest (least negative) North and East ones.

So for example, if your map is made up of squares SK, TF, SP and TL (highlighted) then its overall extent is as follows:

South = 51.68
West = -2.01
North = 53.49
East = 1.02

Note that the 100 km squares are not 'square' in terms of latitude and longitude. This is because of the curvature of the Earth.

If you have problems with the above or you want to download the footpath data for the whole UK, then in the following stages you can use the values S=49.76, W=-8.74, N=61.57 and E=2.52 but be aware that this will generate quite a large download (approx. 1.6 gigabyte).

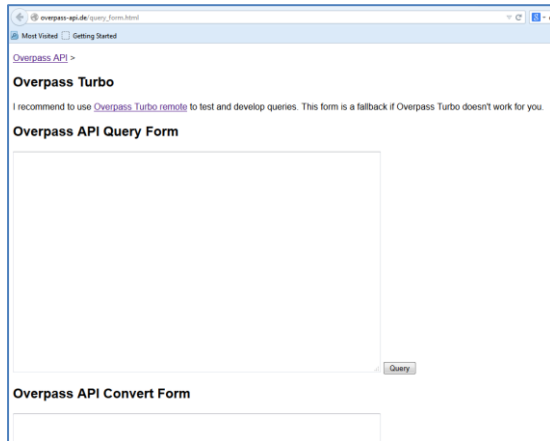
We will now go ahead and download and extract the footpath data for the area of your map.

You can alternately of course create separate maps for different 100km squares if you prefer, as long as you give them each a different family-id (*see later*). In that case you would probably download a separate footpath file for each square.

³ a spreadsheet copy of this chart is included at *maps\docs\OS_grid_extents.xlsx* (unhide all columns to see the workings)

To extract and download the footpath data we will use the Overpass API Query Form. This is probably not the most elegant approach but I've found it to be the most reliable.

Browse to http://overpass-api.de/query_form.html. This will open up a page that looks like the image below:



Now copy and paste the XML query⁴ below into the window labelled *Overpass API Query Form*. You will have to change the coordinates in each of the *<bbox-query>* parts (shown in red) to be the extent of your map in degrees of latitude and longitude.

```
<osm-script timeout="1000" element-limit="1500000000">
  <union>
    <query type="way">
      <has-kv k="highway" regv="bridleway|cycleway|footway|path|track"/>
      <bbox-query s="51.69" w="-2.01" n="52.59" e="-0.52"/>
    </query>
    <query type="way">
      <has-kv k="designation" regv="restricted_byway|public_byway|public_bridleway|public_footpath|byway_open_to_all_traffic"/>
      <bbox-query s="51.69" w="-2.01" n="52.59" e="-0.52"/>
    </query>
    <query type="way">
      <has-kv k="horse"/>
      <has-kv k="horse" modv="not" v="no"/>
      <bbox-query s="51.69" w="-2.01" n="52.59" e="-0.52"/>
    </query>
    <query type="way">
      <has-kv k="bicycle"/>
      <has-kv k="bicycle" modv="not" v="no"/>
      <bbox-query s="51.69" w="-2.01" n="52.59" e="-0.52"/>
    </query>
    <query type="way">
      <has-kv k="cycle"/>
      <has-kv k="cycle" modv="not" v="no"/>
      <bbox-query s="51.69" w="-2.01" n="52.59" e="-0.52"/>
    </query>
    <query type="way">
      <has-kv k="foot"/>
      <has-kv k="foot" modv="not" v="no"/>
      <bbox-query s="51.69" w="-2.01" n="52.59" e="-0.52"/>
    </query>
  </union>
  <print mode="meta" geometry="skeleton" order="quadtile"/>
  <recurse type="down"/>
  <print mode="meta" geometry="skeleton" order="quadtile"/>
</osm-script>
```

Finally click the 'Query' button to the right of the window. Your browser will wait for a few tens of seconds while the OpenStreetMap server prepares the data. It will probably then pop up a window asking you what you want to do with the file, in which case select 'save'. Make a note of the filename (it will probably be *'interpreter'*). At this point the download will begin.

⁴ the complexity of this query reflects the many different ways in which rights of way are labelled in OpenStreetMap data - to understand the commands used here, see http://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL

Once the download has completed - which may take some time - locate it in your download folder and move it to the *OSM* folder of the data directory that you created in Step 1(d). You should also rename it to something more sensible like *'footpaths.osm'*.

Note: if the OSM servers are exceptionally busy or if your Internet connection is slow then the above command may timeout. If this happens then it is probably worth just trying again an hour or so later; but if this fails too you could try slightly increasing the timeout value in the command above.

Optional but recommended: OpenStreetMap bridge data

The Vectormap District series doesn't include information about bridges – but you can include them in your map by extracting them from the OSM dataset (see also example 2 in the appendix).

The process is the same as above but the query is much simpler because the labelling of bridges is much more consistent than rights of way.

Use the query below to extract all the ways tagged as bridges within the area of your map. Of course you will need to adjust the coordinates of the bounding box as before.

```
<osm-script timeout="1000" element-limit="1500000000">
  <query type="way">
    <has-kv k="bridge"/>
    <bbox-query into="_" s="51.69" w="-2.01" n="52.59" e="-0.52"/>
  </query>
  <print mode="meta" geometry="skeleton" order="quadtile"/>
  <recurse type="down"/>
  <print mode="meta" geometry="skeleton" order="quadtile"/>
</osm-script>
```

*Download and save the data as above, rename it to something appropriate like *bridges.osm*, and add it to the list of *extra_osm_files* in *settings.txt* (see next section).*

Step 4: configure the settings file

Note that the content of the settings file has changed slightly since the previous release. If you are migrating from a previous version you are strongly recommended to start again with the new settings file included in the V1.3 distribution.

The main script that we will use to drive the map conversion process is *vmd2garmin.py* which can be found in the *maps\scripts\vmd2garmin* folder. This script calls all the other tools in order to convert and style the map.

The script reads its configuration from a settings file that tells it where to find the other tools and what actions you want it to perform. An example of the settings file is provided as *settings.txt* in the same directory as the script, but you will need to edit this to meet your own requirements. This is a straightforward process and the file has comments explaining what the different settings do.

Open the *settings.txt* file with a text editor and have a look. The layout and format follow the normal conventions of a Windows settings file. Lines beginning with *'#'* are ignored.

By all means look at all of the settings and comments in the file but under normal circumstances you will only have to change the ones listed below, all of which appear near the top. Overwrite the example value for each setting to match your own configuration and remember to save the file afterwards.

a) **base_dir**

Where to find the 'maps' folder that contains all the other tools and files. Enter the name of the directory where you put the 'maps' folder.

b) **actions**

The processing steps that the script should perform. More information on what the different steps do is given in the next step, but if you are building a new map you probably want to leave this at the default setting of *sort, convert, merge, split, compile, install*.

If you are just re-styling the map you might only need to specify *compile* and *install* depending on precisely what you are doing.

Note that implementation of the 'convert' action has changed from a python based tool to one that uses Java. This is to permit the project to run on smaller memory systems. The new implementation makes it unnecessary to use the old 'scan' action⁵.

c) **grid_squares**

The list of 100km squares to be included in the map. Obviously you need to have downloaded and unpacked all of these beforehand as described in step 2.

As shown in the example value, the list must be comma separated and may extend over more than one line.

d) **extra_osm_files**

The list of additional OSM formatted data files to be included in the map. This might be just the footpath data that you downloaded in step 3 or it might be a coma-separated list of files. Include the filename of the optional bridge data here, if you downloaded it above.

Note that the 'sort' action overwrites each of these files with a re-ordered version sorted by entity type and ID. Therefore the 'sort' action only needs to be executed once unless new data is added.

e) **map_name**

The name of the map as it will appear on the PC and the GPS unit. For optimum compatibility with different devices you should keep this name short and not include any spaces.

⁵ If you have plenty of memory (10GB or more) you may find it slightly quicker to use the old python based tool. This also requires the 'scan' action so use something like *'actions = sort, scan, convert_python, merge ...'*

The script also uses this name to create a working directory in `maps\data` to hold the binary Garmin `.img` files created by the map compilation process, the final `gmapsupp.img` file and the windows installer.

f) **family_id**

The family ID is a four digit number that is used to identify a Garmin map on a device or in Basecamp. If you have several maps on a device then they must all have different IDs or they won't all show up⁶.

If you are building several maps – e.g. I know at least one person who prefers to keep their 100km squares as separate maps – then give them each a different family ID number. I recommend starting at 6111 and counting upwards.

The script automatically patches the TYP file used to style the map, with the right family ID.

g) **java_options**

You can include options here that will be passed to Java such as `-Xmx` or `-Xms` to control its memory usage.

I recommend that it is no longer necessary to use these options unless you really want to.

h) **nsis**

The location of the Nullsoft Scriptable Installer tool (NSIS), as installed in step 1. Enter the full path and filename of the `makensis.exe` file.

Note that if for some reason you do not wish to build an installer for the PC then you can leave this value blank. This will cause an error at the end of the `compile` stage but the `gmapsupp.img` file will nonetheless be created in the working directory.

Having edited and saved the `settings.txt` file you should be ready to start the conversion process.

Step 5: build the map

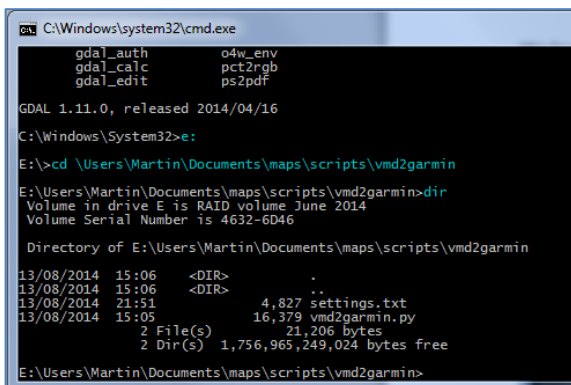
The process of building the map is controlled by the `vmd2garmin` script. During a full build the script goes through the following processes:

- *sort* the OSM imported data by entity type and ID
- *scan* the OSM imported data to find the highest entity ID
- *convert* the layers of each included 100km Vectormap square into OSM format
- *merge* all the converted Vectormap squares and OSM imported data into one big OSM map
- *split* the big OSM map into a set of rectangular tiles according to the max number of nodes
- *compile* each of the tiles to produce a Garmin `.img` file and an overall `gmapsupp.img` file
- *install* the tiles into the PC's registry so that Basecamp will recognise it

⁶ Don't use the Product ID. It won't do what you expect due to limitations with the installer.

The above process does not install the map onto the GPS device itself – we'll do that in the next step.

We have to run *vmd2garmin* from within OSGeo4W in order to get all the necessary environment



```
C:\Windows\system32\cmd.exe
gdal_auth      o4w_env
gdal_calc      pct2rgb
gdal_edit      ps2pdf

GDAL 1.11.0, released 2014/04/16
C:\Windows\System32>e:
E:\>cd \Users\Martin\Documents\maps\scripts\vmd2garmin
E:\Users\Martin\Documents\maps\scripts\vmd2garmin>dir
Volume in drive E is RAID volume June 2014
Volume Serial Number is 4632-6D46

Directory of E:\Users\Martin\Documents\maps\scripts\vmd2garmin

13/08/2014  15:06    <DIR>        .
13/08/2014  15:06    <DIR>        ..
13/08/2014  21:51                4,827 settings.txt
13/08/2014  15:05            16,379 vmd2garmin.py
               2 File(s)              21,206 bytes
               2 Dir(s)  1,756,965,249,024 bytes free

E:\Users\Martin\Documents\maps\scripts\vmd2garmin>
```

settings. So find the OSGeo4W folder in the start menu and click on *OSGeo4W Shell*. This will open a command window. Within this command window use the '**cd**' (change directory) command to navigate to your *maps\scripts\vmd2garmin* directory. You can use the '**dir**' command to check that you are in the right place.

Change directory to the location of your vmd2garmin script - mine lives on the E: drive.

The conversion process can take quite a while - a four core 3.5 GHz PC with 16GB of physical memory and a fast disk can compile a complex 100km square in about fifteen minutes, but a machine with less memory and/or a slower disk might take two to or three hours because of the additional time spent swapping to and from disk. If you are compiling a large map it's a good idea to temporarily adjust your power settings to stop your machine going into sleep mode during the build process. The way you do this varies between different versions of Windows but is generally something like *start->control panel->power options*.

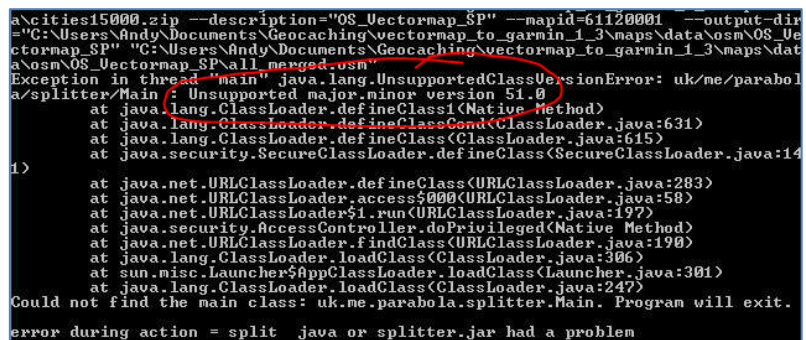
You can launch the script with the following command: '**python vmd2garmin.py settings.txt**'.

The script begins by making a few basic checks of the settings in the configuration file and then executes the specified actions. If it hits a problem it will print an error message and then stop – if that happens then carefully read the information shown and try to figure out what went wrong.

Don't be too surprised if your machine gets pretty unresponsive during the conversion process unless it really does have plenty of memory. If it seems to have 'died' then it is probably best just working very hard and is best left to get on with it for an hour or two – especially if the disk access light appears to be solidly on. Do not reboot it unless you really have to.

The script displays various progress information as it goes through the different stages of the conversion process. Assuming all goes well it will finally finish or arrive at the *install* stage – if the latter it will stop and ask you to hit 'enter' when you are ready to proceed (this is to prevent the installer timing out).

If the script stops with an error message from Java, check for a line that says '*Unsupported major.minor version number*'. If you see this then it means that the script is somehow using an outdated version of Java. See step 1(c) for a fix.



```
a:\cities15000.zip --description="OS_Vectormap SP" --mapid=61120001 --output-dir
="C:\Users\Andy\Documents\Geocaching\vectormap_to_garmin_1_3\maps\data\osm\OS_Ve
ctormap_SP" "C:\Users\Andy\Documents\Geocaching\vectormap_to_garmin_1_3\maps\dat
a\osm\OS_Vectormap_SP\all_merged.osm"
Exception in thread "main" java.lang.UnsupportedClassVersionError: uk/me/parabola
/splitter/Main : Unsupported major.minor version 51.0
    at java.lang.ClassLoader.defineClass1(Native Method)
    at java.lang.ClassLoader.defineClassCond(ClassLoader.java:631)
    at java.lang.ClassLoader.defineClass(ClassLoader.java:615)
    at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:14
1)
    at java.net.URLClassLoader.defineClass(URLClassLoader.java:283)
    at java.net.URLClassLoader.access$000(URLClassLoader.java:58)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:197)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:306)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:247)
Could not find the main class: uk.me.parabola.splitter.Main. Program will exit.
error during action = split java or splitter.jar had a problem
```

After the *compile* stage has completed the finished map will have been created in the working directory in the *maps\data\<map name>* folder. This should contain several different types of files as described below.

Documents library		
OS_Vectormap		
Name	Date modified	Type
63240069.img	11/07/2014 22:59	Disc Image File
63240070.img	11/07/2014 22:59	Disc Image File
63240071.img	11/07/2014 22:59	Disc Image File
63240072.img	11/07/2014 22:59	Disc Image File
63240073.img	11/07/2014 23:00	Disc Image File
gmapsupp.img	11/07/2014 23:00	Disc Image File
OS_Vectormap.exe	11/07/2014 23:04	Application
OS_Vectormap.img	11/07/2014 23:00	Disc Image File
OS_Vectormap.nsi	11/07/2014 23:00	NSIS Script File
OS_Vectormap.tdb	11/07/2014 23:00	TDB File
OS_Vectormap_license.txt	11/07/2014 23:00	Text Document
ovm_63240001.img	11/07/2014 22:39	Disc Image File
ovm_63240002.img	11/07/2014 22:40	Disc Image File

1) Several *.img* files with numeric filenames – these are the individual tiles of the map.

2) A larger *gmapsupp.img* file - this is the final completed map that you will transfer to the GPS device.

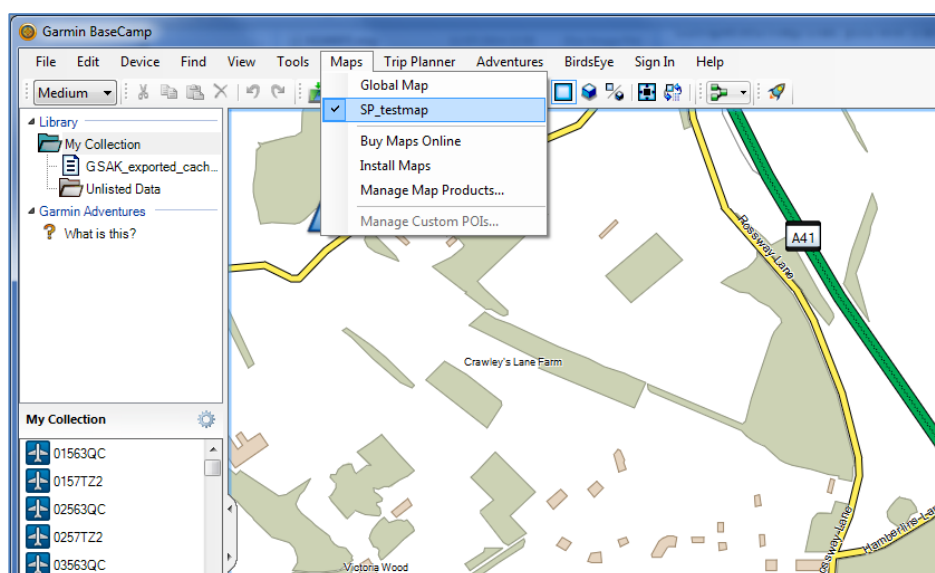
3) An executable *.exe* file – this is the installer that sets up the map on a PC to view with Basecamp.

4) Some other files including a license file.

The *install* stage of the script simply calls the executable – you can equally well run it manually if you prefer. The executable contains a compressed version of the whole map so if you want to install the finished map on another PC you just need to transfer across and execute this file.

Similarly the *gmapsupp.img* file also contains a compressed version of the whole map and is the only file that you need to transfer across to a Garmin GPS device.

When you have run the installer you should be able to run *Basecamp* and see your map (without the GPS connected). If it doesn't show up straight away then click on the 'maps' tab and select it from the drop-down list.



Note that if you make changes to the styling of an existing map then they may not show up straight away because *basecamp* caches the tiles internally. With modern versions of Basecamp you can force a refresh of the tile cache by hitting *ctrl-G* twice.

Step 6: install the map on your GPS

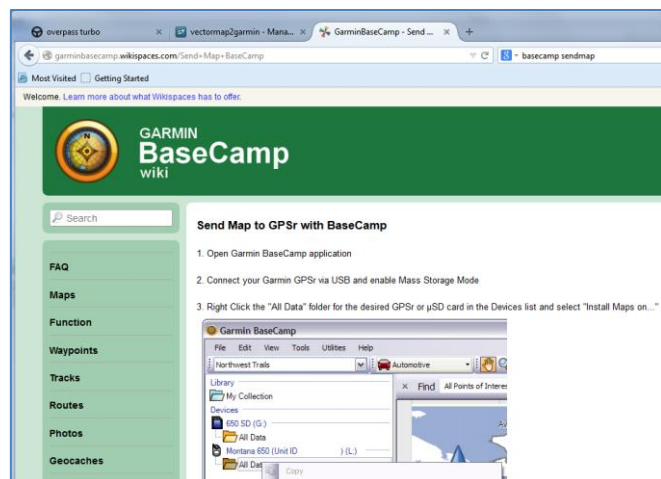
If you have a GPS with a removable memory card then it is best to install the map onto that rather than the internal memory of the device, so that you can easily remove it if there is a problem.

If you have an existing memory card with the Ordnance Survey 1:50k maps that came with the device, then there may be enough spare space to fit the map on there. Be very careful not to overwrite any of the existing files however.

To install the map simply connect your GPS device via the USB cable and copy the *gmapsupp.img* file to the GARMIN folder of the memory card (or the device if appropriate). You can change the name of the file if there's a different gmapsupp.img already there.

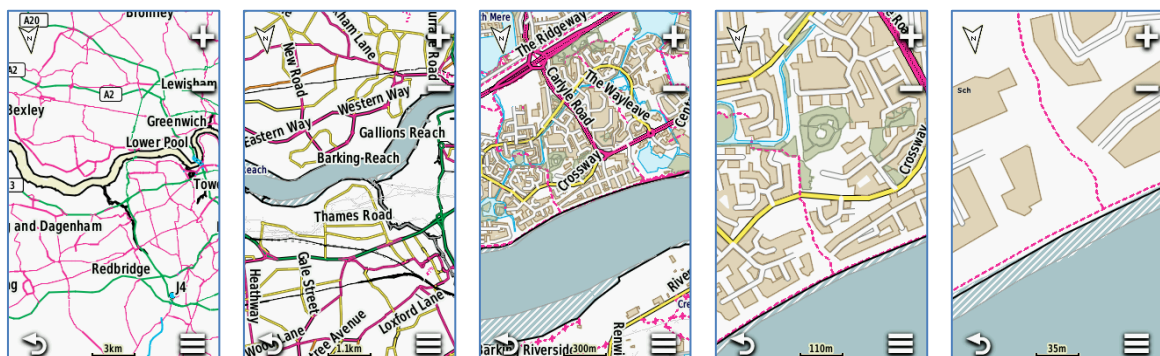
If you have problems getting your device to connect via USB then you might need to install or update to the latest Garmin USB drivers: http://www8.garmin.com/support/download_details.jsp?id=591

If you have a version of GPS unit that doesn't support USB then you might need to transfer the map across using Basecamp, as described at <http://garminbasecamp.wikispaces.com/Send+Map+BaseCamp>.



After you have installed the map on your GPS unit, check its instructions for to see how to select it to be displayed. On my device (Oregon 600) it's under *setup->map->configure maps*. Initially you might want to de-select all the other maps on the device to make sure that the new one shows up.

Below are some screenshots of a converted map at different zoom levels on my device. This shows the big advantage of using vector mapping, which is that the detail stays sharp as you zoom in.



Appendix: customising the map

There are many ways to customise the map to make it look exactly the way you want and experimenting is a great way to learn about the tools. I'm still learning and exploring them myself.

The following section starts with a description of the different rules that configure the map, and then shows a simple and a more complex example of how to use them.

Overview of the different rules

When customising the map it is useful to have an outline understanding of the different rules that define what is included and how it looks. Then you can decide what to change.

Remember that the general process is to convert everything into OpenStreetmap (OSM) format first, and then to use the OSM tools to convert that into Garmin binary format.

1: shapefile to OSM conversion rules

NOTE: the new Java based 'convert' action uses a different set of conversion rules contained in `maps\data\styling\vmd-shp-to-osm-rules.txt` instead of what is described here. You can read about the new format [here](https://github.com/iandees/shp-to-osm/blob/master/README.txt): <https://github.com/iandees/shp-to-osm/blob/master/README.txt>

The first set of rules is contained in `maps\data\styling\vmd-attribute-translator.py`. These rules are used by the `ogr2osm` tool to pick out objects from the VMD shapefiles and convert them into similar OSM objects.

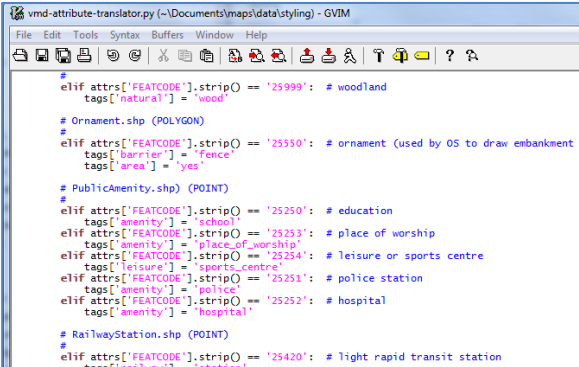
You will need to work with these rules if you want to incorporate additional data from shapefiles. This is the format in which most public bodies publish geographic information, such as the datasets released by *Natural England* (http://www.gis.naturalengland.org.uk/pubs/gis/GIS_register.asp).

Objects in shapefiles are identified by 'attributes' which can be examined with a tool like 'QGIS' that you should be included in your OSGeo4W folder. Run the 'QGIS Desktop' application then use *layer->add vector layer* to open one of the shapefiles in your VMD data, and *layer->open attribute table* to see the attributes of the individual objects. All objects in the VMD layers are identified with a numeric feature code ('FEATCODE') and some have additional attributes such as road names.

You can find the full list of attributes used in the Vectormap District shapefiles, in Chapter 4 of the Ordnance Survey user guide at <http://www.ordnancesurvey.co.uk/docs/user-guides/os-vectormap-district-user-guide.pdf>

The rules are written in python which is an easy language to learn – but you can probably achieve what you want by simply copying and modifying the existing code.

Here is an example of a few of the rules in the file:



```
vmd-attribute-translator.py (-\Documents\maps\data\styling) - GVIM
File Edit Tools Syntax Buffers Window Help
[Icons]
# elif attrs['FEATCODE'].strip() == '25999': # woodland
#     tags['natural'] = 'wood'
# Ornament.shp (POLYGON)
# elif attrs['FEATCODE'].strip() == '25550': # ornament (used by OS to draw embankment
#     tags['barrier'] = 'fence'
#     tags['area'] = 'yes'
# PublicAmenity.shp (POINT)
# elif attrs['FEATCODE'].strip() == '25250': # education
#     tags['amenity'] = 'school'
# elif attrs['FEATCODE'].strip() == '25253': # place of worship
#     tags['amenity'] = 'place_of_worship'
# elif attrs['FEATCODE'].strip() == '25254': # leisure or sports centre
#     tags['leisure'] = 'sports_centre'
# elif attrs['FEATCODE'].strip() == '25251': # police station
#     tags['amenity'] = 'police'
# elif attrs['FEATCODE'].strip() == '25252': # hospital
#     tags['amenity'] = 'hospital'
# RailwayStation.shp (POINT)
# elif attrs['FEATCODE'].strip() == '25420': # light rapid transit station
#     tags['railway'] = 'station'
```


2: OSM to Garmin conversion rules

The second set of rules is contained in the files in `maps\data\styling\mkgmap_styles\VMD_style`.

These rules are used by the `mkgmap` tool to convert the objects in the OSM map into POIs, lines and polygons in the different zoom levels of the Garmin binary map.

You will need to work with these rules if you want to add new features to the map from data that has been converted into OSM format from a shapefile, or data imported directly from the OSM project.

The rules are very flexible and powerful, and are organised into different files to make them easier to manage. The format is fully defined in the *mkgmap Style Manual*

(<http://www.mkgmap.org.uk/doc/pdf/style-manual.pdf>) but you can understand it fairly well by studying the rules and comments in the different files. There is also an active user community online.

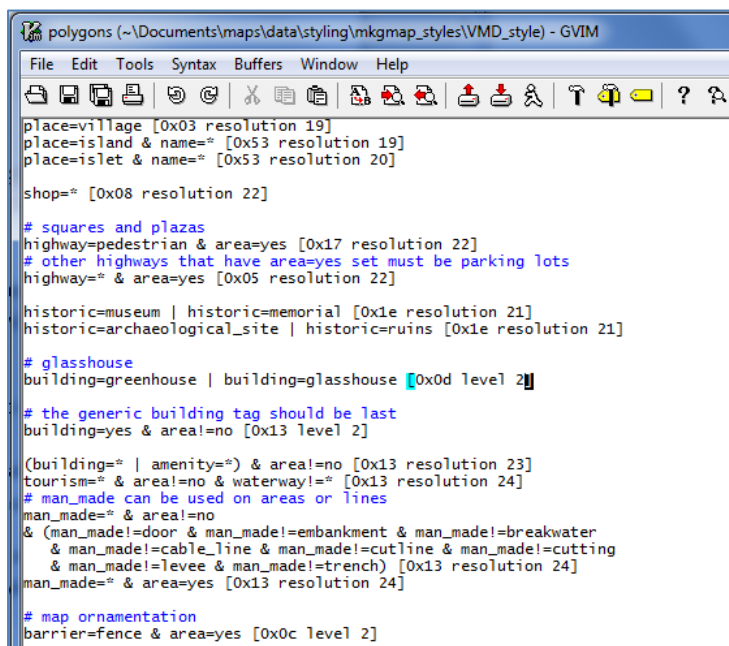
An important function of the rules is to manage how objects appear differently at different zoom levels of the Garmin map. You can read more about this topic by reading the style guide and other resources such as http://wiki.openstreetmap.org/wiki/Mkgmap/help/style_rules.

Note that I created the `VMD_style` by modifying the standard `mkgmap` style so some of the original rules that are still present are no longer used (it could probably do with a bit of a tidy up).

The second example below gives a little more detail about how to incorporate a rule to handle a new type of object.

When working with these rules you will often want to know what Garmin object types are available to customise to your needs. Some information on these, and my usage of them, is included in the files in the `maps\docs` directory (especially in *VMD type conversions.xlsx*).

Example of *mkgmap* rules (from the *polygons* file):

A screenshot of a GVIM editor window titled "polygons (~\Documents\maps\data\styling\mkgmap_styles\VMD_style) - GVIM". The window shows a list of mkgmap rules for polygons. The rules are organized into sections with comments. The visible rules include: place=village [0x03 resolution 19], place=island & name="" [0x53 resolution 19], place=islet & name="" [0x53 resolution 20], shop="" [0x08 resolution 22], # squares and plazas, highway=pedestrian & area=yes [0x17 resolution 22], # other highways that have area=yes set must be parking lots, highway="" & area=yes [0x05 resolution 22], historic=museum | historic=memorial [0x1e resolution 21], historic=archaeological_site | historic=ruins [0x1e resolution 21], # glasshouse, building=greenhouse | building=glasshouse [0x0d level 2], # the generic building tag should be last, building=yes & area=no [0x13 level 2], (building="" | amenity="" & area=no [0x13 resolution 23], tourism="" & area=no & waterway!=" [0x13 resolution 24], # man_made can be used on areas or lines, man_made="" & area=no, (man_made!=door & man_made!=embankment & man_made!=breakwater & man_made!=cable_line & man_made!=cutline & man_made!=cutting & man_made!=levee & man_made!=trench) [0x13 resolution 24], man_made="" & area=yes [0x13 resolution 24], # map ornamentation, barrier=fence & area=yes [0x0c level 2].

```
place=village [0x03 resolution 19]
place=island & name="" [0x53 resolution 19]
place=islet & name="" [0x53 resolution 20]

shop="" [0x08 resolution 22]

# squares and plazas
highway=pedestrian & area=yes [0x17 resolution 22]
# other highways that have area=yes set must be parking lots
highway="" & area=yes [0x05 resolution 22]

historic=museum | historic=memorial [0x1e resolution 21]
historic=archaeological_site | historic=ruins [0x1e resolution 21]

# glasshouse
building=greenhouse | building=glasshouse [0x0d level 2]

# the generic building tag should be last
building=yes & area=no [0x13 level 2]

(building="" | amenity="" & area=no [0x13 resolution 23]
tourism="" & area=no & waterway!=" [0x13 resolution 24]
# man_made can be used on areas or lines
man_made="" & area=no
(man_made!=door & man_made!=embankment & man_made!=breakwater
& man_made!=cable_line & man_made!=cutline & man_made!=cutting
& man_made!=levee & man_made!=trench) [0x13 resolution 24]
man_made="" & area=yes [0x13 resolution 24]

# map ornamentation
barrier=fence & area=yes [0x0c level 2]
```

3. Garmin rendering (TYP) rules

The final set of rules is the easiest to work with but has perhaps the biggest impact on how the map actually looks on the screen.

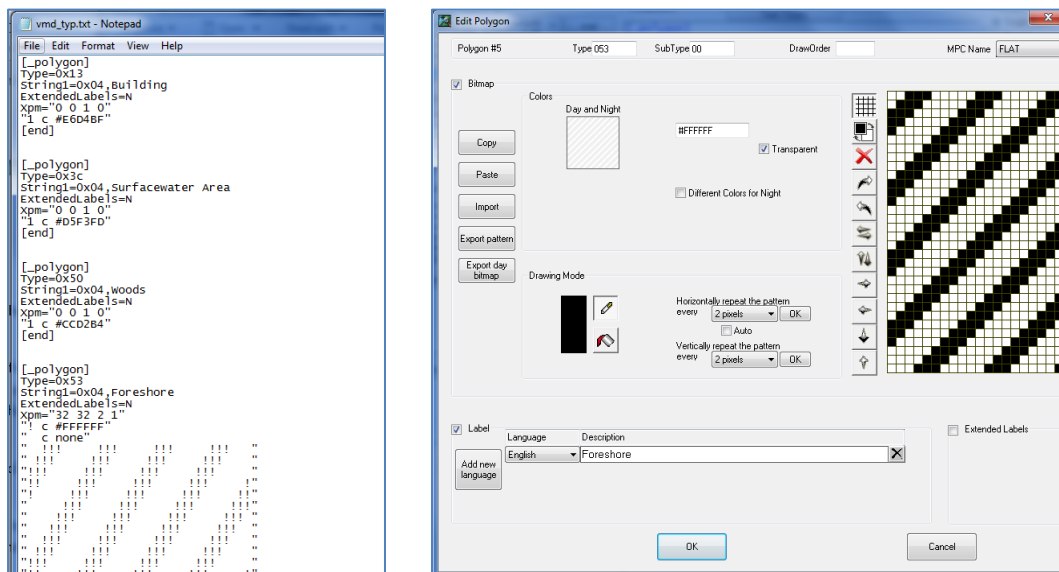
These rules tell the Garmin device or the *Basecamp* application how to render (draw) the different objects in the binary file. All Garmin devices have default ways of displaying these types, but we can override these with the definitions in this file. This allows us to control to a very fine degree how the map looks, and to display our own additional types of data.

The rules are contained in the file `maps\data\styling\vmd_typ.txt`. They are compiled and included in the final map by the *mkgmap* tool.

Whilst you *can* work with the `vmd_typ.txt` file manually, it is much easier to use a visual editing tool. I like the free one called 'TYPViewer' (available from <https://sites.google.com/site/sherco40/>) and I find that this works very well.

More information about the way to use TYP files can be found in the examples below and via online sources such as <http://pinns.co.uk/osm/styles.html> and in the *mkgmap* style guide.

Example of the rules in the `vmd_typ.txt` file and editing a rule in TYPViewer:



Note: TYPViewer is very good but it sometimes crashes when you do a lot of complex edits one after the other. It is wise to save your work periodically as you go.

Example 1: Change the way woodland areas appear

This is a rather synthetic example to show how you can work with the TYP file to quickly change the appearance of the final map.

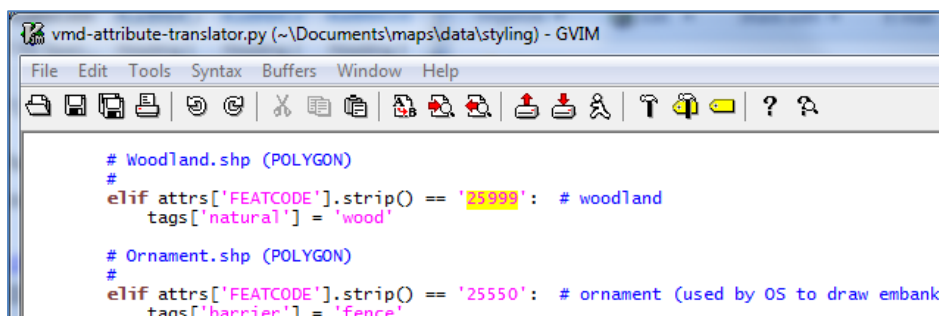
Let's imagine that we want woodlands to appear with tiled patterns of little trees instead of a plain green area. As we simply want to change the way an existing feature of the map is rendered, we can do this by simply working with the rules in *maps\data\styling\vmd_typ.txt*.

First of all we need to find out which object it is that we want to change. For this simple case we could simply look through the TYP file, but if necessary we could follow the chain all the way from the original VMD object through to the Garmin map:

- 1) From Chapter 5 of <http://www.ordnancesurvey.co.uk/docs/user-guides/os-vectormap-district-user-guide.pdf> we can see that woodlands are given feature code **FEATCODE 25999**:

	Hospital	25232
HeritageSite		24801
Woodland		25999
Ornament		25550
ElectricityTransmissionLine		25402

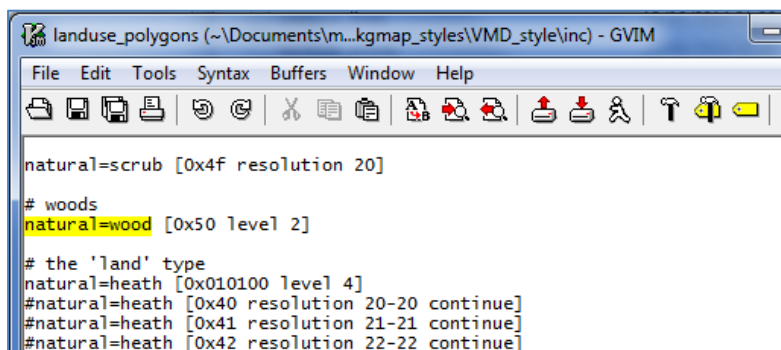
- 2) From *vmd-attribute-translator.py* we can see that FEATCODE 25999 is translated into an OSM tag with the key '**natural**' and the value '**wood**':



```
# Woodland.shp (POLYGON)
#
elif attrs['FEATCODE'].strip() == '25999': # woodland
    tags['natural'] = 'wood'

# Ornament.shp (POLYGON)
#
elif attrs['FEATCODE'].strip() == '25550': # ornament (used by OS to draw embankments)
    tags['barrier'] = 'fence'
```

- 3) From *maps\data\styling\mkgmap_styles\VMD_style\inc\landuse_polygons* we can see that OSM tag **natural=wood** is converted into Garmin object type **0x50** at zoom levels 2 and above:



```
natural=scrub [0x4f resolution 20]

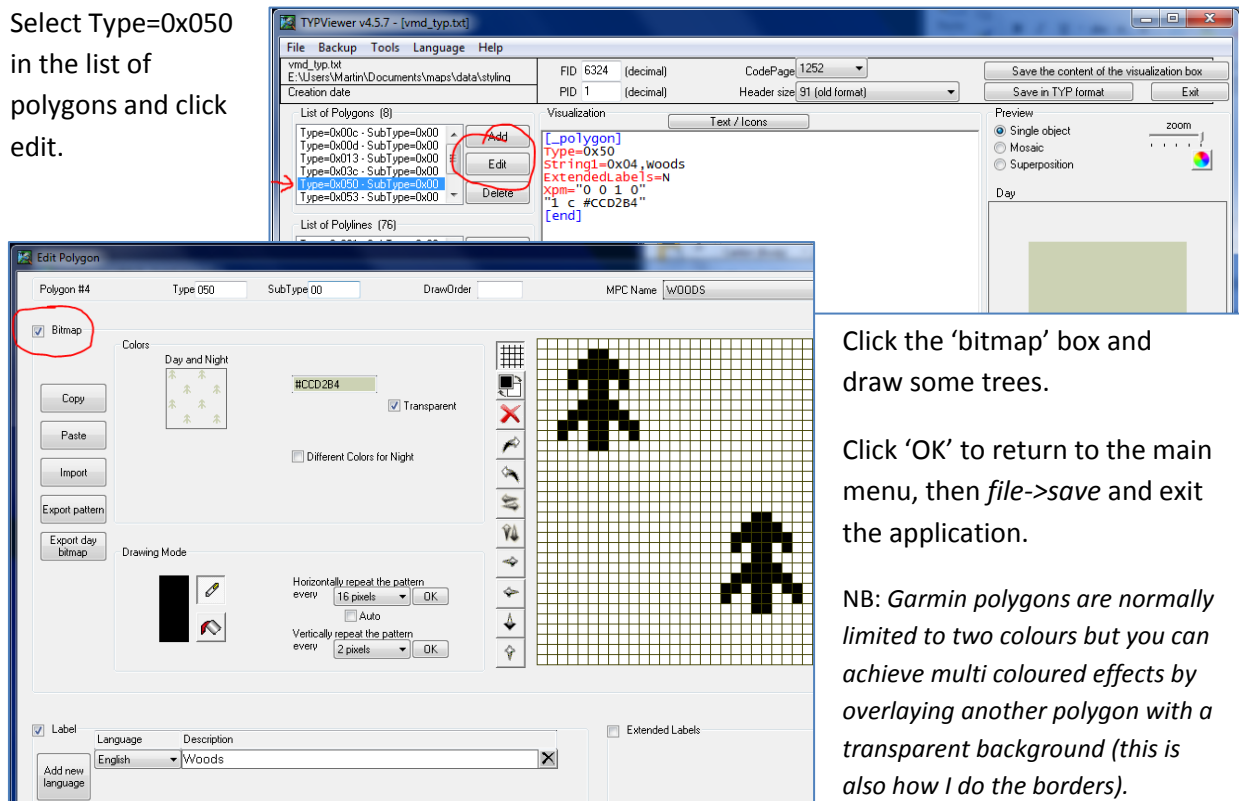
# woods
natural=wood [0x50 level 2]

# the 'land' type
natural=heath [0x010100 level 4]
#natural=heath [0x40 resolution 20-20 continue]
#natural=heath [0x41 resolution 21-21 continue]
#natural=heath [0x42 resolution 22-22 continue]
```

So we need to change the entry in the TYP file for polygon type 0x50.

Install TYPViewer from <https://sites.google.com/site/sherco40/> and use *file->open* to load *maps\data\styling\vmd_typ.txt*.

Select Type=0x050 in the list of polygons and click edit.



Click the 'bitmap' box and draw some trees.

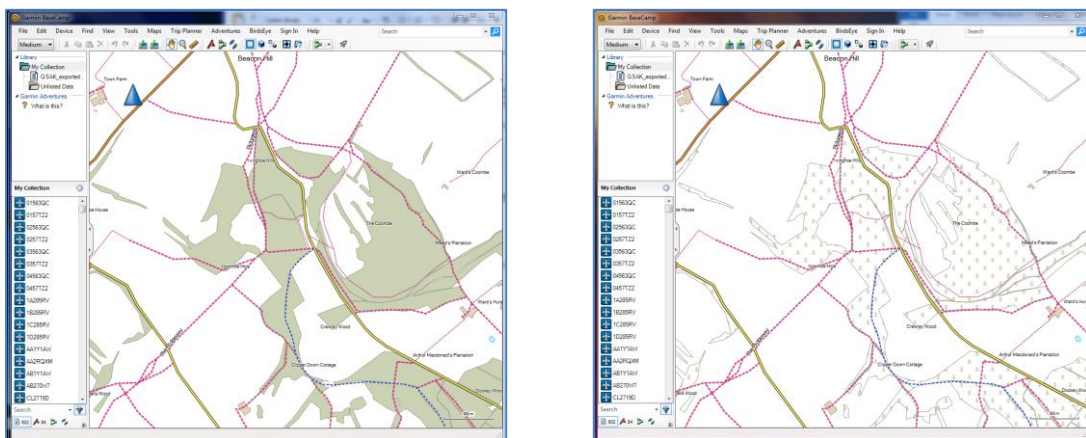
Click 'OK' to return to the main menu, then *file->save* and exit the application.

NB: Garmin polygons are normally limited to two colours but you can achieve multi coloured effects by overlaying another polygon with a transparent background (this is also how I do the borders).

You can now go ahead and rebuild the map using the normal process. As we haven't added or removed any map objects we can change the actions in the settings.txt file to just 'compile, install'.

*Note that if you reuse the same the **map_name** then the installer will prompt you to uninstall the existing version first. Alternately if you change the name then you will need to uninstall the previous version manually (using the uninstaller in the same directory as the installer) as Basecamp will only recognise one map with a given combination of Family ID and Product ID for the map. You can of course install several maps with different Family IDs (see main text).*

Here's how the original and modified versions of the map appear in basecamp. If you kept the same **map_name** you will need to hit *ctrl-G* twice to refresh the tile cache and make the changes show up.



Example 2: importing bridges from OSM data

This example is intended to show how to include and style additional features extracted from the OpenStreetMap (OSM) data set. In particular it describes how I went about adding bridges to the map.

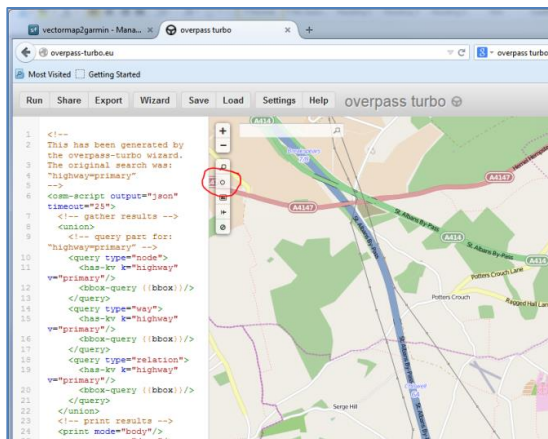
Step 1: extracting the OSM data

First of all I needed to understand how to find bridges in the OSM data. In the OSM world map every features has a unique ID number and a list of 'key = value' pairs called tags. For example the section of road highlighted in red below has ID = 15218060 and seven tags:



highway	primary
lanes	2
lit	yes
maxspeed	30mph
name	Watling Street
ref	A5183
source	OS_OpenData_Street View

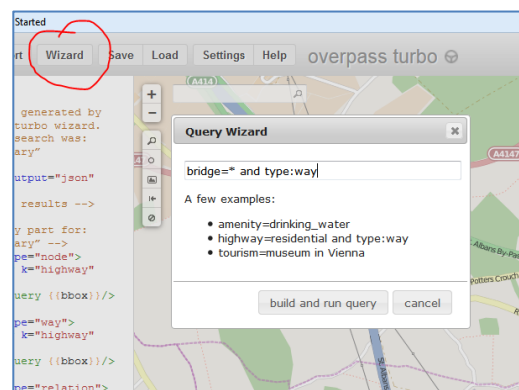
If you google for ‘osm bridge’ (or look at http://wiki.openstreetmap.org/wiki/Map_Features) you will find that the part of a route (known as a ‘way’) that crosses a bridge, is given a tag with the key ‘*bridge*’ and some value like ‘*yes*’, ‘*viaduct*’ etc.



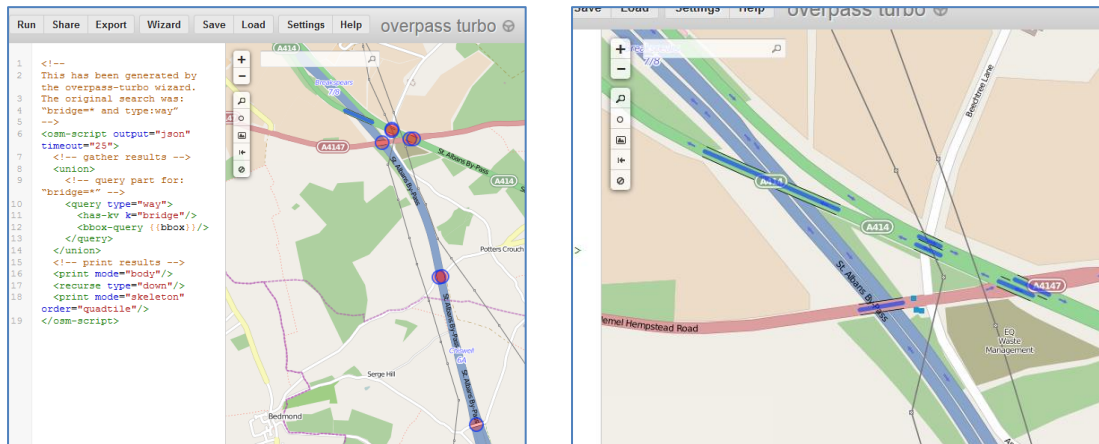
So I thought that I could probably extract the data I wanted with a query like “`bridge=* and type:way`”. Before running any kind of OSM query it is sensible to test it to make sure it does what you want. You can do this with the excellent online query tool called *overpass turbo*.

To do this browse to <http://overpass-turbo.eu/> and zoom to a medium sized area of the map in an area that you are familiar with. One way to do this is to click the *locate me* button (highlighted).

Clicking the *wizard* button pops up a window where you can type a query (see right). Type your query and click *build and run query*.



Clicking *build and run* converts the query into the overpass language and highlights all matching features on the map. By zooming in you can see the individual ways (right - highlighted in blue):



The query worked the way I wanted so I used the text in the left-hand pane to produce a modified version of the query used to extract the footpath data in Step 3 of the main instructions. You can do this by picking out the bits in the new 'query part' and grafting them into the original text.

The resulting query looks like the following, which is what you should use if you want to show the bridges on your version of the map. As before, you need to adjust the coordinates of the bounding box. Run this query as via the Overpass API Query Form at http://overpass-api.de/query_form.html and save the results in your *maps\data\osm* directory as something like '*bridges_SP.osm*'.

```
<osm-script timeout="1000" element-limit="1500000000">
  <query type="way">
    <has-kv k="bridge"/>
    <bbox-query into="_" s="51.69" w="-2.01" n="52.59" e="-0.52"/>
  </query>
  <print mode="meta" geometry="skeleton" order="quadtile"/>
  <recurse type="down"/>
  <print mode="meta" geometry="skeleton" order="quadtile"/>
</osm-script>
```

Step 2: converting the new OSM feature to a Garmin object type

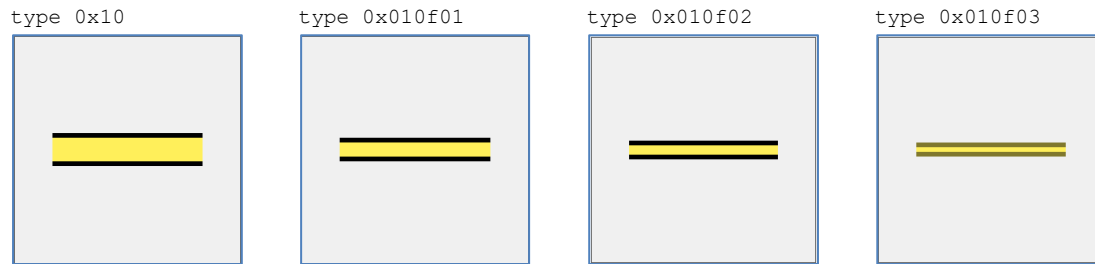
In order to include the bridges on the Garmin map I first needed to find an unused object type to convert them into; then to define a new rule in *maps\data\styling\mkgmap_styles\VMD_style*.

As described above, the files in that directory define rules that *mkgmap* follows in order to convert OSM features into Garmin object types.

My style uses rules that convert the same OSM feature into different Garmin object types depending on the zoom level of the map. For example the rules below, from the '*lines*' file, enable minor roads to be shown as thinner lines as you zoom out and then to disappear altogether at zoom levels 4 and above.

```
# minor road
highway=tertiary_link [0x10 level 0-0 continue]
highway=tertiary_link [0x010f01 level 1-1 continue]
highway=tertiary_link [0x010f02 level 2-2 continue]
highway=tertiary_link [0x010f03 level 3-3]
```


In the TYP *vmd_typ.txt* file each of the object types 0x10, 0x010f01, 0x010f02 and 0x010f03 are defined as a different width line.



Having checked that there were no existing rules to match bridges I needed to add a new one. As I wanted them to appear as lines, I added the new rule to the 'lines' file.

To create the rule I needed to decide which zoom level the bridges will appear at, and which Garmin object type they would be converted into. From the TYP file I could see that the first unused custom line type was 0x010f13.

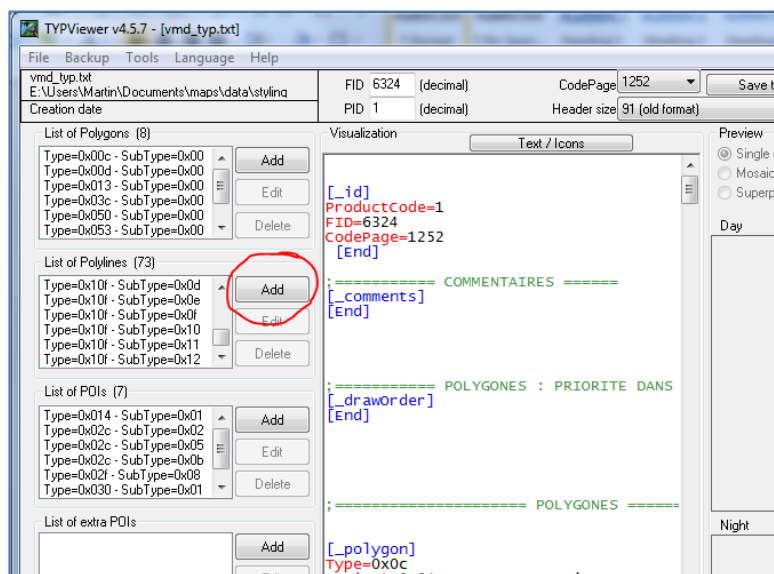
The following rule will show bridges at the highest two zoom levels only as object type 0x010f13.

```
# bridge (all types)
bridge=* [0x010f13 level 0-1]
```

In fact the actual rules I use for bridges are slightly different to this (you can find them near the top of the file) so that the processing 'falls through' to allow footpaths over bridges to be drawn in from the OSM data but then stops so that other types of ways that come from the Vectormap data are not drawn twice.

Step 3: defining how the object will appear

The next step was to define how this new Garmin object type would appear when it was drawn on the map. I did this by adding statements to the *vmd_typ.txt* file in the directory *maps\data\styling*.



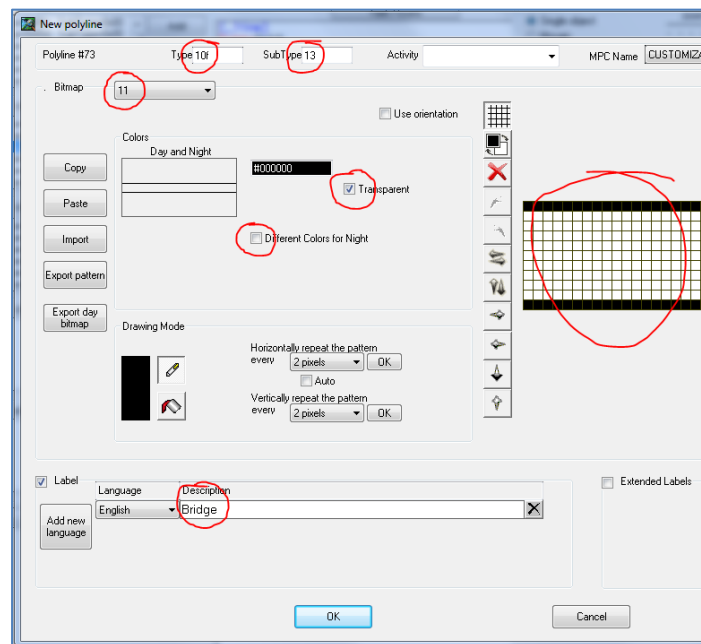
I opened the *vmd_typ.txt* file with TYPViewer (see above) and clicked the 'add' button to create a new polyline type.

This created a new window as shown below.

I used the settings as shown to set the styling for the bridge object as an 11pixel wide transparent bitmap with black borders so that it shows up as an extra border when it is drawn overlaying a section of road.

Note that in the TYP file, 0x010f13 is represented as type **10f** and sub-type **13**.

I clicked the OK button to add the new definition to the TYP file and then used *file->save* to update *vmd_typ.txt*.



Step 4: updating the settings file

I then modified the settings file in the *maps/scripts/vmd2garmin* directory to include the newly extracted bridge data in the **extra_osm_files** list.

```
...
grid_squares = SP

#-----
# the list of additional openstreetmap objects to include (e.g. footpaths)
#-----
extra_osm_files = %(base_dir)s\maps\data\osm\footpaths_SP.osm,
                  %(base_dir)s\maps\data\osm\bridges_SP.osm

#-----
# the name of the compiled map (no spaces)
#-----
map_name = SP_with_bridges
...
```

*Note that if you reuse the same the **map_name** then the installer will prompt you to uninstall the existing version first. Alternately if you change the name then you will need to uninstall the previous version manually (using the uninstaller in the same directory as the installer) as Basecamp will only recognise one map with a given combination of Family ID and Product ID for the map. You can of course install several maps with different Family IDs (see main text).*

Remember that when you run *basecamp* you might need to hit *ctrl-G* twice in order to refresh the tile cache.

Here is a snapshot of the new version of the map (bridges highlighted):

