HANDS-ON WORKSHOP: SPATIAL DATA VISUALIZATION

Using selected Open Source tools and Open Data to visualize your own spatial data

Barend Köbben

Version 2.0
September 4, 2014

Contents

Key points

This is the exercise description for the hands-on workshop "Cartography in a Web World", part of the Nationale GI Minor. In these exercises you are invited to experiment with web-based technologies to visualise spatial data. There are two main parts (A and B), which you can do according to your own interest. The content offered is:

- **Using OpenStreetMap**, the crowd source initiative to create and provide free geographic data, such as street maps, to anyone;
- **A: using CartoDB to create an Enschede atlas**. CartoDB is an interactive web site that you can use to visualize your spatial data on the web;
- **B: Using OpenLayers and QGIS to build your own webmap of the Aamsveen**. OpenLayer is a JavaScript Library to build WebMapping clients; QGIS is an Open Source, stand-alone GIS software.

! → In many cases during these exercises, you will have to type code (HTML, JavaScript or MapServer configuration code). It's very easy to make mistakes in such code. HTML code and MapServer map files are not case–sensitive, but JavaScript is: the variable `mySomething` is different than the variable `MySomeThing`! Also take care of the *special character* (→) in the code examples we provide:

```
here is some code that should all be typed on 1 line in your→
 file but is divided over 2 lines in our example...
```

→ this character means the line should be typed **without interruption**, the move to the next line in our example is only because it would not fit otherwise. So do **not** type a `return` or `enter` in this place!

Typing the code of longer listings is usually not necessary: You can copy the code from the `filefragments` folder in the exercise data. In this folder you will find all code fragments from the exercises in text files with the same names as the listing. Do **not copy** from the PDF exercise description!

Note that throughout this exercise, you will need to exchange the placeholders `<NETPATH>`, `<URLPATH>` and `<CGIPATH>` with the actual paths needed (see the document "Setup of MapServer" for that...!

There are several software tools that can help you: Use a text–editor that is more intelligent than your basic text editor, e.g. on MacOSX and Linux use *TextWrangler*, on Windows *Notepad++*.This will provide you with line numbers, automatic highlighting of recog-

nised HTML and JavaScript keywords, etcetera.

Use a modern web–browser, i.e a recent FireFox, Chrome or Opera, or Internet Explore if you can install version 9 or higher. This because these are HTML5 compatible and have built-in web developer tools (e.g. an error console). This gives you useful error messages, code views of HTML, CSS and a JavaScript console, network traffic monitoring, etc. . .

1 Open Data: using OpenStreetMap

First we'll show you a prime source of free maps and data on the web: OpenStreetMap.

> **Note:** The OpenStreetMap Project, based at openstreetmap.org, is the worldwide mapping effort that includes more than 400,000 volunteers around the globe. OpenStreetMap is an initiative to create and provide free geographic data, such as street maps, to anyone.

There are many ways in which you can access the OpenStreetMap: as a simple webmapping service (not unlike Google and Bing Maps, but based on truly free non-proprietary data on a non-commercial website), as a webservice in various gis–viewers and as a database service, providing the actual vector data in raw form. This is maybe the most important difference between OSM and the others: You can download the raw data of OSM and use it for your own projects free of charge, and you can also edit the data directly and in this way contribute to the building of the OSM itself!

Task 1 : Visit http://www.openstreetmap.org/ using a web browser. Try to find the ITC building (it's just North–West of the main station in downtown Enschede, a town in the East of The Netherlands). . . •

The OpenStreetMap site itself uses the OpenLayers Javascript API, just as we will do later ourselves. The icons you see in the map are the default Graphics User Interface (GUI) of OpenLayers. They offer the following interactivity:

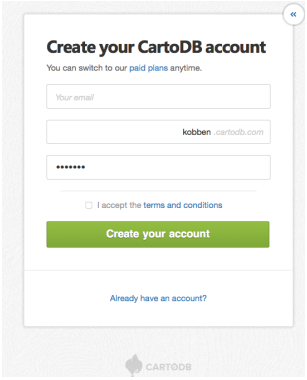
- You can *pan* using the arrow icons, or by dragging the map;
- You can *zoom in* using the + icon, or shift–drag a zoom box in the map;
- You can *zoom out* using the – icon;

Using the OpenStreetMap site as described above is fine for casual map browsing, but what if we want to have more control over the content and looks of our map? There are many ways to reach that goal, in this workshop we use two quite different solutions:

1. Using an on-line map creation and data visualization application, called **CartoDB**;
2. Creating our own webpage with interactive maps, using the **OpenLayers** javascript API, and using OpenStreetMap as a reference. We will also add some new data using QGIS, a free Open Source GIS.

2 A: USING CARTODB TO CREATE AN ENSCHEDE ATLAS

2.1 The CartoDB web application



CartoDB is an interactive web site that you can use to visualize your spatial data on the web. On their website (`http://cartodb.com`) they describe it as follows: "CartoDB is a product of Vizzuality, a data visualization consulting company, and was first created to meet the companyńs own internal needs to map large and complex geospatial data sets. While using the platform for various internal projects we realized its full usefulness and potential and decided to make it open source".

On the site there is also a small intro movie: `http://vimeo.com/vizzuality/introducing-cartodb`

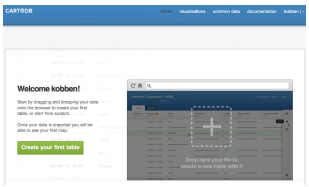
CartoDB is free to use for testing and small projects (using up to 5 data tables and 5Mb of data storage), but you will need to sign up for it:

Task 2 : Go to the website and click the "Sign up now" button. Put in your details and create your account. •

2.2 Mapping socio-economic data for the Enschede neighbourhoods

We will use socio-economic data for the "buurten" (neighbourhoods) of Enschede municipality . This data was retreived from the Netherlands Central Bureau of Statistics in 2011. The data can be found in your data directory: There is a file `enschede.zip`, which is a ZIPped version of a shape file, and the data directory also includes a small document with *metadata* for this dataset.

Creating a data table



We will first have to import the data into the web site's database.

Task 3 : After you have logged in to the CartoDB site, Click the "Create you first table" button. Click "select a file" and browse to the file `enschede.zip`. You do not have to unzip the archive first, CartoDB uploads the file, unzips the shape file and then imports it into the PostgreSQL/PostGIS database on the CartoDB servers. . . •

Note: PostgeSQL/PostGIS is a popular Open Source *spatial database back–end* that stores the spatial data using the Open

Geospatial Consortium Simple Features specifications. As a platform, the object–relational DBMS PostgreSQL is a solid DBMS that has a reasonably gentle learning curve, yet is very appropriate for advanced database applications, and its documentation is very transparent. PostGIS in addition, is the leading open standards implementation of spatial vector management, and enjoys a lively and supportive user/developer community.

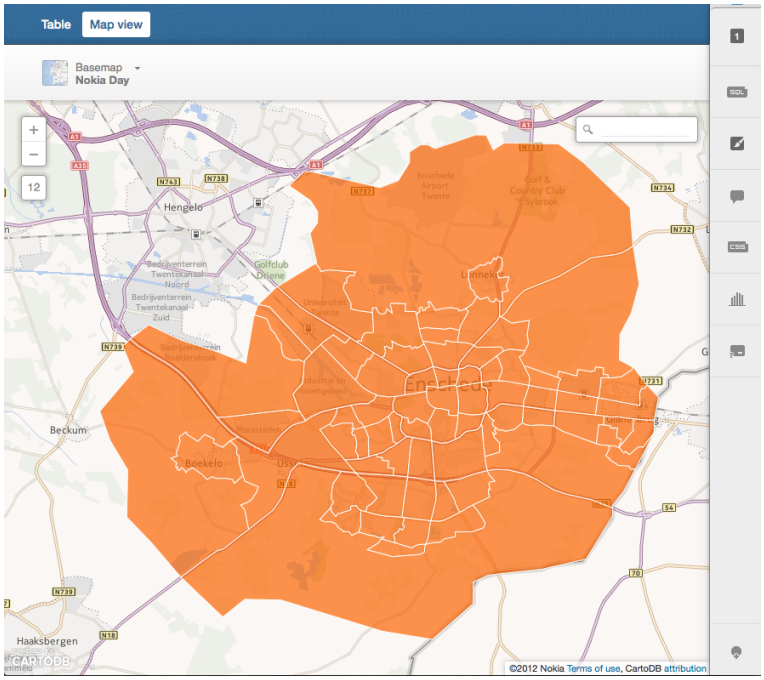
You can now use this data in several ways. Because it is stored in a database, you can use the tools to do selections, editing and even spatial analysis. If you are familiar with the SQL query language, you can use the small "SQL" tab in the right-hand side to directly work with the data in SQL.

Creating a map

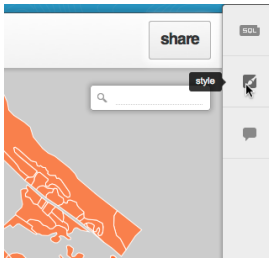
But of course, you can also show the data as a map:

Task 4 : Click the "Map View" tab. You see the data shown on top of a general background map. You can change which background map to use with the 'Basemap' menu at the top. There are familiar layers such as the Google Maps ("Gmaps Roadmap", etc.). The layers called "CartoDB" are visualizations made by the CartoDB people on the basis of OpenStreetMap data.

Now try to zoom out so that the whole Enschede dataset is shown, on a suitable background map layer... •



Changing the map visualization



You can click on the "wizards" tab to change the way the data is mapped. There are wizards to generate four basic types of thematic maps:

1. **Simple:** This is a simple type where all elements (in our case polygons) are mapped the same. You can use the menus below to set Polygon Fill and Stroke colors, as well as their transparencies. You can also choose to label the elements with the content of one of the attribute fields;

2. **Choropleth:** A "choropleth map" depicts relative ratio values in different colour *values*. Colour values for cartographers are *shades* of a colour: E.g. ranging from white to black, or from light red to dark red.

3. **Category:** This is what cartographers call a "chorochromatic map", where you map nominal (sometimes called categorical) attributes with an polygon fill for each different instance.

4. **Bubbles:** This is what cartographers call a "proportional point symbol map", where you proportionally scale symbols (in this case circles) with some numerical attribute.

Task 5 : Now decide which Style is suitable for the attribute column "BEV_DICHTH" (this column holds the population density in persons/km2 per "buurt"). Experiment with the setting in the wizard to get a map that is cartographically correct and also nice to look at. This is not an easy task, do not be satisfied too quickly! Ask your fellow students and/or staff members for their opinion on colours, classes and the usability of your map. •

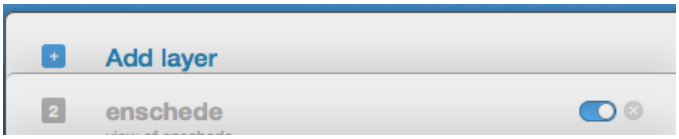
You can also tweak the mapping beyond the wizard possibilities by using the more powerful CartoCSS styling: Advanced styling with CartoCSS uses the concept of Cascading Style Sheets (a W3C standard for styling in webpages) to style the maps. The CartoCSS language is an easy, flexible, and powerful way to making a better looking map. If you know how to use CSS to style websites, you already know how to use CartoCSS. The CartoCSS tab also has a link to the reference site for the CartoCSS language. In this exercise we will not use these more elaborate styling possibilities.

Changing the info window contents

Note that if you click anywhere in your map, an Info Window will pop up with the attribute data of the polygon you clicked on. You can actually edit which of the data attributes will be shown:

Task 6 : Click the "info window" tab. You see all the attributes found in the data, with a small switch to change the visibility in the info window (on/off), as well as a "title" checkbox to determine if the title (name) of the attribute should be show also. Experiment with this until you get the items in the info window that you find relevant to show... •

Adding layers



You now have a map with two layers: a Base Layer and the added thematic layer. You can add additional thematic layers to this map: Simply use the "Add layer" button to add additional map levers, either from the same data or from other data sets.

Publishing your visualizations

CartoDB offers a wizard to publish your maps on the WWW for others to use. Once you have a map view, with all the layers that you like and want to publish, you can click the "publish" button that appears in the upper-right corner of the page. A dialog will pop up that lets you configure the interface of the map (e.g. whether you do or do not want a search box, a layer selector etcetera).

At the bottom of the dialog you can copy several types of links for pointing your users to:

1. a simple web URL where the users can find your map;
2. HTML code to use for embedding the map in a web page of your own;
3. an API URL that can be used to integrate the map in a larger web application of your own.

2.3 Creating an Enschede Buurt Atlas

Now create and publish appropriate maps for selected other attributes in the socio-economic dataset:

Task 7 : Out of the available attributes, select those that you think together would form a representative view of the different neighbourhoods. For each of the data attributes, think carefully which of the map types you should be using according to the rules of Cartographic Grammar and the Thematic Mapping theory that you have learned. Create maps that are as complete as possible, i.e. with appropriate titles, classification, legends and info-windows. . . •

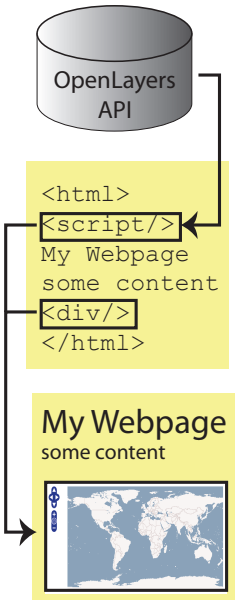
---

3 B: USING OPENLAYERS AND QGIS TO BUILD YOUR OWN WEBMAP OF THE AAMSVEEN

3.1 The OpenLayers API

OpenLayers makes it easy to put a dynamic map in any web page. It can display map tiles and markers loaded from any source.

> **Note:** OpenLayers is a pure JavaScript library for displaying map data in most modern web browsers, with no server-side dependencies. OpenLayers implements a JavaScript API (Application Programming Interface) for building rich web-based geographic applications, similar to the Google Maps, with one important difference: OpenLayers is Free Software, developed for and by the Open Source software community based at `http://openlayers.org/`. OpenLayers is written **in object-oriented JavaScript**, using components from *Prototype.js* and the *Rico* library. In these exercises, we will only show the basic building blocks, and how to employ them. Those wanting to go further, should check out the development pages and the examples at the website.



The latest version of the OpenLayers script library is always available on the OpenLayers website. You can "install" the API by including a link to the Javascript files in your own HTML webpages and then call the methods and properties of the library using simple JavaScript functions. Using the Openlayers API is done by creating *webpages* (using HTML) that include Javascript ***script***; this code makes calls to the API methods to create the necessary map object and connect that to an HTML *placeholder*. Mostly we use an HTML `<div>` element as a placeholder.

The OpenLayers API has two concepts which are important to understand in order to build your first map: *Map*, and *Layer*. An OpenLayers *Map* stores information about the default projection, extents, units, and so on of the map. Inside the map, data is displayed via *Layer*s. A *Layer* is a data source – information about how OpenLayers should request data and display it. We then uses the methods and properties of the API to change the content and behaviour of the map. In practice, all this means typing (and/or copying) HTML and JavaScript code.

3.2 Making an OpenLayers viewer for OpenStreetMap

In listing 1 you see the most basic example of using OpenLayers with the OpenStreetMap service.

Task 8 : Create an HTML page with the content of listing 1

and save it as a `osm.html`. You can type the code, but it is easier to copy it from the file we stored in the `filefragments` folder (in the exercise folder that was provided with this exercise). Do **not** copy from this PDF file! Make sure you save this file as a new file with the extension `.html`, *not* `.html.txt`!). View the result in a web browser. •

Listing 1: osm.html

```
<!DOCTYPE html>                                          html5 document header
<html>
<head><meta charset="utf-8">
<title>OpenLayers Basic Single OSM Example</title>       html title
<script src="OpenLayers/OpenLayers.js"></script>         include the OpenLayers API
<script type="text/javascript">                          script for our map
  var myMap, myOSMLayer;                                 define map and layer object
  var myCenter = new OpenLayers.LonLat(                  define center
    254031,6254016 );                                   XY of Paris
  function init() {                                      function triggered on load
    myMap = new OpenLayers.Map("mapDiv");                create map object
    myOSMLayer = new OpenLayers.Layer.OSM("OSM Map");    create OSM layer
    myMap.addLayers([myOSMLayer]);                       add layer to map
    myMap.setCenter(myCenter,16);                        zoom to center
  } // end    init
</script>
</head>
<body onload="init()">                                   run init script after page loaded
<div id="mapDiv"                                         map placeholder
   style="width:400px; height:400px;"></div>             placeholder style
</body>
</html>
```

The result should look like figure 1, showing the OpenStreetMap for the Porte Maillot area in Paris (France).

You can set up the OpenStreetMap to start at any place on the globe, by changing the coordinates that were used in the `myCenter` variable:

```
var myCenter = new OpenLayers.LonLat(254031,6254016);
```

But in order to find which coordinates to use to zoom to, it would be nice to have a knowledge of where (in coordinates) you are in the map. For that we will include a coordinate–readout line and a

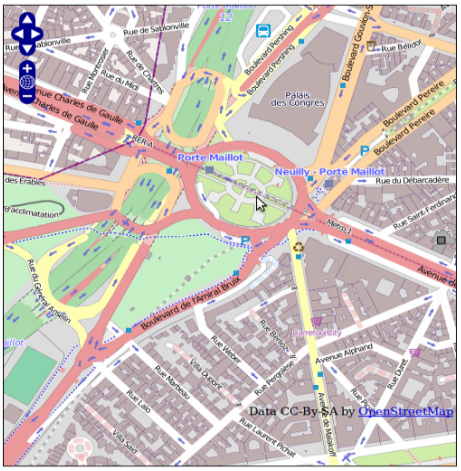


Figure 1: result of loading listing 1.

scale bar:

Task 9 : Add the following line in the script **just before** the line with the `myMap.SetCenter` command:

```
myMap.addControl(new OpenLayers.Control.MousePosition());
myMap.addControl(new OpenLayers.Control.ScaleLine());
```

Save the results as `osmPlusCoordinates.html`. Try out the result in the browser. •

The coordinates you see are X– and Y–coordinates in a Mercator projection on the spherical WGS84 datum. This is used nowadays by most popular public webmapping services (such as Google Maps, Bing Maps and OpenStreetMap). The projection is officially standardized as **EPSG code 3857**, and named "WGS 84 / Pseudo-Mercator".

> **Note:** Unfortunately, lots of software uses instead the un-official EPSG code 900913 (chosen because it sort of spells "google"), that was introduced and has become popular before the official standard was set.

Now you can change the line `myMap.setCenter(myCenter,16)` to set an alternative starting point (change `myCenter` variable) and zoom (from 0–18) for the map.

Task 10 : Try setting up the map in such a way, that it starts zoomed in on the **Aamsveen Nature Reserve**... •

Note: The Aamsveen Nature Reserve is a 130 ha large area just South-East of Enschede. It comprises of the last remnants of a very large peat area which in 1976 was turned into a protected nature reserve. Stichting Het Overijssels Landschap is responsible for the maintenance and protection. See http://www.landschapoverijssel.nl/aamsveen for more information (in Dutch).

3.3 Introduction of QGIS

QGIS (http://qgis.org) is an Open Source, stand-alone GIS client, programmed in C++ using the multi-platform Qt framework. You can use it to work with vector– and raster–files, databases or any open standard WMS or WFS–compliant server. A strong point of QGIS is its extensibility: you can add plug–ins that are written in either C++ or Python, and you can connect it to GRASS, a powerful GIS analysis tool.

Task 11 : Users at ITC can use the ITC software manager (if installed on your computer). Other users go to the URL http://qgis.org/ and download the latest stable version (sometimes called 'release', whereas 'master' is the development version). Choose the one suitable for your operating system. Windows users should use the 'standalone installer', and take care to install the correct 64- or 32-bits version, depending on their hardware. •

QGIS can load maps and data from a huge array of possible sources:

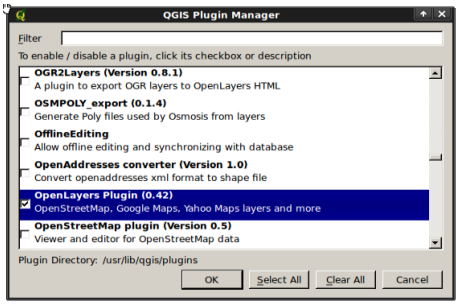
- online maps served as an OGC-compliant Web Map Service (WMS);
- online spatial data served as an OGC-compliant Web Feature Service (WFS) and Web Coverage Services (WCS);
- various other map services, such as OpenStreetMap, Google Maps, Bing Maps, etcetera;
- most vector formats supported by the OGR library, including ESRI shapefiles, MapInfo, KML, GPX and GML;
- raster formats supported by the GDAL library, such as digital elevation models, aerial photography or satellite imagery;
- spatially-enabled PostgreSQL tables using PostGIS and SpatiaLite, by means of a 'live' connection to such databases;
- locations and mapsets from GRASS (an open source GIS);

The list is in principle endless, because the functionality of QGIS can be extended by *plugins*. Plugins add functionality to QGIS, and

they are usually made by others than the main QGIS developers. Because QGIS is an Open Source software, anyone can add plugins, they can be programmed using C++ or Python.

3.4 Using QGIS to create your own data

QGIS is not limited to loading data from services and files, it also allows you to create data in many of these formats, and that is what we are going to do next. We will use an existing background map from the OpenStreetMap webmapping service to digitize some of our own data...



Task 12 : Start QGIS. To enable the use of OpenStreetMap, we will use a plug–in that offers that functionality.

First check if the OpenLayers plugin is already installed: Choose `Plugins > Manage Plugins...`, and type "openlayers" in the Filter box. If the plugin is available, it will be listed below. You can go on to the next Task.

If not, open `Plugins > Fetch Python Plugins...`, go to the tab `Repositories` and check the list. Now go to the tab `Plugins` and find the one called "OpenLayers plugin" and enable it by making sure it is selected. If there are several versions, use the latest one. Press OK. •

This plugin offers access to many publicly available map services, such as OpenStreetmAp, Google Maps, Bing Maps and others. But despite its confusing name it does *not* offer direct access to the OpenLayers Javascript API...!

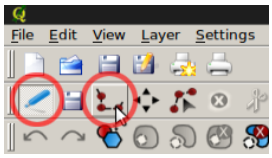
Now let us use the plugin to have OpenStreetMap in the background:

Task 13 : An item called `OpenLayers Plugin` should now be available in the `Plugins` menu. Choose `Add OpenStreetMap layer` from this submenu, and the OpenStreetMap will be opened, zoomed out on the whole world.

Navigate to the location you set for the OpenLayers webpage you made earlier in the exercises. •

We will create a very simple vector line dataset, that depicts a walking route. When creating new data, QGIS will adopt the cur-

rent projection of the map window. Therefore we must make sure that the data is saved correctly projected, otherwise our new layer won't fit the OpenStreetMap base later on:



TASK 14 : Choose the menu `Layer > New > New Shapefile Layer...` The New Layer dialog opens (see below). Make the following settings:

1. For `Type` choose `Line`;
2. Click `Specify CRS` and make sure you choose the WGS84/Pseudo Mercator (EPSG:3857). It can be found under `Projected → Coordinate Systems > Mercator`;
3. In the `New attribute` section, create one attribute of type `Text`, call it for example routeName. Click the `Add to → attribute list` button to actually add it;
4. Click OK to create the new file. Save it named "myRoutes.shp".

Now you can start adding lines for your route:

Click first the `Toggle Editing` button in the QGIS menubar, then the `Capture Line` button. Create a nice walking route from the Enschede main station to the ITC building. You can add points to the line by clicking in the map, undo them by using the `CTRL-Z` key or `Edit > Undo` menu.

If you have finished a route, right-click, fill in the route name and press OK. Use the `Toggle Editing` button again to stop editing. Remember to **save regularly**!

You can change the visualisation of the line by right-clicking the layer name in the layers list, or choosing the `Layer > Properties` menu.

•

3.5 Using QGIS to save KML data

Note: KML is an XML notation for expressing geographic annotation and visualization within Internet–based, two–dimensional maps and three–dimensional Earth browsers. KML was developed for use with Google Earth, which was originally named Keyhole Earth Viewer. The KML file specifies a set of features (place marks, images, polygons, 3D models, textual descriptions, etc.), and locations are always expressed in longitude and a latitude. KML shares some of its structural grammar with GML. KML files are very often distributed in KMZ files, which are zipped versions of the file, with a `.kmz` extension.

KML is now an international standard of the Open Geospatial Consortium, and can be used in many geo-browsers and GIS software.

It is simple to save a QGIS vector layer to the Keyhole Markup Language format:

Task 15 : Right-click on the layer in the Layers list or choose the menu `Layer > Save as...` In the Format menu, choose "Keyhole Markup Language (KML)", choose a place and name ("myRoutes.kml") and save the file by clicking OK. •

3.6 Testing and refining the KML

The KML that you have created can be tested in many environments. Most GIS software nowadays can import (and export) KML. But the most logical test platform is probably the software that the KML format was engineered for: *Google Earth*.

Task 16 : Start Google Earth. Choose the `File > Open...` menu and find the file 'myRoutes.kml' you created earlier. The file should be loaded and Google Earth zooms to the place on earth where the KML is placed. •

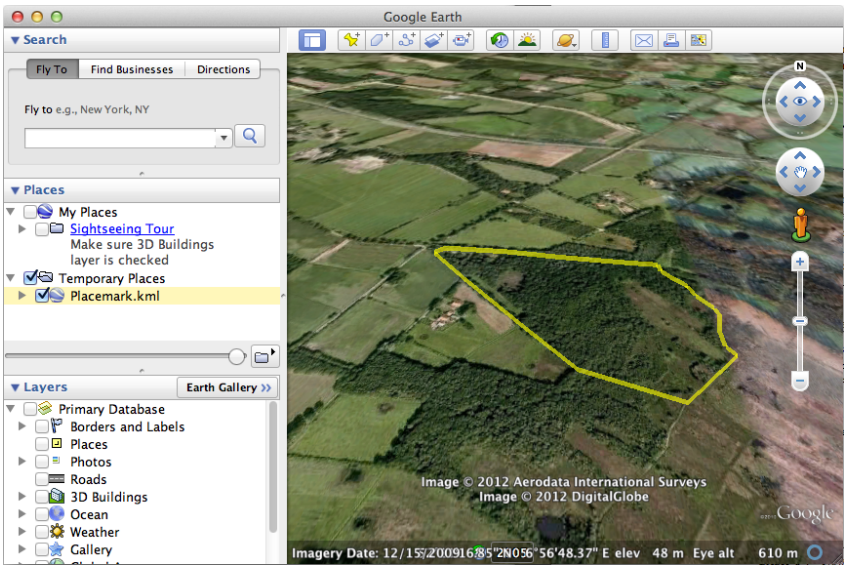


Figure 2: result of loading the route KML in Google Earth.

You can also use Google Earth to refine the visualization of the KML. You can right-click on the name of the KML file in the Google

Earth 'Places' panel and use the `Get Info` menu to edit things like the description, the style (color and width of the line) the starting view, etcetera.

Important Note: If you made changes, do not forget to right-click again and use the `Save Place As...` menu! It is actually a good idea to use Google Earth to "save as" the KML anyway, because the KML generated by GE is the most sure way to have a KML according to the official KML standard (that was created by Google, after all)...

3.7 Combining OpenStreetMap with our KML overlay data

In the previous section we have created some local data that fit the OpenStreetMap web map. The last step now is to publish this overlay data on the webpage. Let's go back to the OpenLayers web page we have created earlier, and add the KML file we created to the OpenStreetMap background:

Task 17 : Add the code in listing 2 to the html–file you made earlier (use the filefragments folder). Add it **after** the `myOSMLayer` definition. Now look for the line in the code that reads:

```
myMap.addLayer([myOSMLayer]);
```

To include the new layer in the `map` object **change** it to:

```
myMap.addLayers([myOSMLayer,myKML]);
```

Test it in the browser. •

Listing 2: kmlLayer.txt

```
myKML = new OpenLayers.Layer.Vector("KML route", {                  generic vector type
        strategies: [new OpenLayers.Strategy.Fixed()],              loads all features at once
        protocol: new OpenLayers.Protocol.HTTP({                    using web protocol
            url: "myRoutes.kml",                                    path to file
            format: new OpenLayers.Format.KML({                     use KML parser
                extractStyles: true,                                use KML styles
                extractAttributes: true                             use KML attributes
            })
        })
    })
```

Now we have more than one layer, we want to control which of those are shown in our map. For that, we will add a layer-switcher

control:

Task 18 : After the other addControl statements, add the line:

```
myMap.addControl(new OpenLayers.Control.LayerSwitcher());
```

Try it out in the browser. •

A new control should have been added to the map: next to the *pan* and *zoom* tools, there now should be a little + icon in the *upper-right*. Clicking it will reveal a Layer Control, in which you can switch layers on and off.

For your final map, you could use this mechanism to have several routes, in different KML files (having different symbols and/or colours), in different layers in the OpenLayers map page...