Term Project – Web Mapping Service

WMS – Vancouver

GEOG – 413

Michel Laterman

230080069

1. Introduction

A web mapping service (WMS) is a system to provide georeferenced maps over the internet. For my project I have chosen to set up a WMS to serve maps that are created on request. I chose to do my project on something that was more related to my field of study (computer science) and not strictly analysis or cartography. The area of interest for my project is the city of Vancouver.

The WMS server software I am using is ms4w (http://www.maptools.org/ms4w/) – a software suite that installs a MapServer environment on a windows machine, I chose this because the operating system I used was Windows 7. Some of the data that I am using was hosted in a database on the local machine, the database itself is managed by PostgreSQL (http://www.postgresql.org/) with the PostGIS (http://postgis.refractions.net/) extension.

2. Data

2.1 Data Sources

I used the city of Vancouver's data-sets (http://data.vancouver.ca/) for my mapping service. These are provided in many different formats, the ones I chose to use are ESRI's shapefile – as it is the industry standard and is well supported by MapServer, and KML (Keyhole Markup Language) files – as they are quickly becoming popular due in part to support and development efforts from Google. The background satellite image of the region was found using the United States Geological Survey's Glovis system.

2.2 Data Preparation

A majority of the layers defined in the mapfile are simply shapefiles. These required no special preparation in regards to getting them displaying on the server. For more general applications the shapefiles should follow the best practices of data management for GIS – it would likely need some preparation and possibly reprojection. As all the data was pulled from one provider this was not an issue in production.

Some other files taken from the datasets Vancouver has provided were KML files. More accurately they were .kmz, MapServer is able to support kmz files but I had difficulty getting the archived format to display so instead I simply extracted the kml file inside the archive and used that as the layer.

To have a better understanding of getting data from different formats to display I imported some shapefiles into a PostGIS database. The importing process was done with the GUI the PostGIS extension came with and no difficulties were encountered.

The satellite image was intended purely for cartographic reasons – without it the map would look pretty boring. The image itself was downloaded as a georeferenced JPEG file. Unfortunately MapServer can not display color JPEGs – the server is limited to grayscale (for JPEG images). To resolve this issue I used QuantumGIS to convert the file to a geoTIFF, this resulted in a very large image file (100+ MB) so the image was clipped to the study region.

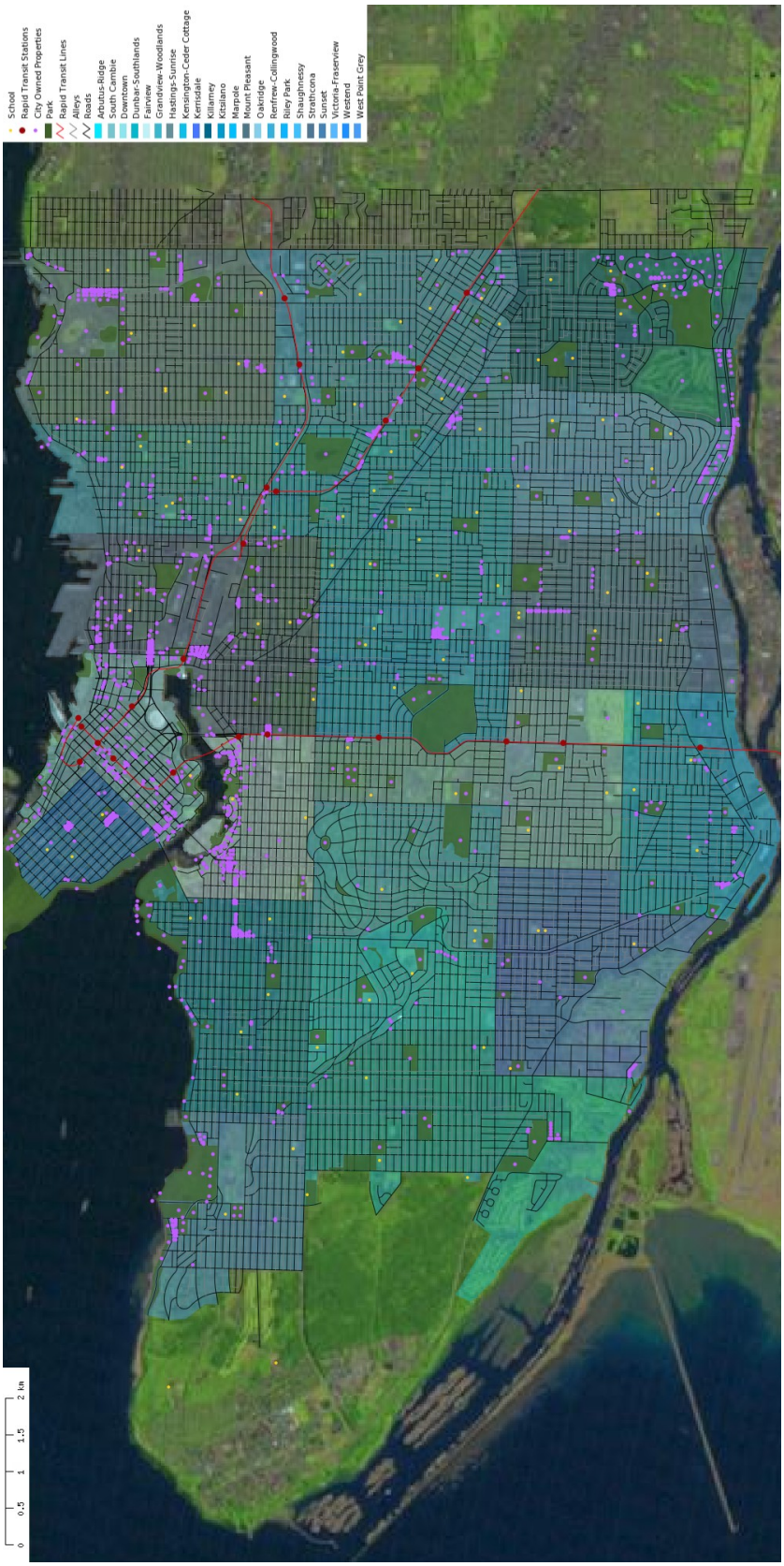
3. Mapfile

The mapfile is the file the server accesses when assembling a map. In the mapfile all the map layers are defined as well as some other information such as the output format and default projection system. The mapfile I created has some elements present that I do not use – such as the web definition section. This section is used when the map is accessed through another page (such as HTML) – this was not used as it is not in the scope of my project. The file follows the syntax required by the server. The extent of the study region is defined in a block using UTM coordinates as well as the projection system (UTM 10N), in a separate block. Output format and size are also specified – for this map I chose to create a 2000x1000 .png image. The png file has had the AGG filter applied to give a smoother output.

The layers are then defined starting with the satellite image, this is done because MapServer draws the layers in the order they are listed. The next layer is the local area regions shapefile (polygon), this layers is split into classes based on the “CODE” attribute, this is done so that the output map and the legend may distinguish between the separate polygons in the layer, the layer itself has a set opacity so that the background is not obstructed by the polygons. The next layers are the roads, alleyways, and one way roads (lines)– I did not give 'one way roads' a name attribute in the class (where the output color is defined) as the layer overlays with the public road layer and it would make the legend more confusing. The next layer is the transit lines KML file – unlike the shapefiles the data attribute of the kml layer specifies the layer of the file to use – not the data location (this is done in the connection attribute). The public parks polygons are then added from the PostGIS database – in this layer the connection attribute describes how to access the database and the data attribute describes what to retrieve from the database. The next layers are the public parks (polygon) city properties (points), transit stations (points) and schools (points). This order was chosen so that the points would not be obstructed by the other layers and the parks polygon was placed over the other polygons to improve output display.

4. Results

Map Generated with all layers visible.



5. Conclusion

A web mapping service is able to produce detailed maps on the fly for user requests. One issue I ran into when working on the project is that there did not seem to be a way to get the map to automatically color its classes – I had to specify exactly what colors to use. This feature may be possible when serving the map through an HTML page but it does not seem to be a function of the mapfile. Another issue is that some of the city property points are being displayed in the water – this is due to the fact that KML files use their own geographic system that is slightly offset from the conventional ones (that MapServer outputs them as). The scalebar and legend were embedded on the sides of the image to not obstruct the content – a white background was chosen (on the scalebar) so that it is visible against the dark blue water.