Incubating Project

Making Maps Pretty

The art of SLD and tricks for great looking maps

Nov 2, 2009

page intentionally left blank

Table of Contents

0 About the Authors

Jim Groffen is a Senior Software Engineer at LISAsoft. Working in IT since 1998, Jim has been with LISAsoft since 2005 working on various Spatial projects. Jim participated in OGC projects such as CGDI-IP and OWS-6. As part of OWS-6 Jim will be contributing updated WMTS support to the TileCache open source project. Other relevant areas of interest include spatial catalogues and registries, OpenLS and all things Python.

Andrea Aime is Geospatial Solutions Engineer at OpenGeo, a Open source enthusiast and a long time Java developer. Personal interest range from high performance software development, huge data volume management, software testing and correctness, spatial data analysis algorithms. Andrea Aime’s Specialties: Java development, Hibernate, Spring, GeoTools, GeoServer, J2EE in general, GIS theory and practice knowledge.

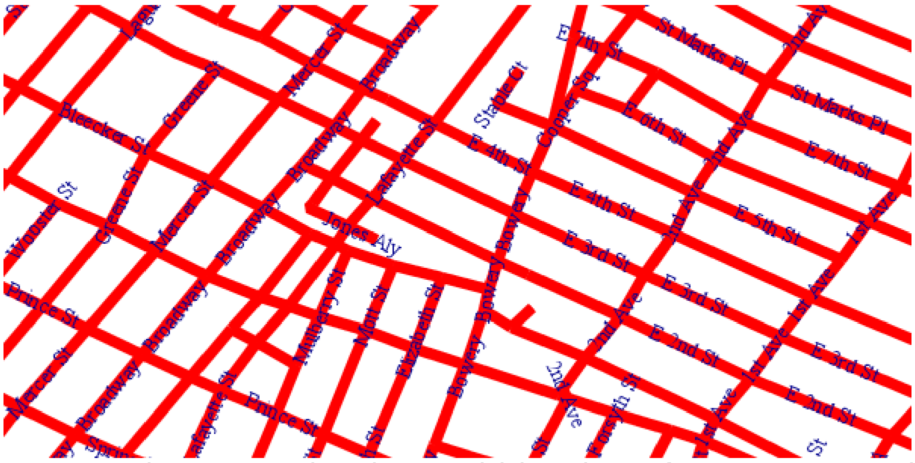
The authors would also like to acknowledge Jody Garnett who contributed various information, diagrams and review feedback. Paul Ramsey and Adrian Custer also provided useful information.

1 Introducing Cartography

When we talk about the style of maps we mean how maps are visually portrayed. What represents a road or a lake. In map making this is called Cartography. Cartography has long been considered a junction of science and aesthetics, the challenge is to make a map accurate, useful for a specific purpose and visually appealing.

Assuming the technical challenge of making a map correct, lets consider the other two goals. Both are tied to map styling. Consider the following map:

An isometric city scape looks great but it useful for vehicle navigation?



This map may work as a navigational aid but hopefully we are all in agreement that it's not the most aesthetically appealing map.

1.1 Setting Goals

Before starting work on styling maps lets first set some goals. What are these maps going to be used for? If we have a clear idea of what we are trying to get from out maps we can more easily make decisions between styling options. Lets break these goals down into a few questions:

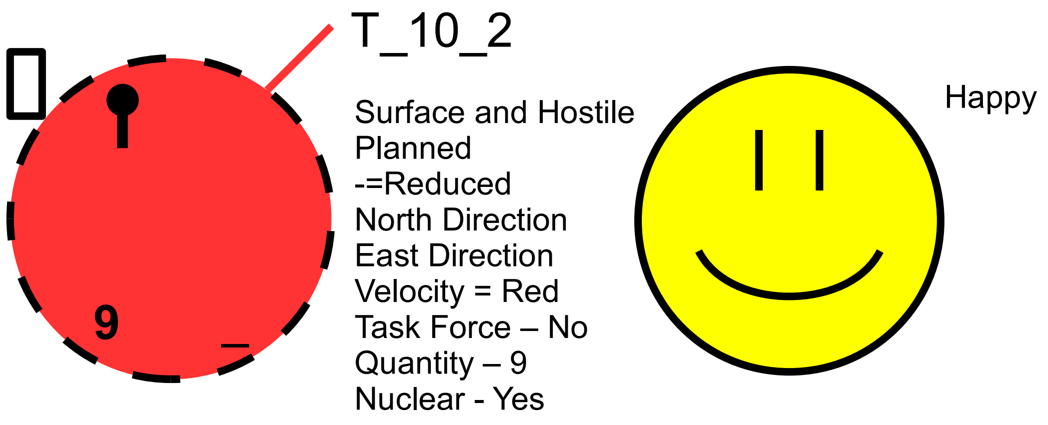
1.1.1 Who is the target audience?

- Who will be using these maps?
- What is their knowledge / skill level in relation to these maps and cartography in general?

An example of the kinds of decisions made easier by having a clear understanding of your target audience includes which set of symbol to use.

Below is a comparison of a MIL2525B symbol and a common internet convention.

MIL2525B has ten kinds of information to communicate in a single symbol!



In this scenario both symbol sets are relevant, it is the target audience that determines which is appropriate.

1.1.2What is the target medium?

Are the maps going to be viewed electronically or printed on big glossy sheets with any number of colours available? Target medium questions include:

- What is the device used? Desktop computers? Laptops? PDA's? Mobiles?
- What is the viewing area available to the map? Number of pixels wide and high for electronic display or some measurements for print mediums.
- How many colours will be available? Are the maps to be presented in black and white? This is relevant to printed maps and devices with limited display capabilities, and is also a performance consideration.
- If displaying on a portable device, what kind of bandwidth is available? Can the bandwidth handle large images with many colours?
- Will the printed maps be folded? In what way?

1.1.3What is the purpose of the maps?

- What will the target audience be using the maps for?
- What do we want the maps to convey to the target audience?
- Are they "You are here" maps? Are they "how to get here" maps?
- Are the maps meant to point out similarities between things? Are they meant to highlight the differences between things?

Maps for Navigation

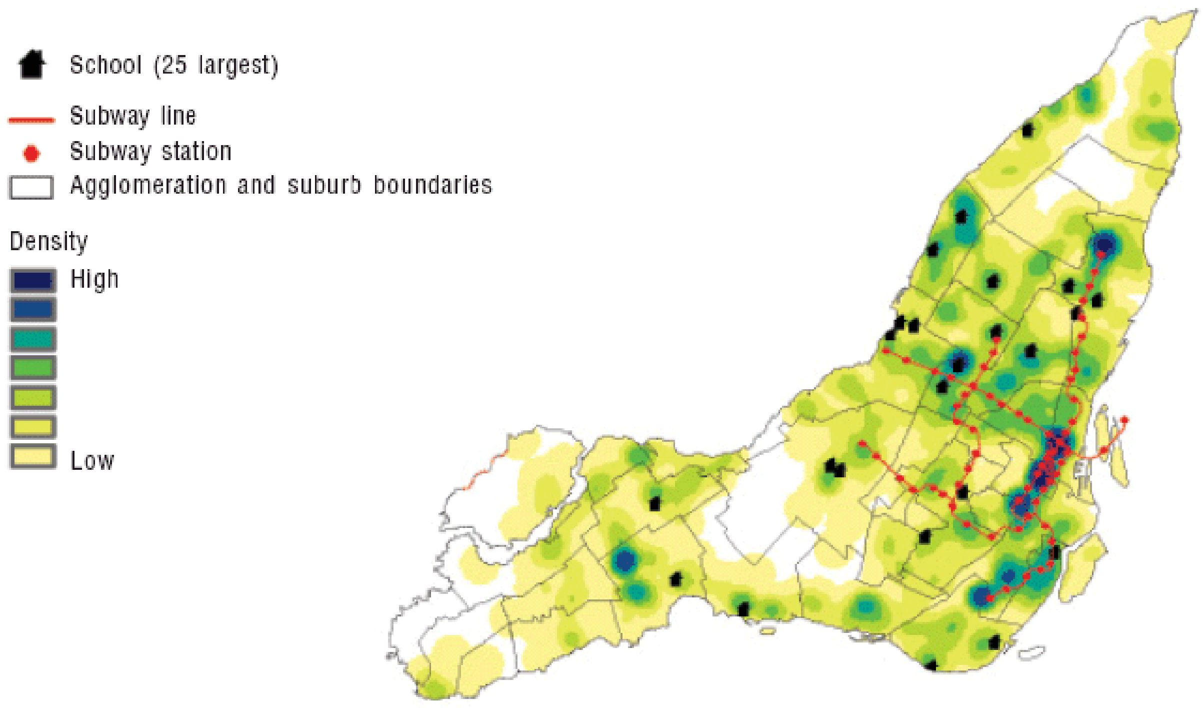
Pretty much everyone has experienced using a map for navigation. There are a variety of navigation map types. A directory showing what is near the current location. Various modes of transportation have different needs from the map. Consider a road map for driving, sailing or walking.

Spatial Distribution

When data has a spatial element it is often useful to be able to see the distribution of spatial data. For example say we are the Bureau of Meteorology and we have a data set of rainfall reading stations including their location. If we look at these stations on a map we could ask questions like; Are there are gaps in our rainfall reading locations; are there stations too close together? Mapping the spatial distribution lets us look for clusters and gaps.

Below is an example of a spatial distribution map that looks at the density of youth crime on the Island of Montréal, 2001.

Following the legend we can see that large schools are included on the map. This is an example of comparing two distributions; youth crime and large schools. Showing both distributions allows us to find contrasts or correlations between the data sets. Is youth crime more likely near large schools? Is there a correlation between youth crime and the public transport system? These kinds of questions can be considered when looking at this map.



1.2 Clarity

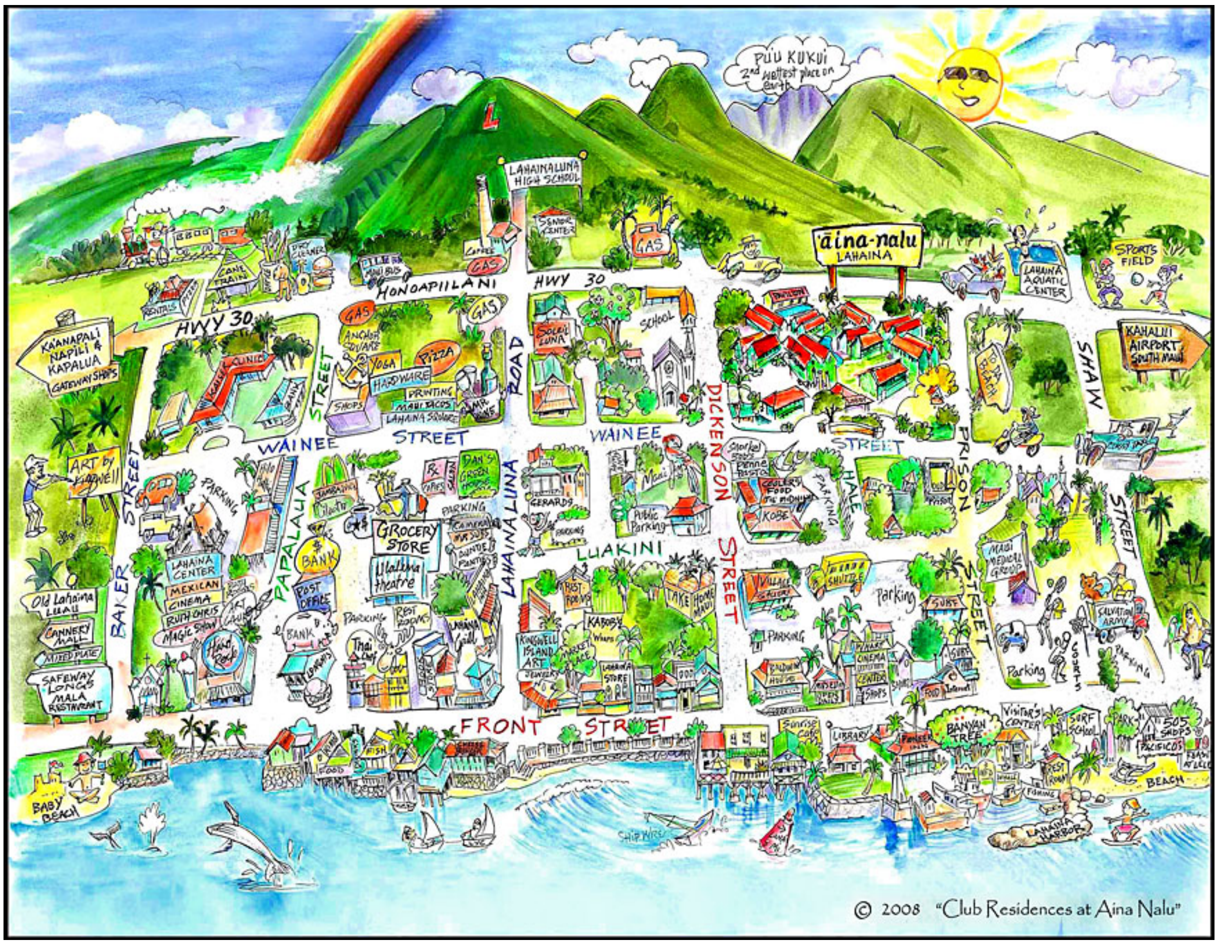
Knowing what the maps are for, who is going to use them and how they will be used means we can focus our efforts on ensuring all style decisions made target these goals. Maps can convey so many things, if we don't focus our styling efforts on our goals our maps will loose their clarity of purpose.

The goal of map styling should be: make your map as clear and easy to use for the purpose it has been created for.

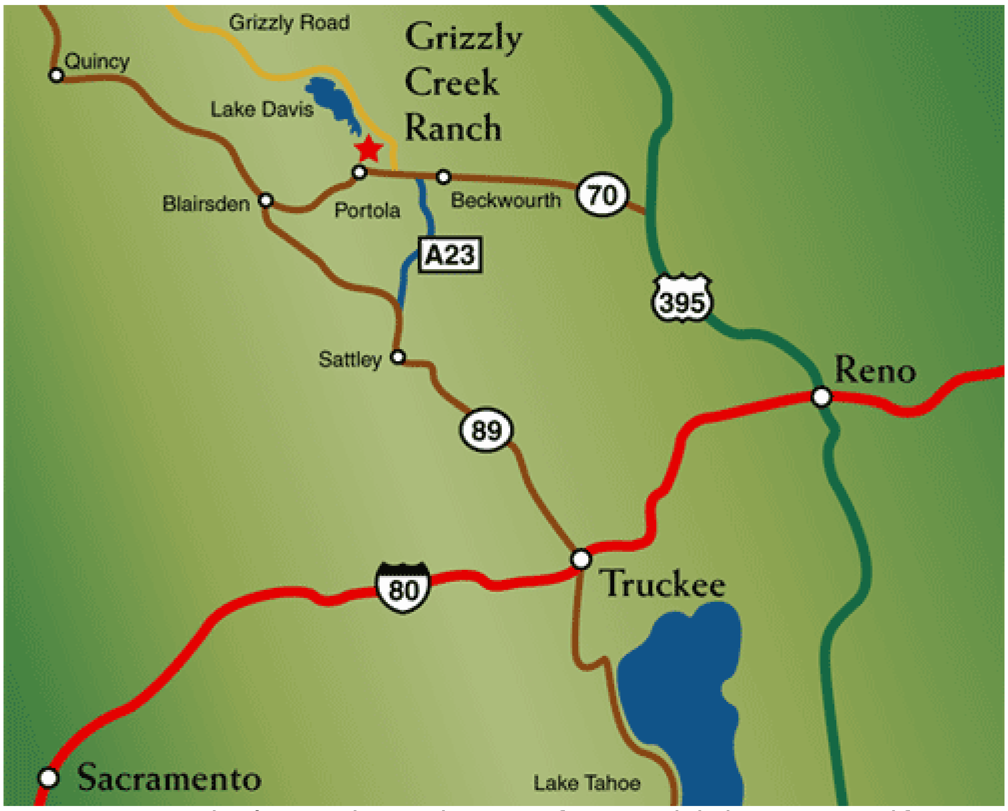
Lets look at some choices that may come up during our efforts.

1.2.1 Information Density

Out of the following two maps, Which map is better?



From http://www.mauifractionals.com/images/walking_map_large.jpg



From http://www.grizzlycreekranch.org/gfx/gcr_driving_map.gif

First of all we haven't set any goals yet so our decision at this point would be purely aesthetic. Lets look at these maps with particular goals in mind:

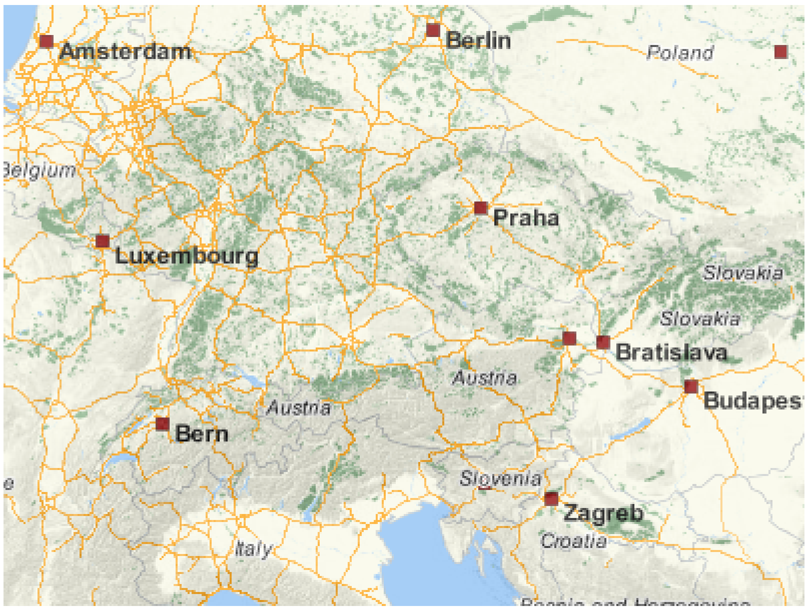
- Which map lets me find my way from A to B easier?
- Which map gives a better indication of what is nearby?
- If I'm navigating using this map, is one of them more useful if I'm in a car? What about if I'm walking?

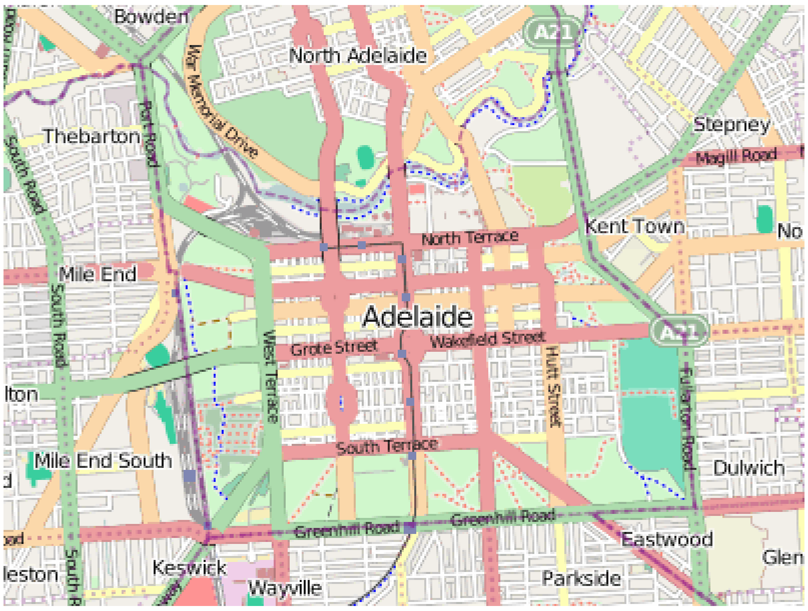
Information density affects the clarity of the map by obscuring information the user needs with information the user doesn't need.

The directions map has the minimum amount of information needed to get to Grizzly Creek Ranch from various places. It is clear and simple to use. When driving you want to spend as little time as possible interpreting the map. The walking map on the other hand has a broader purpose so more information is relevant to the user.

1.2.2Labels

This time we'll look at two maps and try to come up with situations where one map is superior to the other.

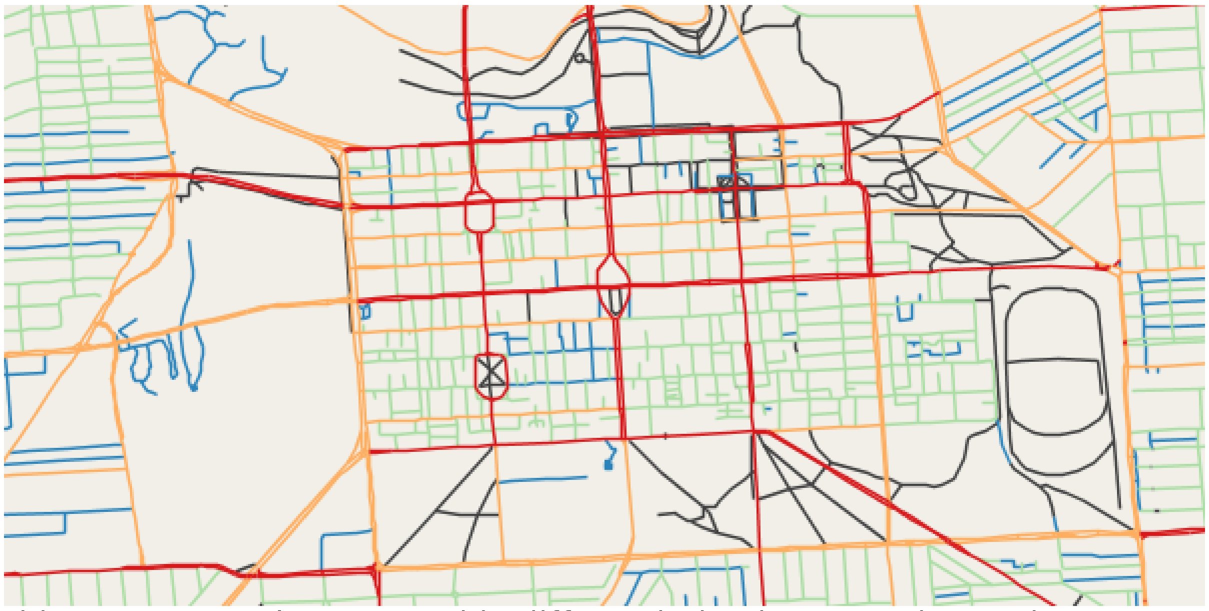




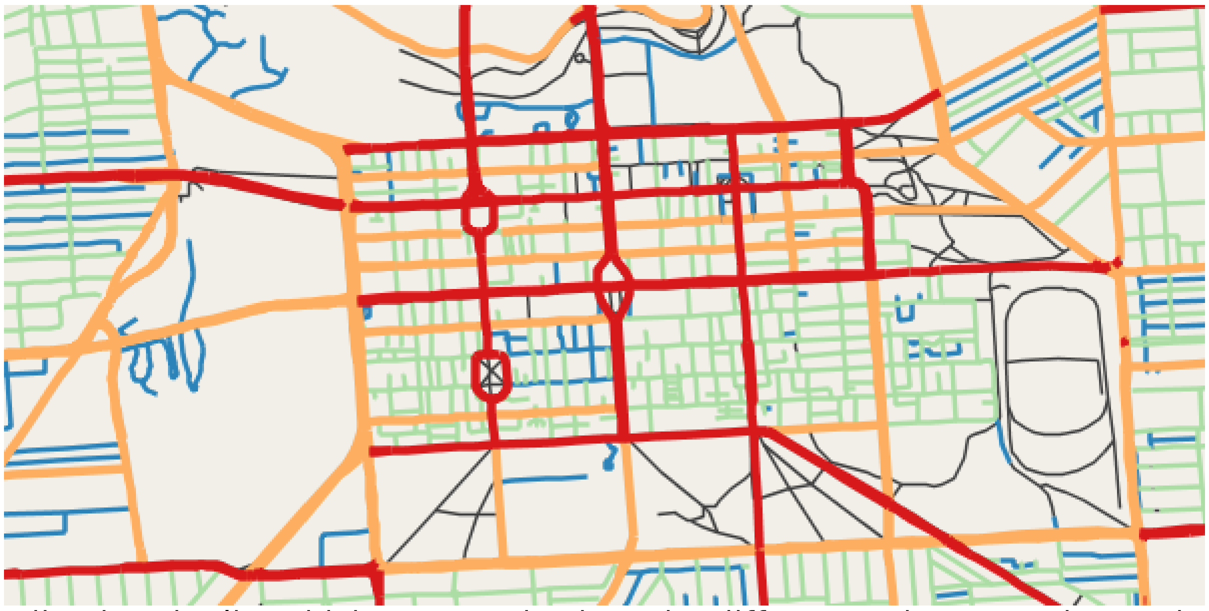
These maps both have labels with halos and without halos. A common reason to add halos to labels is to add definition – ensuring whats under the map can't interact with the label text. What are some reasons to use halos? What are reasons to avoid them?

1.2.3Line Styling

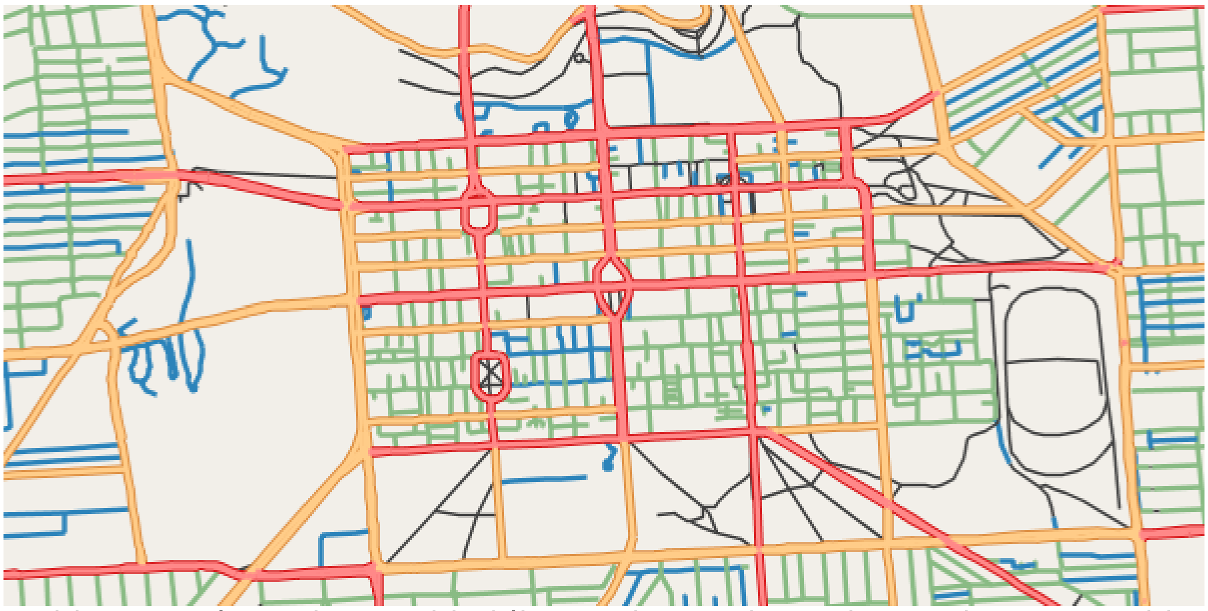
This time we are looking at adding edging to some lines on the map:

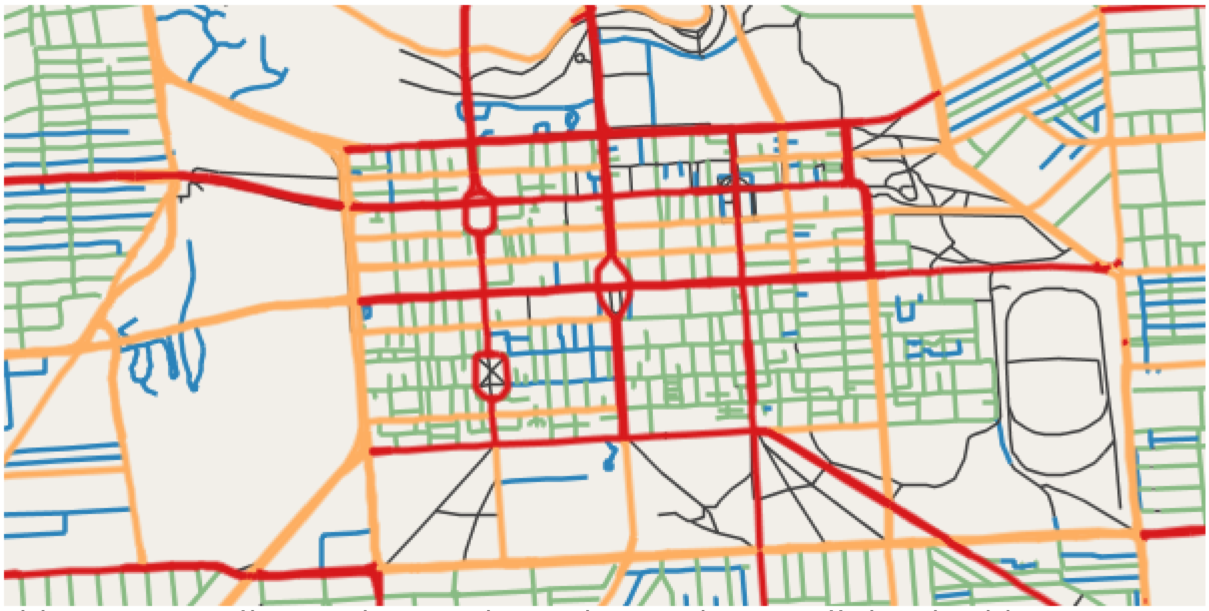


This map uses colors to provide differentiation between the road types.



Adjusting the line thickness emphasizes the differences between the road types.


In this example we have added line casing to the major road types. In this example we make the interior road colour pale with a dark edge.

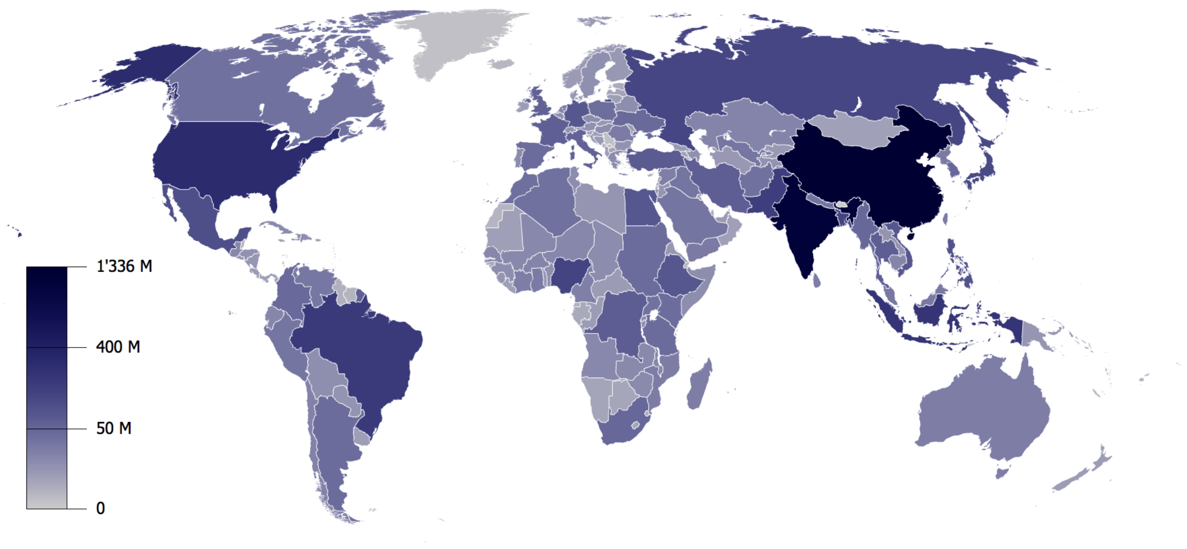

This map uses line casing on the major roads as well, but in this case we have used a 50% transparent edge to soften the edges of the major roads. Compare this image to the thickness image to evaluate this effect.

1.2.4Colour Scheme

The colour scheme of the map is not only important aesthetically but is also a tool for conveying the nature of the data. To illustrate, we'll look at some example spatial distributions.

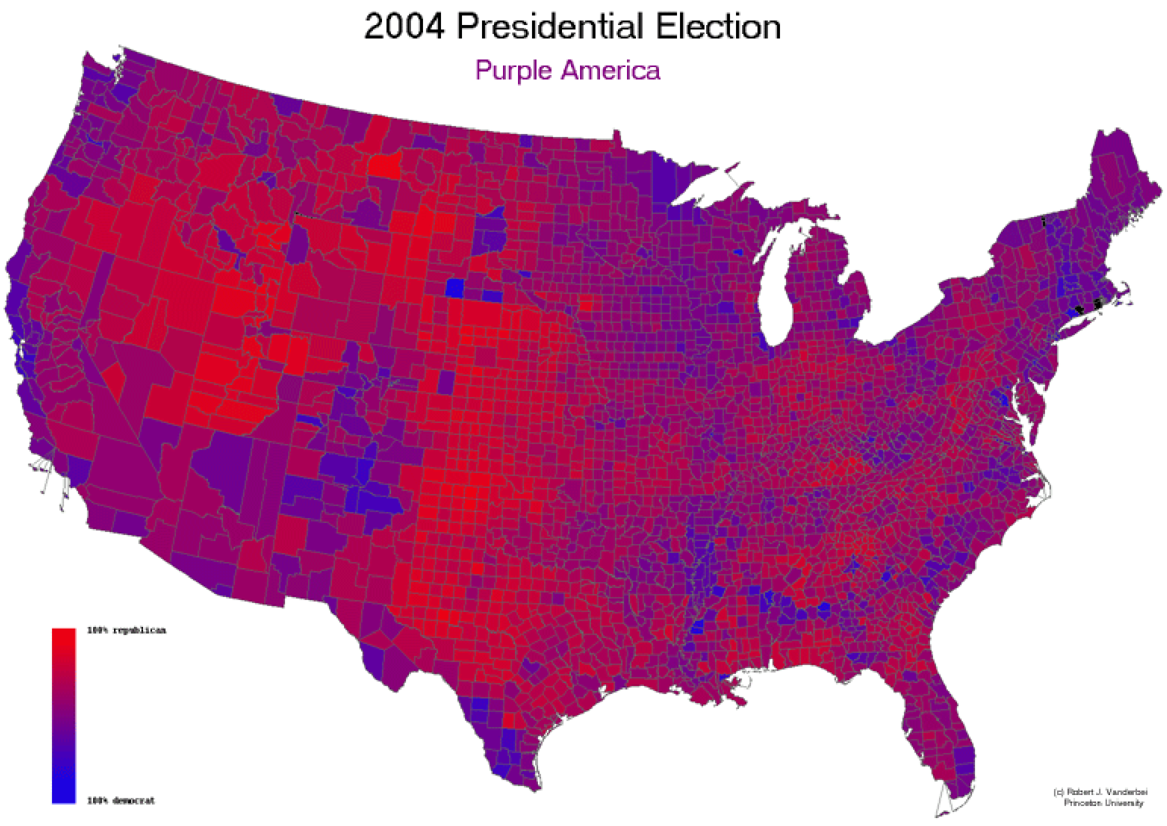
Sequential

This first image represents country population levels. The gradient of colour from light to dark intuitively represents smaller to higher population values. A sequential colour scheme suits this purpose well.



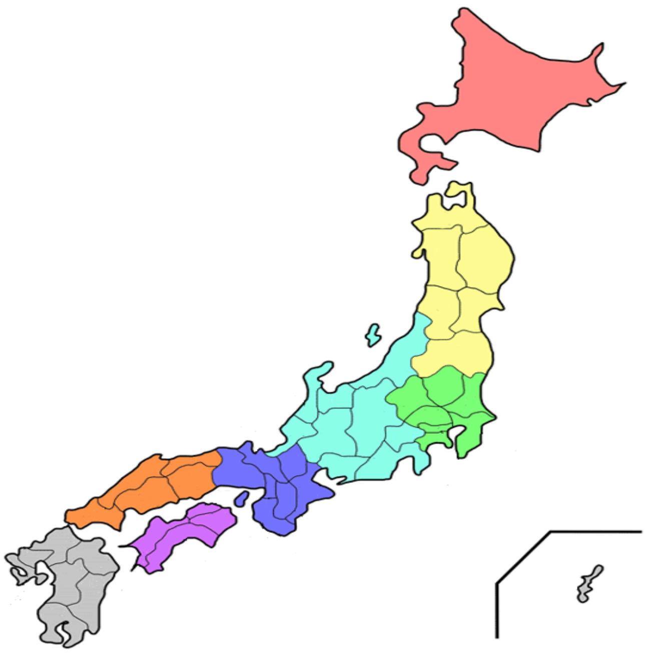
Diverging

Next is a map that represents what percentage of voters voted Democrat (blue) vs Republican (red) in the 2004 election. The image was taken from http://www.princeton.edu/~rvdb/JAVA/election2004/purple_america_2004_small.gif



As you can see there is lots of purple in this map, indicating regions where the vote was more evenly split. Divergent colour schemes normally have two dark, contrasting colours representing the divergent extremes with a light colour as the balance point to emphasize balanced regions.

Qualitative

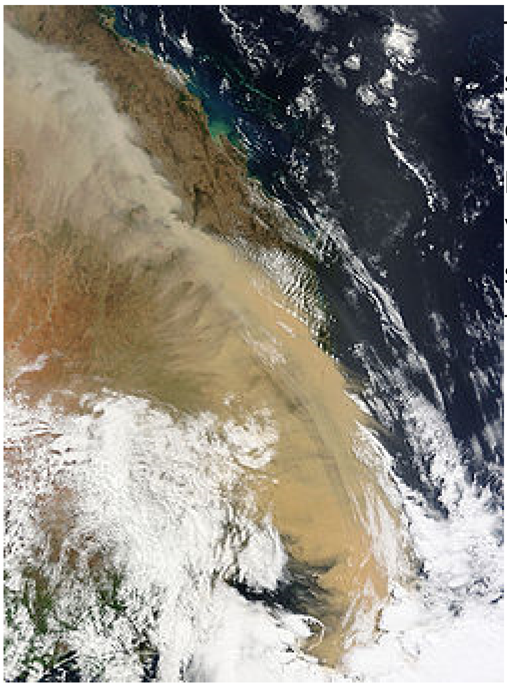


The map of Japan to the right is an example of a qualitative colour schema. This map differentiates the regions of japan by colour. The Kantō region is in green and the Kyūshū region is in gray. As the data represented is categorical, the different colours in the schema have no relation to each other. Regions are not official administrative boundaries but a traditional grouping of the 47 official prefectures of Japan. These regions are used instead of the prefectures in many contexts such as weather reporting. This map shows what prefectures are in what region.

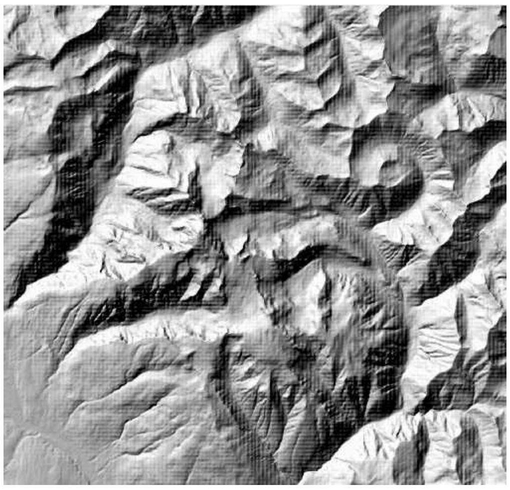
Looking at the alphabetical list of prefectures on Wikipedia (http://en.wikipedia.org/wiki/Prefectures_of_Japan) lists each prefecture with details such as population, area, density and so on. Various uses of this data set would benefit from one of the colour schemas described above based on what you are trying to communicate.

1.2.5Raster and Imagery

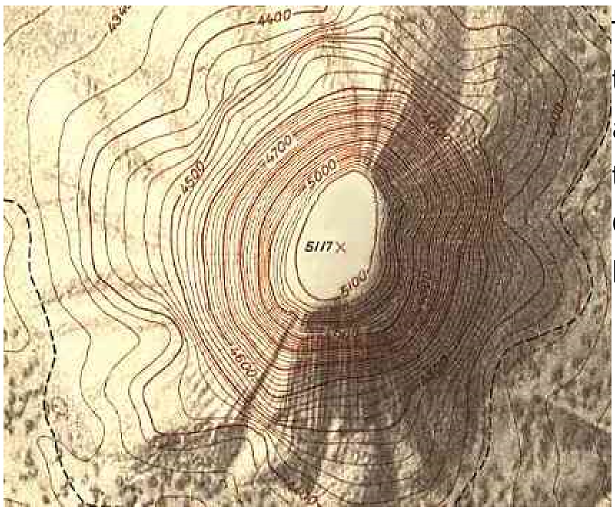
Satellite imagery can convey a great deal of useful information in a map. Other imagery used in raster maps include hillshade and topography.



This is a satellite image of the 2009 dust storm that swept across the eastern states of Australia in September. The dust storm is plainly visible from the satellite. If you look very closely you will see me (Jim Groffen) sitting in Coolangatta airport waiting for a flight back to Adelaide that never came.



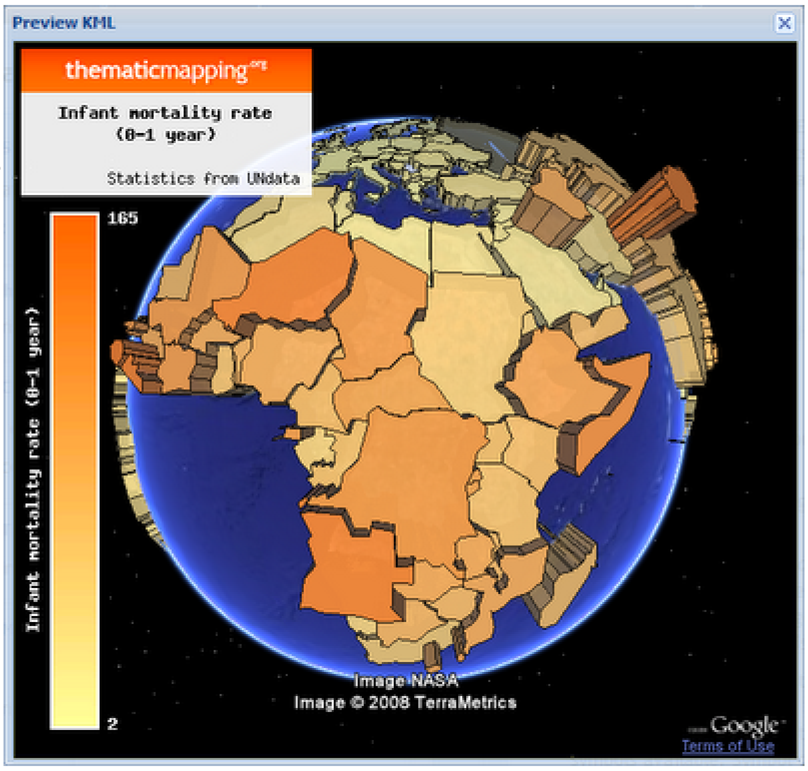
To the right is an example of hillshading. Hillshading generates the optical illusion of depth by applying relief shading to elevation information. It assumes the angle of a light source, darkening elevations that incline towards the light source and lightening elevations that incline away from the light.



Topography is the representation of the surface shape and features of an area. Hiillshading above is one way to convey topographic information. Contour lines depicted left is a hillshaded map with contour lines.

1.3 Misrepresentation

There are known issues when conveying information in a map that you should be aware of when considering style.


Preview KML
thematicmapping.org
Infant mortality rate (0-1 year)
Statistics from UNdata
165
2
Infant mortality rate (0-1 year)
Image NASA
Image © 2008 TerraMetrics
Google
Terms of Use

1.3.1The Atomic Fallacy

The 'Atomic' fallacy arises when you treat a geographic element, say a country, as a single, homogeneous whole element (an atom) ignoring the internal spatial variations within it. For example, if you consider France as a single homogeneous element you will miss the fact that in the South of France people drink much more wine than beer, whereas in the north-east of France, the relation is opposite.

1.3.2The Ecological Fallacy

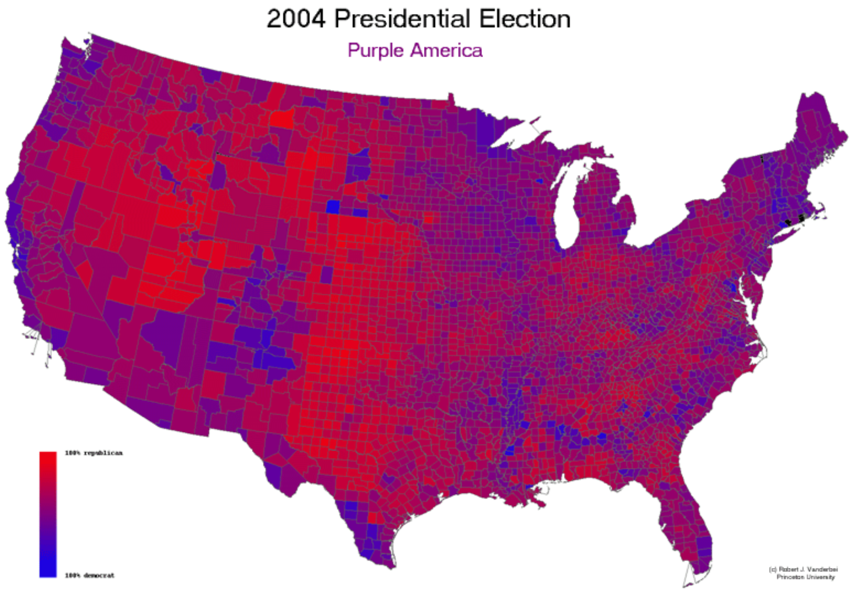
This fallacy is caused by an error in the interpretation of statistical data in an ecological study. This would arise if you considered all cities in France to drink the same average per-capita amount of beer and wine.

1.3.3The Modifiable Areal Unit Problem

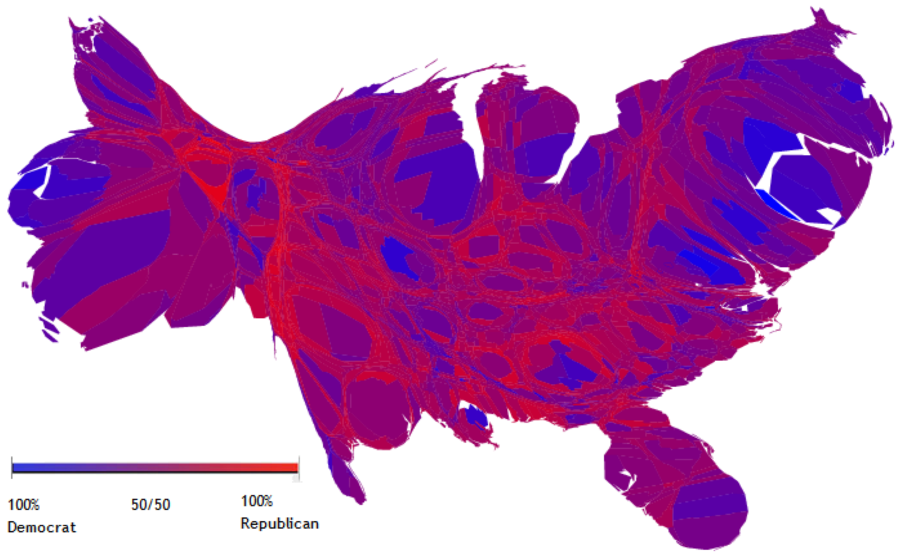
Tied closely to the ecological fallacy, a common cause for this error is that methods of collection of the statistical data may be driven by administrative boundaries or convenience and not reflect boundaries that are appropriate to the data collected.

Consider that drinking habits of information aggregated by voting district could be drastically different from information aggregated by postcode even if the two grouping methods have similar granularity.

For an example lets look at our map of the 2004 presidential election.



This map looks like a competition between Red and Purple, not Red and Blue. This is because there are many large counties with low populations that vote republican. One method for solving this problem is using a cartogram. Below is an example of the same election but the land areas have been adjusted to represent population.



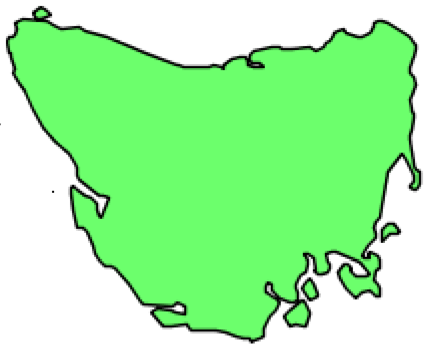
For more information, see:

- http://en.wikipedia.org/wiki/Spatial_analysis#Common_errors_in_spatial_analysis
- http://en.wikipedia.org/wiki/Ecological_fallacy.

2 Map Elements

Maps are visual representation of data that has a spatial component. The variety of data formats is staggering, especially in the spatial domain. Thankfully there is a standard terminology used when discussing spatial data that is used. This section discusses the common spatial types; what they are used for and how they can be styled.

2.1 Polygon



Polygons represent an area. Polygons are usually described by a list of straight line segments that chain together, bounding an area. Some polygons are conceptual only – such as administrative subdivisions. Other polygons represent an actual spatial feature such as the coastline of Tasmania.

Polygons are really two different stylable components in one – the line and fill. The fill of a polygon can be a solid colour, semi-transparent or something more complicated like a repeating pattern or gradient.

The border or edge of a polygon can be styled just as a line geometry can. The section on lines below pertains to a polygon edge.

Assigning colours to polygons based on measurements is known as theming or thematic mapping. How you theme is based on what attribute you are trying to communicate. An example would be colouring countries based on population density.

Polygons are also used to represent spatial features that make up parts of the map. In this case colours are defined according to what is represented (water = blue; land = green; city = gray).

2.2 Point

I would forgive you if you didn't realize the image to the right represents Hobart, the capital of Tasmania. Without any context a point on a map is not very useful.

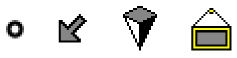
Points represent entities on a map that have a position but no dimensions. This may be because they are too small to have any relevant dimensions on the map.

Points on a map are represented by symbols or graphics. A common issue when selecting symbols is balancing the size and detail of the symbol. Larger symbols can be more detailed and convey a more complex topic, but they take up more map real estate. Smaller symbols must be simple to retain clarity.

Symbols should be easily discernible from other map elements. As a symbol is a graphic, a common technique is to incorporate a halo into the graphic.

Instead of different graphical symbols, a color code can be used to categorize the points. A combination of both graphical shape and colour is often employed with good results.

Perhaps various sizes of the same image is more appropriate for the concept you are trying to convey. For example, cities, towns and villages could all use the same symbol at different sizes. This can more intuitively convey the concept in some cases. Point size is used in thematic mapping as well. For instance a map of fixed speed camera locations could indicate how many speeders are caught at that camera by the size of the point.



The symbols to the left highlight another issue to do with using a graphic to represent a spatial location. A point at most map scales is just a single dot on a map. The graphic representing the point shows where the point is but because the graphic is going to be larger than one little dot on the map it needs to be clear from the graphic where that dot is. In the example the first one would put that dot right in the middle of the graphic where the arrow is at the bottom left point. This is known as the hot-spot of the graphic.

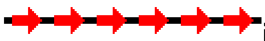
2.3 Line / Polygon Edge

Lines can represent actual landscape features. Examples are roads, walking tracks and rivers. Lines can also represent a conceptual feature but such as isolines or to represent traffic congestion on roads.

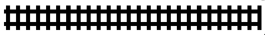
Line styling needs to consider the need to differentiate the line from the background of the map.

Roads are a good example of what can be done with line styling. The lines can be different widths or colours to represent road category – from thin pale minor roads to think, dark major roads.

Walking paths and tracks are often represented as a dashed line.



The image to the left is an example of a repeating graphic included in the drawing of a line. These graphics can turn with the line, rotating to follow the contour of the line.



An example of a more complicated concept when line drawing is train tracks. One way to achieve this is to use a single bar as a repeating graphic with a double line style.

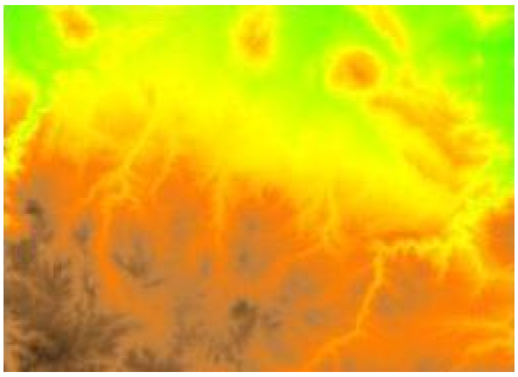
2.4 Raster

Points, lines and polygons are considered vector data. In the world of maps there is another category of spatial data – raster. In computer graphics, raster is a term for a data structure used to represent a grid of pixels or points of colour. In GIS, raster is also a grid of points, and while the points can be colour values (such as satellite imagery) the point values can also be some other useful value such as elevation, temperatures or rainfall.

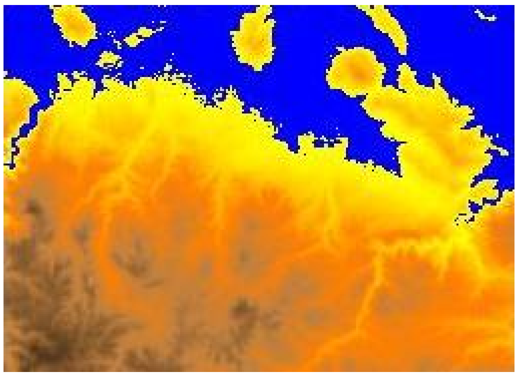
Visualizing raster data onto a map can seem pretty straightforward – you would expect that satellite imagery doesn't involve many styling options on it's own. Satellite imagery may not be of RGB imagery that is in the visible spectrum. Sensors used to collect the data may return 7 or more bands that need to be selected, and sometimes combined with a simple expression, to provide images that can be viewed. Sometimes a single band will be selected and depicted with a color map as if it was an elevation model.

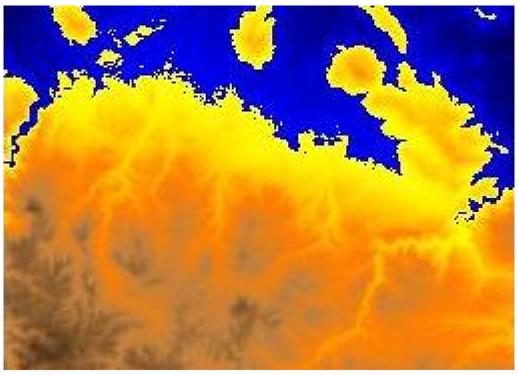
Dealing with satellite imagery in combination with overlay vector data involves ensuring contrast between the satellite imagery and the “drawn” geometries.

The image to the right is a visualization of raster data. The point values in the raster grid have been grouped into colours. What could the point values represent? In this case the raster is of a digital elevation model, green being low elevations and orange to black being higher elevations.

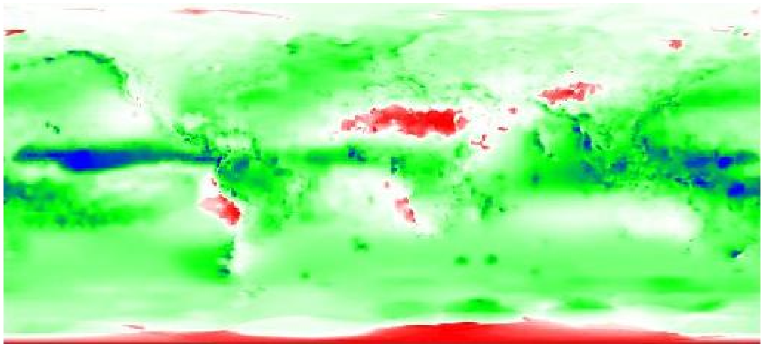


I've played with the styling used for this image to show what would happen if the oceans rose by 1200 meters for the above terrain. As you can see I've styled all terrain below the raised water level to be solid blue. I've included a second alternative on the right that shows the gradient of terrain below the new sea level.

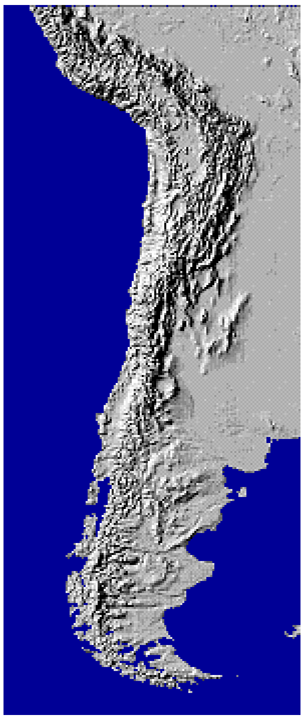




The image below shows rainfall levels for the world. Blue is high rainfall, green to white being average to low rainfall and red being low to no rainfall. From this image I can tell that the equator gets a high concentration of rainfall while Antarctica gets almost no rainfall.



The image to the right is an example of relief shading or hill shading. This image shows the elevation of the western coast of South America. The mountain range that separates Chile from Argentina is especially prominent, as it is a point of high elevation. Relief shading attempts to infer depth as a third dimension by having high elevations be dark on one side and light on the other to simulate a light source over a terrain.



When rendering raster information contrast enhancement is often used. Contrast enhancement will make the light colours lighter and the dark colours darker so that variations in the raster data is more pronounced visually.

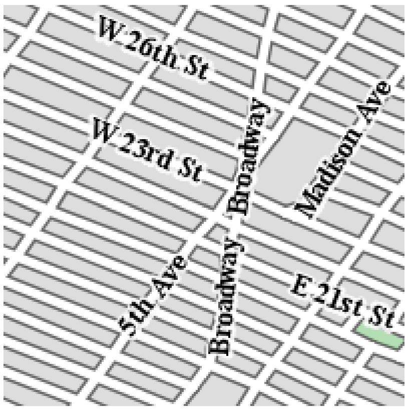
Further Reading:

http://en.wikipedia.org/wiki/False-color

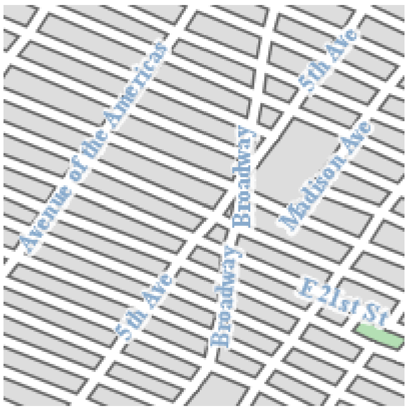
http://docs.codehaus.org/display/GEOTOOLS/Raster+Symbolizer+support

2.5 Text

Text often appears on maps as feature labels, values or used as graphical symbols. Lets discuss some of the considerations of text in maps.

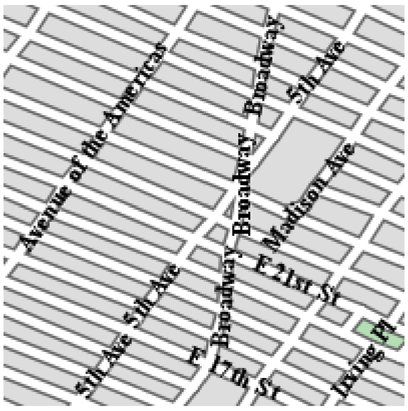


The map to the right is a small sample of the tiger-ny layer provided as an example with GeoServer. This layer shows road labels for some of the streets in Manhattan, New York.

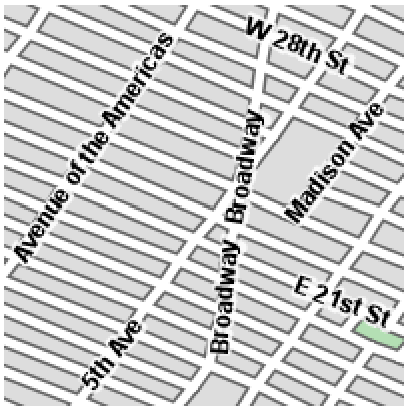


If we change the fill colour of the text to something less contrasting with the background we can see that the readability of the text is reduced.

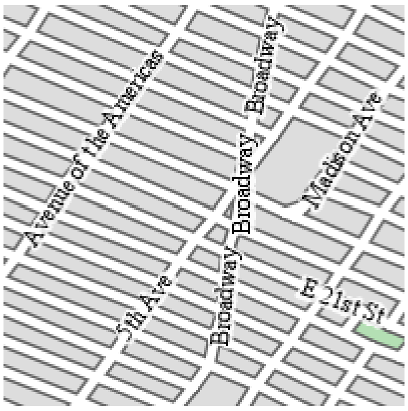
This time we have turned off label halos. When the fill colour of the text has a good contrast with the general map background. In the example here it is clear that a halo is required.



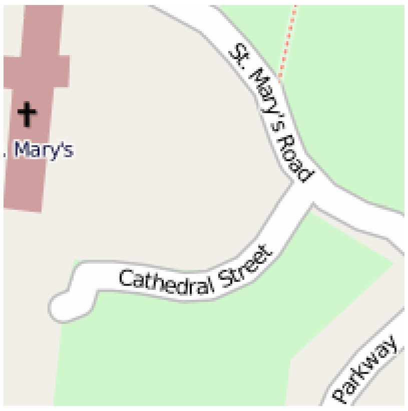
In this example we have switched to Arial font. Using fonts with no serifs (the dangly bits at the ends of the strokes of letters) often improve clarity of the map with minimal impact on aesthetics. A Sans-serif font means “no serifs”.



Turning off bold reduces the footprint of the font but clarity of the map is impacted.



This example from Open Street Map shows labels that follow the curve of the street. Labels that follow the curve of the line they are on is aesthetically pleasing but computationally expensive.



There is also a polygon representing a church building in this image. The cross symbol is incorporated to convey the church concept, and a label has also been applied to identify the building. This highlights the need to consider label positioning which In this case is underneath the cross symbol that is centered on the polygon.

2.6 Colours

Section 1.2.4 discussed colour schemes used to represent a data series on our map. There are other considerations for selecting colours.

Use of colour helps make a map as easy to interpret as possible by presenting the data in a colour scheme that matches the concept. Temperatures are blue for cold and red for hot. Mountains are brown, forests are green and swamps are an icky black/green colour with little pictures of reeds. Fire is red, flood is blue and money is green. Picking a colour that deviates from what culturally represents the colour requires people to think a little bit more than they would otherwise have to.

The choice between colour schemes for spatial distribution maps is usually an intuitive decision. Navigation maps also make use of many different colour schemes with less standards:

- Roads – styled and colour coded based on road type. As roads go from small to large in classification, a sequential colour scheme is often used for this.

- Map Feature Polygons – including parks, water features, airports, hospitals and so on. Green for parks, blue for water features and so on. This is a quantitative colour scheme that matches the concepts.

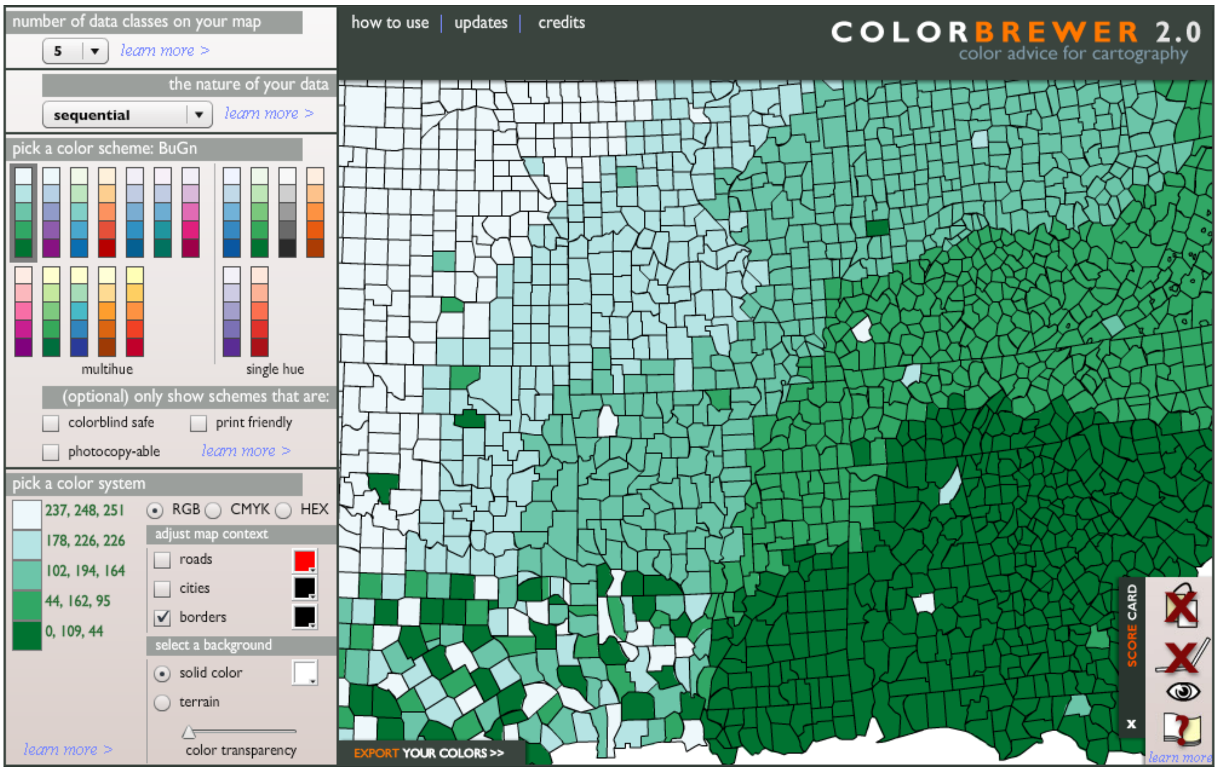
- Points of Interest – parking stations, petrol stations, restaurants, museums and monuments. A combination of symbol and colour can improve differentiation of the types of points of interest, again favoring a quantitative colour scheme.

It's best to pick colours that will render natively on your target medium. Devices may have a limited number of colours. How a colour scheme will look can vary between display devices. The device used to develop the colour scheme can often have a difficult time representing the colours as they will appear on the final medium. This is especially true for print.

Limiting the number of colours on a map improves both the file size and encoding time for image formats that benefit from a limited palette.

2.6.1Exercise

Assuming you are on a laptop with internet access, lets have a look at the colour brewer application. http://colorbrewer2.org/ - a useful tool for selecting colour schemes for maps.


number of data classes on your map: 5

the nature of your data: sequential

pick a color scheme: BuGn

(optional) only show schemes that are: ☐ colorblind safe ☐ print friendly ☐ photocopy-able

pick a color system: [✓ RGB] CMYK HEX

🟦	237, 248, 251
🟦	178, 226, 226
🟩	102, 194, 164
🟩	44, 162, 95
🟩	0, 109, 44

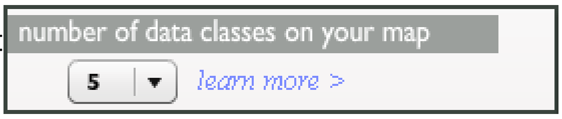
adjust map context: ☐ roads ☐ cities ☑ borders

select a background: [✓ solid color] terrain

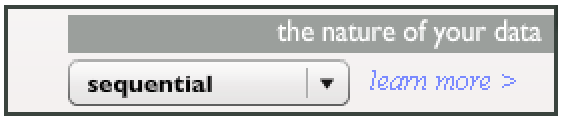
This web application explains itself very well throughout the “how to use” and “learn more >” links. Through these links, the functionality of the application as well as the colour scheme decision making process is explained. Read these links as we progress through the application.

As we step through the web application, keep a particular data set in mind and apply settings based on that data set. First click and read the “how to use” link above the map area.

set a number of classes on the right. Not all combinations of settings support many classes.

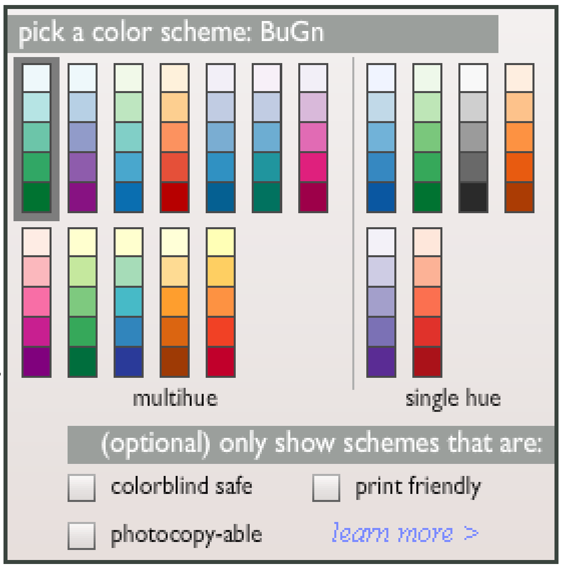

number of data classes on your map: 5

Set a nature – sequential, diverging or quantitative.

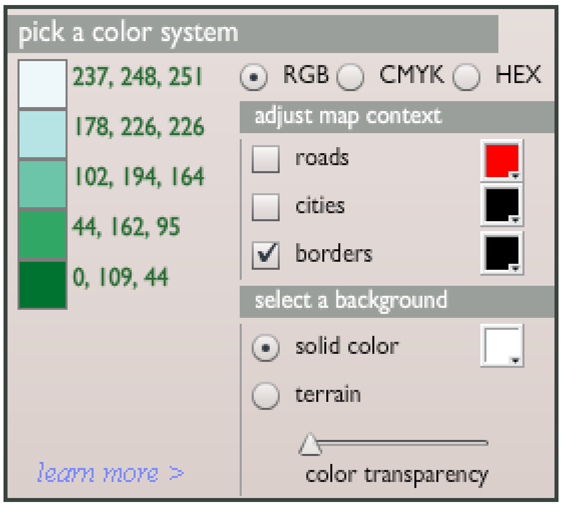

the nature of your data: sequential

Select a colour scheme that you like. The colour schemes presented are based on what nature you selected.

You can also restrict the presented colour schemes based on; colorblind safe, print friendly, photocopy-able. Doing this is likely to reduce the number of available colour schemes to zero quickly though.



Next, picking a colour system. The most common computer readable colour codes are supported – RGB, CMYK and HEX – HEX is useful for SLD as well as HTML and CSS but RGB and CMYK are also often used.



237, 248, 251
178, 226, 226
102, 194, 164
44, 162, 95
0, 109, 44

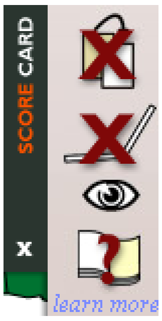
[✓ RGB] CMYK HEX

roads, cities, [✓ borders]

[✓ solid color] terrain

The map context and background controls let you adjust the sample map so you can check how you colour scheme is going to work with other map elements.

The scorecard shows how the current colour scheme scores against various usability considerations. The icons from top to bottom are:



- Photocopy-able
- Laptop (LCD) friendly
- Colourblind safe
- Colour printing friendly

Finally we can export the colour scheme. For manually crafting your SLD you will want to cut and paste HEX values.

3 Using Styles

So how should we handle map styling? First of all we don't want the styling controls for the map embedded in the map data. This would be very difficult to maintain. We would also like to be able to take the styles we have developed and apply them to different data. Lets take a look at a few analogous solutions to this problem.

HTML: To avoid having style information embedded throughout the HTML document itself CSS was developed. CSS stands for Cascading Style Sheet. A CSS file separates the styling information from the HTML file and can be applied to many different HTML files allowing a website of many pages to have on cohesive style applied to it.

MVC: MVC is a design pattern for software development. It stands for Model View Controller. The idea is to separate these three roles of an application. The Model is the information. The View is a representation of a model. Controllers perform actions on models and decide on a view to use to display models. This is another example of separation of the data from the styling or in this case the view.

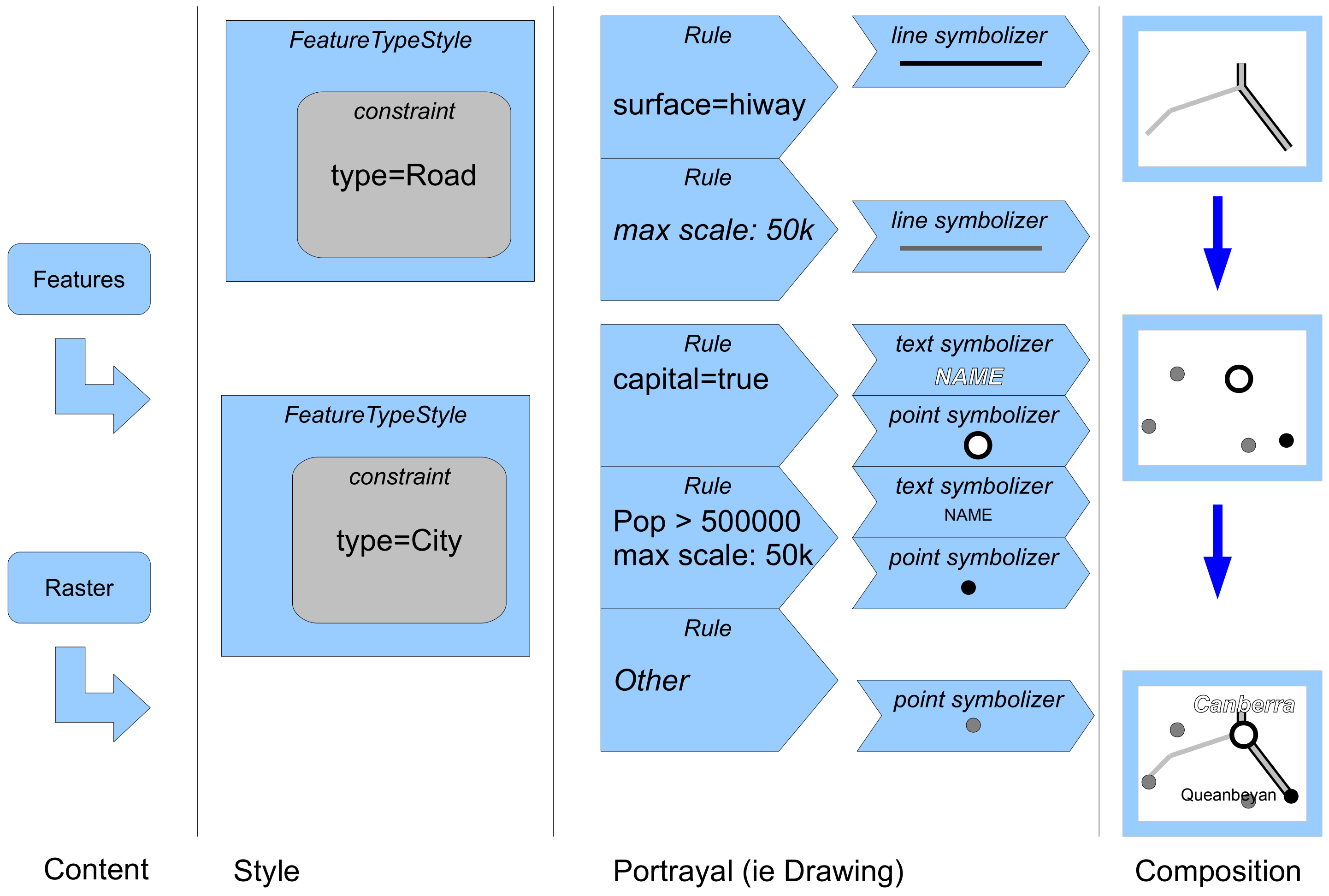
In the world of geospatial standards there is a standardized method of presenting map styling information separate to the map data. This is an OGC standard called SLD or Styled Layer Descriptor.

SLD lets us define a style separate to the data. Because support for SLD is available in most spatial software today we can take our styles and reuse them in other applications.

3.1 Introduction to SLD Concepts

The diagram below shows an example implementation of SLD rendering (in this case GeoTools). From left to right we have:

1. The content to style.
2. Linking styles to features with FeatureTypeStyle elements. The example below has two feature types that are going to be styled, roads and cities.
3. Rule elements control what features of a feature type get styled and when it is appropriate to apply that style.
4. Symbolizers contain the actual style information.
5. Composition is the task of layering the rendered features onto one map.



FeatureTypeStyle, Rule and Symbolizers are parts of an SLD document.

3.1.1NamedLayer, UserLayer, NamedStyle and UserStyle

An SLD is designed to be an almost full definition of a map.

NamedLayer is the name of a layer in the server. NamedStyle is the name of a styler registered in the server. UserLayer is an inline definition of the data in a layer using GML. UserStyle is an inline definition of a style using the SLD syntax itself.

When we just concentrate on defining a style, all we care about is defining UserStyle elements

The other elements still have a meaning that is useful. For instance they are used when making dynamic WMS calls in which the definition of the map is provided via a full fledged SLD document.

For the purpose of defining SLD that contain style information only we will follow a basic structure of an SLD document, here is a template to use:

```
<StyledLayerDescriptor … >
       <NamedLayer>
               <Name>name for style layer</Name>
               <UserStyle>
                       <Title>Title for style layer</Title>
                       <FeatureTypeStyle>
                               …
                       </FeatureTypeStyle>
               </UserStyle>
       </NamedLayer>
</StyledLayerDescriptor>
```

NamedLayer and UserStyle can have Name, Title and Abstract elements.

3.1.2FeatureTypeStyle

Every type of feature to be styled needs to be mentioned in the SLD. This is done with the FeatureTypeStyle element. This element contains one or more Rules that are processed to render the features.

How this rendering process occurs is not completely defined in the SLD specification, so there can be variation between implementations. What GeoServer and uDig assume is that each Rule is processed sequentially for each feature, whilst each FeatureTypeStyle are applied to all features before moving to the next. Some pseudo code that explains this:

```
for fts in FeatureTypeStyle
       for feature in data
               for rule in fts.rules
                       apply rule to feature
```

This separation allows the user to control inner layering and keep on using the more efficient approach for rules (since one does not need to rescan all data for each rule).

FeatureTypeStyle can also have the metadata elements Name, Title and Abstract. Other elements include FeatureTypeName: The name of the feature type to style. It is safer to omit this, allowing the software to make assumptions about what feature types the style applies to based on what symbolizers are specified.

3.2 Rules – What Gets Styled

Rules group rendering instructions (symbolizers) by a set of conditions that the features must meet for the rule to apply. Rules also let you specify some metadata as well- such as a machine friendly name, a title and an abstract that describes the rule.

Rules can also specify a LegendGraphic to use to specify a symbol to use to represent the rule in a map legend.

A rule is limited to describe only one feature type such as points. You can have more than one rule per FeatureTypeStyle though.

You can limit rules to apply to data only at certain scales using MinScaleDenominator and MaxScaleDenominator. Setting both limits the rule to apply to a specific range of scales, but you can specify only one of these if you want.

Scale in this context set what spatial area is represented by one pixel.

```
<MinScaleDenominator>400000</MinScaleDenominator>
```

In the example above we have set the minimum scale denominator to 1 pixel = 400 kilometers. Larger scales (that is, scales with smaller denominators) will not apply the rule.

```
<MaxScaleDenominator>1e6</MaxScaleDenominator>
```

This time we have set the maximum scale denominator to 1 pixel = 1 million metres, or 1000 kilometers. This is a smaller scale (1 / 1,000,000 is smaller than 1 / 400,000). Notice that larger scale denominators make for smaller scale maps.

OGC use a standardized rendering pixel size, defined to be 0.28mm × 0.28mm (millimeters). This is done because the true pixel size of the final rendering device is often unknown. Knowing the physical pixel size of the rendering device allows for a true scale to be calculated.

Another way to limit what data the rule applies to is by using a filter.

3.3 Filters

Filters allow you to control styling of features based on the geometry of the feature and feature attributes.

3.3.1 Spatial Filter

A spatial filter allows you to limit the features styled to ones that meet certain spatial criteria. The example below shows a filter that will only select features that intersect a certain point.

```
<Intersects>
	<PropertyName>GEOMETRY</PropertyName>
	<Literal>
		<gml:Point>
			<gml:coordinates>1 1</gml:coordinates>
		</gml:Point>
	</Literal>
</Intersects>
```

The Filter Encoding Specification lists many spatial filters including: BBOX, Beyond, Contains, Crosses, Disjoint, Distance, Equals, Intersects, Overlaps, Touches, Within and DWithin. Not all of these filters are supported by all software products.

3.3.2 Attribute Filter

Most spatial data is a more traditional table with rows of information where one column of this information is a geospatial type. This leaves all the other rows that we can use for filtering. Given the example Roads table:

name	Text
type	Text
oneway	Boolean – 0 or 1
maxspeed	Integer

We could apply the following complex filter that will select all roads that are not oneway, have a maximum speed limit of more than 60km/h and does interact (not disjoint) with a specified envelope:

```
<ogc:Filter>
	<ogc:PropertyIsGreaterThanOrEqualTo>
		<ogc:PropertyName>maxspeed<ogc:PropertyName>
		<ogc:Literal>80</ogc:Literal>
	</ogc:PropertyIsGreaterThanOrEqualTo>
</ogc:Filter>
```

Attribute filters defined in the Filter Encoding Specification include:

- PropertyIsEqualTo
- PropertyIsNotEqualTo
- PropertyIsLessThan
- PropertyIsLessThanOrEqualTo
- PropertyIsGreatherThan
- PropertyIsGreatherThanOrEqualTo
- PropertyIsBetween.
- PropertyIsLike
- PropertyIsNull

Use of attribute filters allows for some great styling capabilities, but has the drawback of tying the style definition to the data definition – as it is unlikely that two data sets will have matching attribute definitions or comparable values.

Again, not all property filters are supported by all software packages. There is generally less support for PropertyIsLike and PropertyIsNull.

3.3.3Logical Operators

Filters can be grouped under a logical operator. These are:

- And: All grouped filters must be true
- Or: Any one of the grouped filters must be true

- Not: All grouped filters must be false

You can't specify more than one filter without a logical operator to give context to how the two filters logically work together.

To continue on from our previous example. We could apply the following complex filter that will select all roads that are not oneway, have a maximum speed limit of more than 60km/h and interacts with (is not disjoint from) a specified envelope:

```
<ogc:Filter>
       <ogc:And>
              <ogc:PropertyIsEqualTo>
                     <ogc:PropertyName>oneway</ogc:PropertyName>
                     <ogc:Literal>0</ogc:Literal>
              </ogc:PropertyIsEqualTo>
              <ogc:PropertyIsGreaterThan>
                     <ogc:PropertyName>maxspeed</ogc:PropertyName>
                     <ogc:Literal>60</ogc:Literal>
              </ogc:PropertyIsGreaterThan>
              <ogc:Not>
                     <ogc:Disjoint>
                            <ogc:PropertyName>Geometry</ogc:PropertyName>
                            <gml:Envelope srsName="urn:x-ogc:def:crs:EPSG:6.3:4326">
                                   <gml:lowerCorner>138.0 -35.5</gml:lowerCorner>
                                   <gml:upperCorner>139.0 -34.5</gml:upperCorner>
                            </gml:Envelope>
                     </ogc:Disjoint>
              </ogc:Not>
       </ogc:And>
</ogc:Filter>
```

3.3.4Else Filter

A rule that applies an else filter will select all features that previous rules for this FeatureTypeStyle did not select.

```
<ogc:ElseFilter />
```

Using an ElseFilter anywhere except in the last rule of a FeatureTypeStyle with multiple rules is going to have unexpected behavior, or cause an error.

3.3.5Other Capabilities of Filtering

There are lots of other capabilities defined in the Filter Encoding specification – with limited levels of implementation in various spatial software packages. Have a look at http://www.opengeospatial.org/standards/filter for the official specification and check the documentation of your software package for support level.

GeoServer support of filter encoding is documented at:

http://docs.geoserver.org/1.7.x/en/user/styling/sld-reference/filters.html

or for version 2.0:

http://docs.geoserver.org/trunk/en/user/styling/sld-reference/filters.html

3.4 Symbolizers

Each SLD rule can specify multiple Symbolizers. Symbolizers state how to render the feature. There are different symbolizers for different feature types.

Only a brief summary is provided here. For a complete breakdown of the Symbolizers I would recommend our quick reference card or the GeoServer documentation:

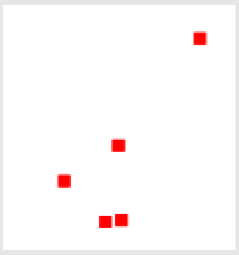
http://docs.geoserver.org/1.7.x/en/user/styling/sld-reference/index.html

http://docs.geoserver.org/trunk/en/user/styling/sld-reference/index.html

3.4.1Point Symbolizer

Below is a simple point symbolizer with some example output.

```
<PointSymbolizer>
  <Graphic>
    <Mark>
      <WellKnownName>square</WellKnownName>
      <Fill>
        <CssParameter name="fill">
           #FF0000
        </CssParameter>
      </Fill>
    </Mark>
    <Size>6</Size>
  </Graphic>
</PointSymbolizer>
```



Point symbolizers always have a Graphic element. Graphic elements describe how to graphically represent the point.

In the example above the Mark element lets you specify a common shape to use as the symbolizer. With mark we can also control fill and stroke.

Instead of Mark we can use ExternalGraphic. This lets you point to an image as a URL resource or a file resource with a path relative to the SLD.

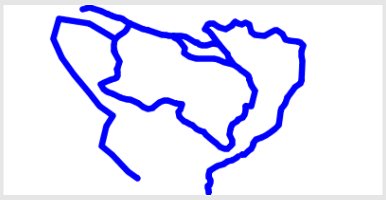
```
<PointSymbolizer>
  <Graphic>
    <ExternalGraphic>
      <!--  avoid hot linking, this is just a sample -->
      <OnlineResource xlink:type="simple"
         xlink:href="http://www.google.com/mapfiles/marker.png" />
      <Format>image/png</Format>
    </ExternalGraphic>
  </Graphic>
</PointSymbolizer>
```

Other attributes you can specify for a graphic is Opacity, Size and Rotation.

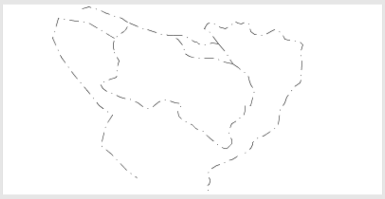
3.4.2Line Symbolizer

Line symbolizer is little more than a Stroke element. Stroke elements support repeating graphics as outlined in 2.3 with GraphicFill and GraphicStroke elements. These elements use a Graphic element as per the Point symbolizer.

```
<LineSymbolizer>
  <Stroke>
    <CssParameter name="stroke">#0000FF</CssParameter>
    <CssParameter name="stroke-width">6</CssParameter>
    <CssParameter name="stroke-linejoin">round</CssParameter>
    <CssParameter name="stroke-linecap">round</CssParameter>
  </Stroke>
</LineSymbolizer>
```



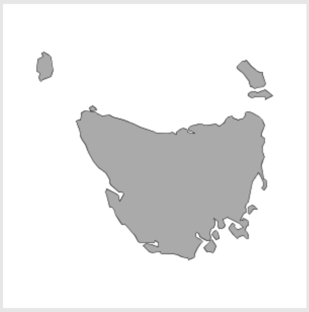
```
<LineSymbolizer>
  <Stroke>
    <CssParameter name="stroke">
       #000000
    </CssParameter>
    <CssParameter name="stroke-dasharray">
        10 5 1 5
    </CssParameter>
  </Stroke>
</LineSymbolizer>
```



3.4.3 Polygon Symbolizer

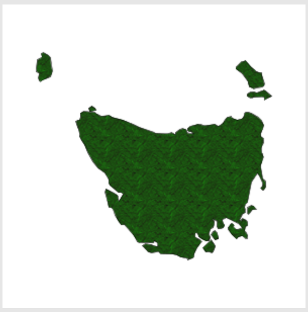
Styling a polygon is a combination of styling the fill and the polygon edge as a stroke.

```
<PolygonSymbolizer>
  <Fill>
    <CssParameter name="fill">
       #AAAAAA
    </CssParameter>
  </Fill>
  <Stroke>
    <CssParameter name="stroke">
       #000000
    </CssParameter>
    <CssParameter name="stroke-width">1</CssParameter>
  </Stroke>
</PolygonSymbolizer>
```



Polygons can have a GraphicFill element as part of it's Fill element.

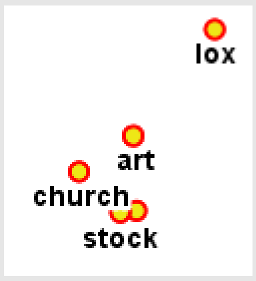
```
<PolygonSymbolizer>
  <Fill>
    <GraphicFill>
      <Graphic>
        <ExternalGraphic>
          <OnlineResource xlink:type="simple"
                          xlink:href="grass_fill.png" />
          <Format>image/png</Format>
        </ExternalGraphic>
      </Graphic>
    </GraphicFill>
  </Fill>
  <Stroke />
</PolygonSymbolizer>
```



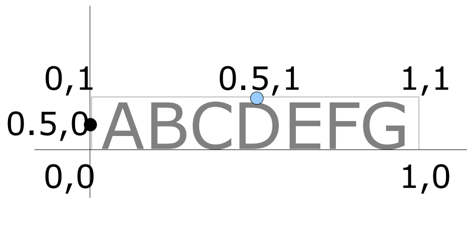
3.4.4Text Symbolizer

Text symbolizers are applied in the same rule as other symbolizers because they apply text from an attribute of a feature to a map based on the location of the feature. The attribute to use for the text is specified using the Label element. Positioning of the text is controlled by the LabelPlacement element and an outline to the text can be applied using the Halo element. Here is an example:

```
<TextSymbolizer>
    <Label>
      <ogc:PropertyName>NAME</ogc:PropertyName>
    </Label>
    <Font>
      <CssParameter name="font-family">Arial</CssParameter>
      <CssParameter name="font-weight">Bold</CssParameter>
      <CssParameter name="font-size">14</CssParameter>
    </Font>
    <LabelPlacement>
      <PointPlacement>
        <AnchorPoint>
          <AnchorPointX>0.5</AnchorPointX>
          <AnchorPointY>0.5</AnchorPointY>
        </AnchorPoint>
        <Displacement>
          <DisplacementX>0</DisplacementX>
          <DisplacementY>-15</DisplacementY>
        </Displacement>
      </PointPlacement>
    </LabelPlacement>
    <Halo>
      <Radius>
        <ogc:Literal>2</ogc:Literal>
      </Radius>
      <Fill>
        <CssParameter name="fill">#FFFFFF</CssParameter>
      </Fill>
    </Halo>
    <Fill>
      <CssParameter name="fill">#000000</CssParameter>
    </Fill>
</TextSymbolizer>
```



Anchor points are represented as a ratio of the width and height of the text. The legend below is a guide to using the AnchorPoint element.

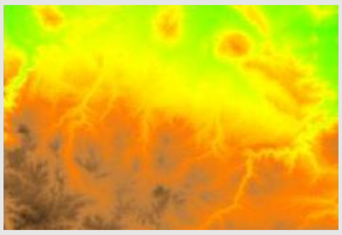


Displacement is in pixels from the spatial element labelled. In the example we displace on the Y axis by -15 pixels.

3.4.5Raster Symbolizer

There are extensive options when visualizing raster data. Lets start with an example of a colour map:

```
<RasterSymbolizer>
  <Opacity>1.0</Opacity>
  <ColorMap>
    <ColorMapEntry color="#AAFFAA" quantity="0" label="values" />
    <ColorMapEntry color="#00FF00" quantity="1000" label="values"/>
    <ColorMapEntry color="#FFFF00" quantity="1200" label="values" />
    <ColorMapEntry color="#FF7F00" quantity="1400" label="values" />
    <ColorMapEntry color="#BF7F3F" quantity="1600" label="values" />
    <ColorMapEntry color="#000000" quantity="2000" label="values" />
  </ColorMap>
</RasterSymbolizer>
```

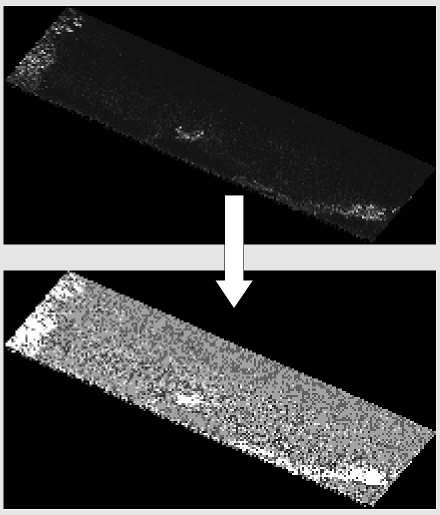


The raster symbolizer uses a colour map to scale elevation values to a series of colours. The colour map will gradient between the colours based on the colour value.

Another raster element is ChannelSelection. Some raster data sets contain channel colour information. With ChannelSelection we can control the mapping of the channels to red, green, blue or gray channels. We can also use ChannelSelection in combination with ColorMap to recolour an RGB image using the extended attribute.

The ContrastEnhancement element allows you to adjust relative brightness of the data in a colour channel using one of three enhancement methods; Normalize, Histogram and GammaValue.

```
<RasterSymbolizer>
  <ChannelSelection>
    <GrayChannel>
      <SourceChannelName>1</SourceChannelName>
      <ContrastEnhancement>
        <Histogram/>
      </ContrastEnhancement>
    </GrayChannel>
  </ChannelSelection>
  <ContrastEnhancement>
    <GammaValue>1</GammaValue>
  </ContrastEnhancement>
</RasterSymbolizer>
```



Less supported raster symbolization elements include:

- ShadedRelief to apply hill shading.
- OverlapBehavior to control how multiple raster layers interact in spatial regions where they overlap on the same map.
- ImageOutline to allow each raster layer in a composite of multiple raster layers to have an outline highlighting the bounds of the raster data.

3.4.6 Common Symbolizer Elements

Many of the SLD Elements that control rendering are reused for many symbolizers. Fill, Graphic and Stroke are the best examples of this.

Fill: on a polygon controls rendering of the interior area of a polygon. In a Mark element it controls the primary colour of the mark. On a text symbolizer fill controls the colour of the text. It is also used to style a halo.

Graphic: can be applied to a stroke for a repeating graphic in the fill of the stroke to give the stroke a pattern, or to repeat on top of the stroke. They can be applied to a polygon as a repeating graphic in the fill area of a polygon.

Stroke: used for line symbolizers and in polygon symbolizers as the polygon border. Stroke is also used for the Mark element of a Graphic.

3.5 SLD Software

We will take a look at the SLD support available in various open source GIS software packages.

3.5.1 OpenLayers

OpenLayers is a JavaScript library that makes it easy to put a dynamic map in any web page. OpenLayers currently supports a limited subset of the SLD specification. OpenLayers does this by providing support for SLD format files, allowing OpenLayers to read SLD documents into a JavaScript object model. This object model can then be used to interact with other OpenLayers objects.

3.5.2GeoServer

GeoServer supports WMS and WFS implementation profiles of SLD. GeoServer manages a list of styles that can be applied to layers by default or when a map request is made. GeoServer also supports applying an SLD to a map as part of the GetMap request.

http://docs.geoserver.org/1.7.x/en/user/styling/index.html

http://docs.geoserver.org/trunk/en/user/styling/index.html

3.5.3GeoServer GeoExt SLD Editor Plugin

GeoExt Styler is a plugin for GeoServer that provides a graphical user interface for managing the styles of layers in GeoServer. GeoExt uses OpenLayers and GeoExt

The following link covers installation of the GeoExt Styler plugin:

http://docs.geoserver.org/trunk/en/user/extensions/styler.html

GeoExt builds on OpenLayers using Ext JS to provide desktop GIS style UI in a browser. For more information on GeoExt:

http://www.geoext.org/

3.5.4MapServer

MapServer is an Open Source OGC standards based spatial web publishing platform. MapServer supports a configuration file for controlling map and style configuration called a .map file. MapServer provides extensive tools for working with .map files, converting SLD to a map file and generating SLD from a map file, applying a supplied SLD with a getmap request, serving style information via the map service as SLD.

For a comprehensive list of the support level of SLD in MapServer including a breakdown of symbolizer support, take a look at:

http://mapserver.org/ogc/sld.html

Also check out the MapServer site:

http://mapserver.org/

3.5.5OpenJump

OpenJump is a Java based desktop GIS application. OpenJump manages styling of spatial data in it's own way but supports SLD by importing SLD files. The following link has extensive details on the OpenJump SLD support: http://www.openjump.org/wiki/show/Import-Export+Layer+Style.

3.5.6uDig

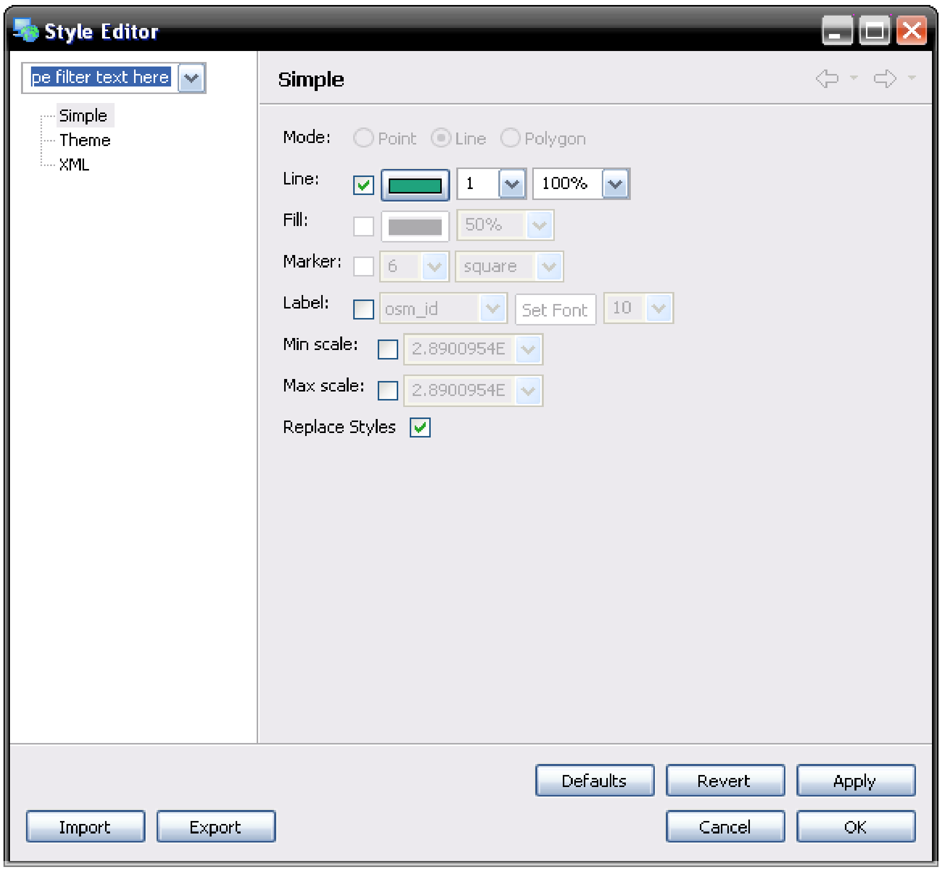
uDig offers import and export of SLD documents as well as SLD editing capabilities via the Style Editor Dialog and the Style view.

http://udig.refractions.net/confluence/display/EN/Style+Layer+Descriptor

http://udig.refractions.net/confluence/display/EN/Style+Editor+Dialog

From the Style Editor Dialog we can import and export SLD. To get to it, right click on a layer in the Layers view and select “Change style...”, or select “Change style...” from the layer menu with a layer selected.

uDig has provided colorbrewer style functionality in the Style Editor Dialog. To try this out, select “Theme” from the list on the left.



Mode: Point [✓ Line] Polygon
Line: ✓ 🟩 1 100%
Fill: ☐ 50%
Marker: ☐ 6 square
Label: ☐ osm_id Set Font 10
Min scale: ☐ 2.8900954E
Max scale: ☐ 2.8900954E
Replace Styles ✓

4 Hands On

4.1 OpenLayers

There are two live examples of OpenLayers SLD capabilities.

Example 1: Live SLD Switching

http://openlayers.org/dev/examples/sld.html

In this example an SLD with numerous named user styles are loaded from this link: http://openlayers.org/dev/examples/tasmania/sld-tasmania.xml. The test page then allows the user to dynamically switch between the user styles. If you are familiar with JavaScript, look at the source and see how the options set the style applied to the layer.

Example 2: WMS POST with SLD Included

http://openlayers.org/dev/examples/WMSPost.html

This example makes use of the ability of a WMS service to be sent an SLD along with the request. This SLD can then be used by the WMS service to render the requested layers.

4.2 uDig

Lets use uDig to make a style. This hands-on assumes you are running from the LiveDVD but if you aren't you should be able to follow along with your own uDig installation and some of your own test data.

1. Open uDig using the shortcut on the desktop.
2. From the “Layer” menu select “Add...”
3. Select “Files” and press “Next >”.

4. You should be looking at your user folder. If not select user from the “Places” list. Then select the “data” folder. Scroll down and select “udig-data”.

5. Find a file called “usa_counties.shp” and double click it. It may take a while but the shape file will be loaded as a layer of uDig.

6. Right click on the layer in the layers list on the left and select “Change Style...”. The Style Editor Dialog will appear.

7. The dialog will show you style options for a polygon symbolizer. We can have a play with these for a while.

8. Now we will try a theme. Show the Theme dialog by ensuring “Theme” is selected in the style editor dialog.

9. Change the first option “Attribute” to “HOUSEHOLDS”.

10. Change “Classes” to “6”.

11. Select a Palette. Pick any palette you prefer.

12. Notice the “Break” option. This sets whether to break the colour bands by equal intervals or to balance the quantity of features that fall into each band. Notice how the “Values” list changes when you change this.

13. Lets try what we have set so far. Hit “Apply” and move the dialog out of the way so we can see what we have done. Try applying a few different palettes. I found that increasing Opacity to 100% looked better than leaving it at 50%.

14. Given our discussion on types of colour schemes you should know that we probably want a sequential colour scheme. Set the Palette box to only show sequential colour schemes.

15. I found that the outline setting was being ignored. Lets have a look at the XML and see why. On the left select “XML”.

16. It looks like the stroke symbolizer is repeated for each colourband. We want that rule to apply to all styles. If you are uncomfortable with XML editing then you may want to skip this step.

Add a new rule to the FeatureTypeStyle that has no filter conditions to apply a stroke to all features. Change this rule to draw a gray stroke with a hex colour value of #888888. You will then need to remove all other stroke elements.

17. We can now cut and paste the XML into our own file or hit the “Export” button. Save this in your user folder as “counties.sld” This should be the default location.

4.3 GeoServer

Lets try taking both the data and style into GeoServer.

1. Run the “Admin GeoServer” shortcut on the desktop.

2. Click the Config link. You will be asked to log in. Use “admin/geoserver”.

3. Create the style. Click on “Data” and then “Style”. Then click “New”.

4. Set a style ID of “county_poly” and click “New”.

5. Click the “Browse” button. You will see a dialog appear showing your user folder. Select the SLD we saved earlier from uDig and click “Open”. Then click “Upload”. Click “Submit”

6. Next create the layer. Click the “Data” link. Then click “DataStores”.

7. Click “New”. Select “Shapefile” and then type “counties” as the “Feature Data Set ID”.

8. In the “url:” field type: “file:/home/user/data/udig-data/usa_counties.shp”

9. Leave all the other options as default / blank and click Submit.

10. The usa_counties FeatureType will automatically be found by GeoServer. Change the “Style: “ to “county_poly”. This tells GeoServer to apply the “county_poly” style to this layer by default.

11. Find the “Generate” button next to the text “Bounding Box” and click it. The bounding box coordinates should pop up pretty quickly.

12. Scroll to the very bottom of the page and click “Submit”.

13. To apply these changes to GeoServer, click the “Apply” button on the far left and then click the “Save” button.

14. Check that this all worked by clicking on the “Welcome” link then “Demo”, then “Map Preview”. Find “topp:usa_counties” and click “OpenLayers”. We can now see the same style being used by GeoServer.

4.4 GeoExt Styler

If there is time, browse to:

http://localhost:8082/geoserver/www/styler/index.html

Use the Layers selector on the left to select the usa_counties_Type. Have a play with the GeoExt styler by:

- Interacting with the map.
- Left clicking on features to inspect them.
- Left clicking on items in the legend to look at the style editor dialog.
- Try turning on labels.
- Look at other GeoServer layers to see what other symbolizer options are supported by GeoExt.

5 Advanced styling

5.1 Thematic maps with a twist

Making thematic maps is an exercise in classification. Usually the thematic map is based on numeric value intervals or uses a one to one match between a string class and colors.

However in some cases the classification one needs to achieve is based on just too many classes. The CFCC class attribute in the old Tiger shapefiles can assume tens of values, sometimes the classification required is just too coarse for the fine grained nature of the attribute.

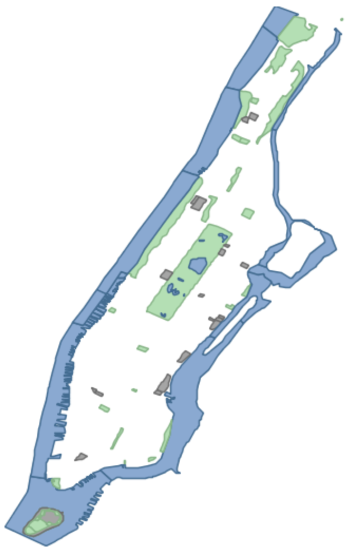
While pre-processing the data and get a better classification is certainly a path worth considering, if the dataset is not too big using creative means of classification can be useful.

Let's assume we want to classify the landmarks polygon file in just three classes, green areas, water and urban. Some inspection reveals all green areas are classified between D81 and D89, all water related areas start with H and the urban areas are those that remain once green and water areas are removed.

In this case the following classification, based on three simple rules, works out nicely (poly_landmarks1.sld):

```
     <FeatureTypeStyle>
        <!-- park and green spaces -->
        <Rule>
          <ogc:Filter>
            <!--  D84, D85, D86, ... -->
            <ogc:PropertyIsLike wildCard="%" singleChar="_" escape="\">
              <ogc:PropertyName>CFCC</ogc:PropertyName>
              <ogc:Literal>D8%</ogc:Literal>
            </ogc:PropertyIsLike>
          </ogc:Filter>
          <PolygonSymbolizer>
            <Fill>
              <CssParameter name="fill">#B4DFB4</CssParameter>
            </Fill>
            <Stroke>
              <CssParameter name="stroke">#88B588</CssParameter>
            </Stroke>
          </PolygonSymbolizer>
        </Rule>

        <!-- water -->
        <Rule>
          <ogc:Filter>
          <ogc:PropertyIsLike wildCard="%" singleChar="_" escape="\">
            <ogc:PropertyName>CFCC</ogc:PropertyName>
            <ogc:Literal>H%</ogc:Literal>
          </ogc:PropertyIsLike>
        </ogc:Filter>

        <PolygonSymbolizer>
          <Fill>
            <CssParameter name="fill">#8AA9D1</CssParameter>
          </Fill>
          <Stroke>
            <CssParameter name="stroke">#436C91</CssParameter>
          </Stroke>
        </PolygonSymbolizer>
      </Rule>

      <!-- urban   -->
      <Rule>
        <ElseFilter />

        <PolygonSymbolizer>
          <Fill>
            <CssParameter name="fill">#A5A5A5</CssParameter>
          </Fill>
          <Stroke>
            <CssParameter name="stroke">#6E6E6E</CssParameter>
          </Stroke>
        </PolygonSymbolizer>
      </Rule>
    </FeatureTypeStyle>
```

The first two classes are classified using a like filter, the first catches whatever starts by “D8”, the second all classes starting by “H”. The third rule uses an ElseFilter, which will activate any time one of the previous rules did not capture the polygon to be drawn.



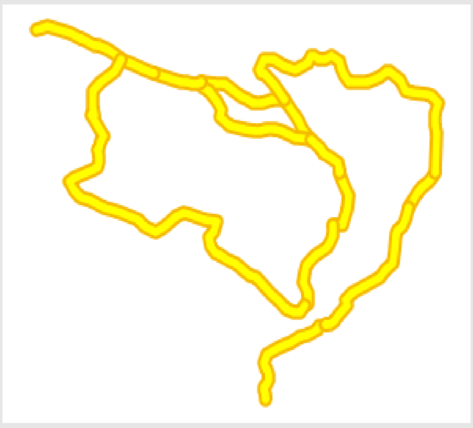
5.2 Stacking multiple symbolizers

Quite a few common styles require the user to stack multiple symbolizers one on top of the other to get interesting visual effects. A simple example, demonstrated by the GeoServer **poi** (points of interest) style is to have a composite symbol, made out of two superimposed marks, a red circle and a yellow smaller circle (poi1.sld):

```
      <FeatureTypeStyle>
       <Rule>
         <PointSymbolizer>
           <Graphic>
             <Mark>
               <WellKnownName>circle</WellKnownName>
               <Fill>
                 <CssParameter name="fill">#FF0000</CssParameter>
               </Fill>
             </Mark>
             <Size>11</Size>
           </Graphic>
         </PointSymbolizer>
         <PointSymbolizer>
           <Graphic>
             <Mark>
               <WellKnownName>circle</WellKnownName>
               <Fill>
                 <CssParameter name="fill">#EDE513</CssParameter>
               </Fill>
             </Mark>
             <Size>7</Size>
           </Graphic>
         </PointSymbolizer>
       </Rule>
      </FeatureTypeStyle>
```

A similar effect is often desired for road networks, where one wants to generate highway like symbols by stacking two lines of different widths and colors one on top of the other. Here is an example based on the Tasmania roads layer (highway1.sld):

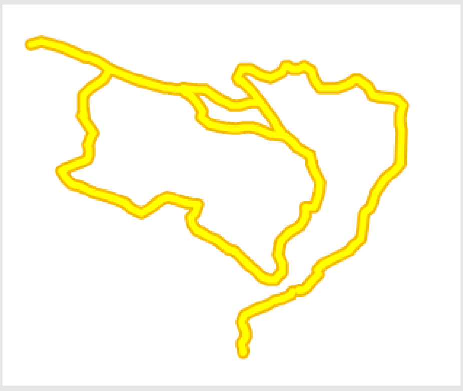
```
    <FeatureTypeStyle>
        <Rule>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#F5B800</CssParameter>
              <CssParameter name="stroke-width">8</CssParameter>
              <CssParameter name="stroke-linejoin">round</CssParameter>
              <CssParameter name="stroke-linecap">round</CssParameter>
            </Stroke>
          </LineSymbolizer>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#FFFF00</CssParameter>
              <CssParameter name="stroke-width">4</CssParameter>
              <CssParameter name="stroke-linejoin">round</CssParameter>
              <CssParameter name="stroke-linecap">round</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>
     </FeatureTypeStyle>
```



As you can see from the sample output there is a problem when lines do join. This is happening because each line is painted first with the first symbolizer, and then the second, and then the next line is started.

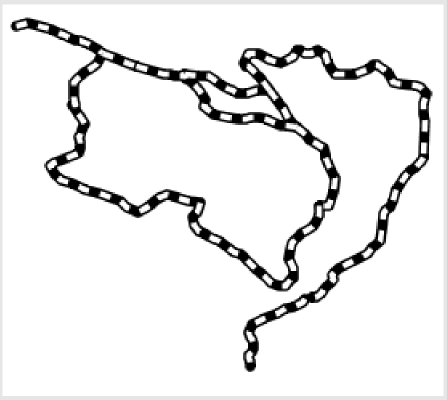
A way to ensure all thick lines are painted before the thin ones is to put the rules into two separate FeatureTypeStyle blocks, as FeatureTypeStyle work as inner layers[1] (highway2.sld):

```
      <FeatureTypeStyle>
        <Rule>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#F5B800</CssParameter>
              <CssParameter name="stroke-width">8</CssParameter>
              <CssParameter name="stroke-linejoin">round</CssParameter>
              <CssParameter name="stroke-linecap">round</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>

      </FeatureTypeStyle>
      <FeatureTypeStyle>
        <Rule>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#FFFF00</CssParameter>
              <CssParameter name="stroke-width">4</CssParameter>
              <CssParameter name="stroke-linejoin">round</CssParameter>
              <CssParameter name="stroke-linecap">round</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>
      </FeatureTypeStyle>
```



With a similar approach, and using dash arrays, it's possible to produce a simple railroad symbol (railroad.sld):

```
<FeatureTypeStyle>
        <Rule>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#000000</CssParameter>

              <CssParameter name="stroke-width">6</CssParameter>
              <CssParameter name="stroke-linejoin">round</CssParameter>
              <CssParameter name="stroke-linecap">round</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>
      </FeatureTypeStyle>
      <FeatureTypeStyle>
        <Rule>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#FFFFFF</CssParameter>
              <CssParameter name="stroke-width">3</CssParameter>
              <CssParameter name="stroke-linejoin">round</CssParameter>
              <CssParameter name="stroke-dasharray">7 7</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>
       </FeatureTypeStyle>
```



[1]The SLD specification does not give clear guidance on how to approach multi-rule or multi symbolizer stacking, different software might implement it in a different way. So, with some software you could get proper stacking even with the first sample SLD.

5.3 Working with scale dependencies

As discussed in previous sections a crowded map is an unreadable one. There is also an obvious performance problem, as drawing too much data takes time. SLD provides ways to enable certain rules only in given scale ranges, that is useful in various cases:

- showing only a few features when fairly zoomed out
- reserve alternate, better looking, but heavier styling to detailed maps
- enabling labels only when fairly zoomed in to avoid getting an overly crowded map

5.3.1 Adding labels in a scale range

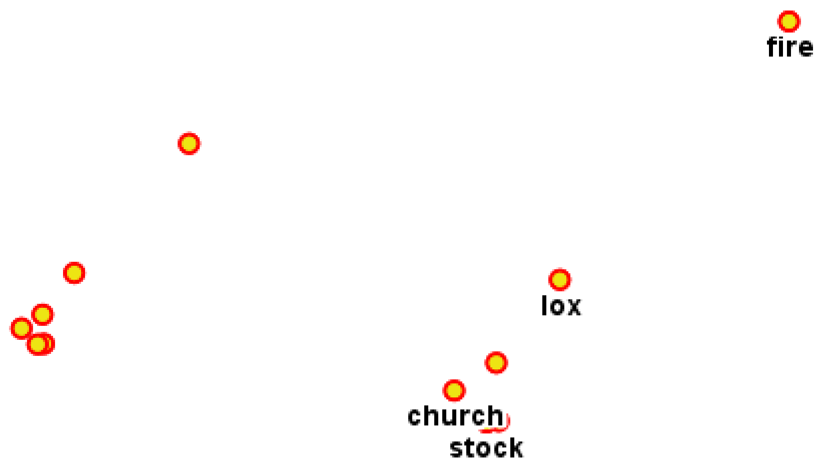
Let's have a look at a fist example. Using the same Point Of Interest layer provided in the previous examples we want to show a label, but only when fairly zoomed in, when the various points are represented far apart enough on the map (poi_2.sld):

```
      <FeatureTypeStyle>
       <Rule>
         <PointSymbolizer>
           <Graphic>
             <Mark>
               <WellKnownName>circle</WellKnownName>
               <Fill>
                 <CssParameter name="fill">#FF0000</CssParameter>
               </Fill>
             </Mark>
             <Size>11</Size>
           </Graphic>
         </PointSymbolizer>
         <PointSymbolizer>
           <Graphic>
             <Mark>
               <WellKnownName>circle</WellKnownName>
               <Fill>
                 <CssParameter name="fill">#EDE513</CssParameter>
               </Fill>
             </Mark>
             <Size>7</Size>
           </Graphic>
         </PointSymbolizer>
       </Rule>
       <Rule>
         <MaxScaleDenominator>32000</MaxScaleDenominator>
         <TextSymbolizer>
           <Label>
             <ogc:PropertyName>NAME</ogc:PropertyName>
           </Label>
           <Font>
             <CssParameter name="font-family">Arial</CssParameter>
             <CssParameter name="font-weight">Bold</CssParameter>
             <CssParameter name="font-size">14</CssParameter>
           </Font>
           <LabelPlacement>
             <PointPlacement>
```

```
                <AnchorPoint>
                  <AnchorPointX>0.5</AnchorPointX>
                  <AnchorPointY>0.5</AnchorPointY>
                </AnchorPoint>
                <Displacement>
                  <DisplacementX>0</DisplacementX>
                  <DisplacementY>-15</DisplacementY>
                </Displacement>
              </PointPlacement>
            </LabelPlacement>
            <Halo>
              <Radius>
                <ogc:Literal>2</ogc:Literal>
              </Radius>
              <Fill>
                <CssParameter name="fill">#FFFFFF</CssParameter>
              </Fill>
            </Halo>
            <Fill>
              <CssParameter name="fill">#000000</CssParameter>
            </Fill>
          </TextSymbolizer>
        </Rule>
      </FeatureTypeStyle>
```

The style builds on top of poi1.sld, but adds a new Rule that will display only when the scale denominator is below 32000. The label is also using various standard TextSymbolizer features:

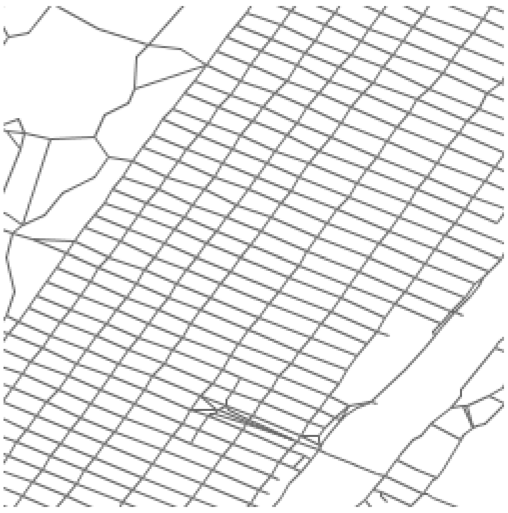
- it's centered relative to the point (the default would be to show it on the left of it) using AnchorPoint
- it's displaced vertically so that it does not overlap with the two circles using Displacement
- uses an Halo to make the label read over a busy background

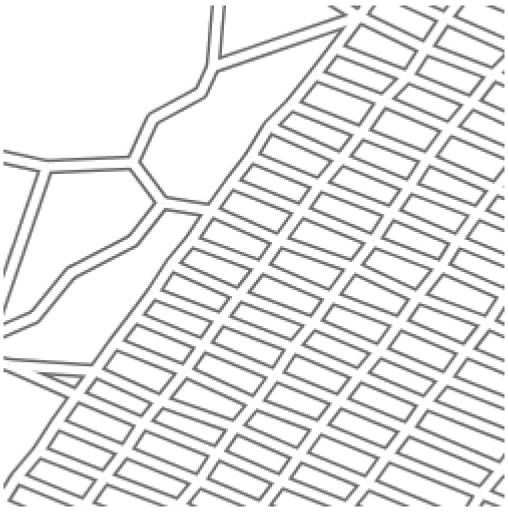


5.3.2Alternate styling based on the zoom level

As a second example let's see how it's possible to have alternate styling based on the scale. We want to show a road network with thin line when zoomed out (below 1:32000), and with a cased line, gray outside, white inside, when zoomed in (above 1:32000). The following SLD does the trick by mixing scale dependencies and multilayer symbol stacking (road1.sld):

```
      <FeatureTypeStyle>
        <Rule> <!-- thin line only at lower scales -->
          <MinScaleDenominator>32000</MinScaleDenominator>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#666666</CssParameter>
              <CssParameter name="stroke-width">0.5</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>
        <Rule><!-- thick line drawn first-->
          <MaxScaleDenominator>32000</MaxScaleDenominator>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#666666</CssParameter>
              <CssParameter name="stroke-width">7</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>
      </FeatureTypeStyle>
      <FeatureTypeStyle>
        <FeatureTypeName>Feature</FeatureTypeName>
        <Rule><!-- thin line drawn second -->
          <MaxScaleDenominator>32000</MaxScaleDenominator>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#FFFFFF</CssParameter>
              <CssParameter name="stroke-width">4</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>
      </FeatureTypeStyle>
```





The two images show the same layer at 1:38000 scale (first image) and at a 1:19000 scale (second image). As you can see the second image is setup so that it's possible to add also labels inside the road area.

5.4 Advanced labeling

SLD misses a few features that are necessary for good map labeling. In particular:

- While it's possible to align a label along with a road using LinePlacement, the spec does not say if the label should be curved or straight
- If multiple segments do have the same label they will be labeled separately, It's also not possible to specify whether to repeat labels over long lines.
- There is no way to specify the maximum width of a label, or how to auto-wrap it in case that is desirable.

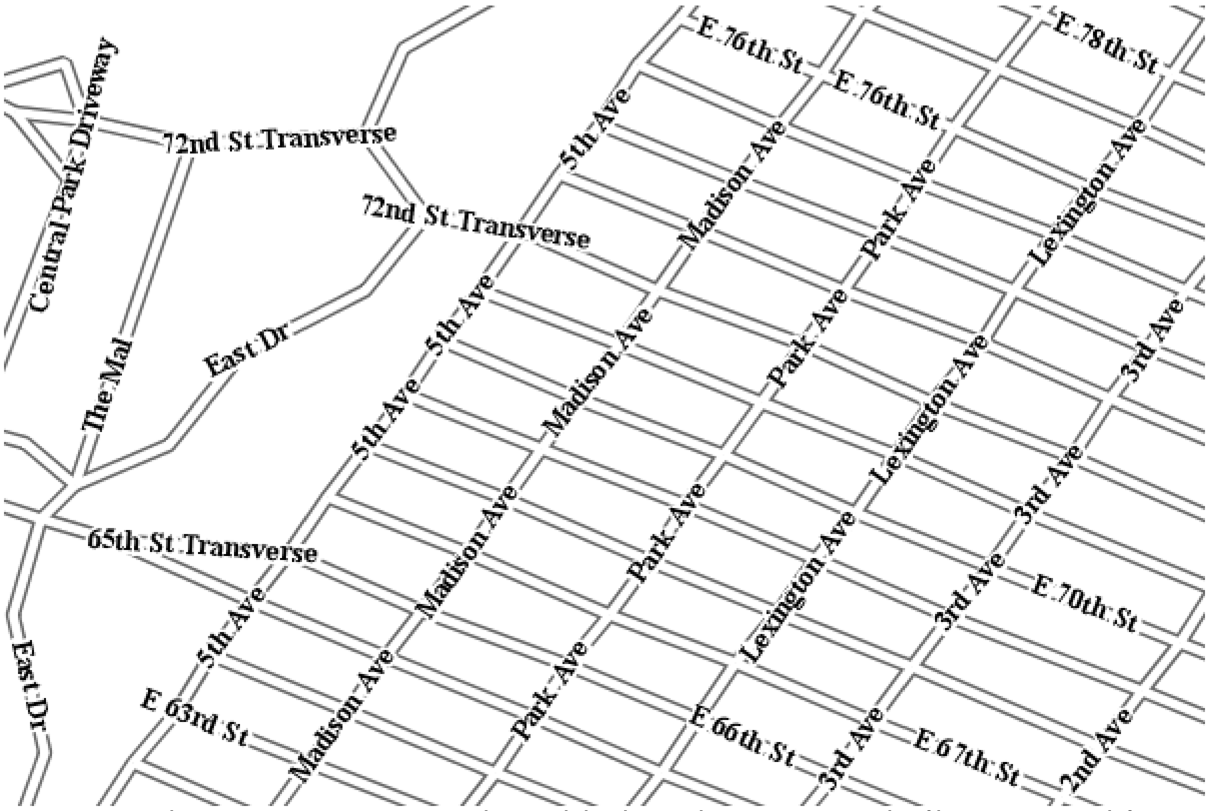
GeoServer and uDig can use an extended, non standard SLD syntax to deal with the above problems, among others.

5.4.1 Curved labels, grouping and repetitions

To deal with curved labels and label repetition let's add labels to the Manhattan road layer we worked on before. The second FeatureTypeStyle now will contain a TextSymbolizer (roads2.sld):

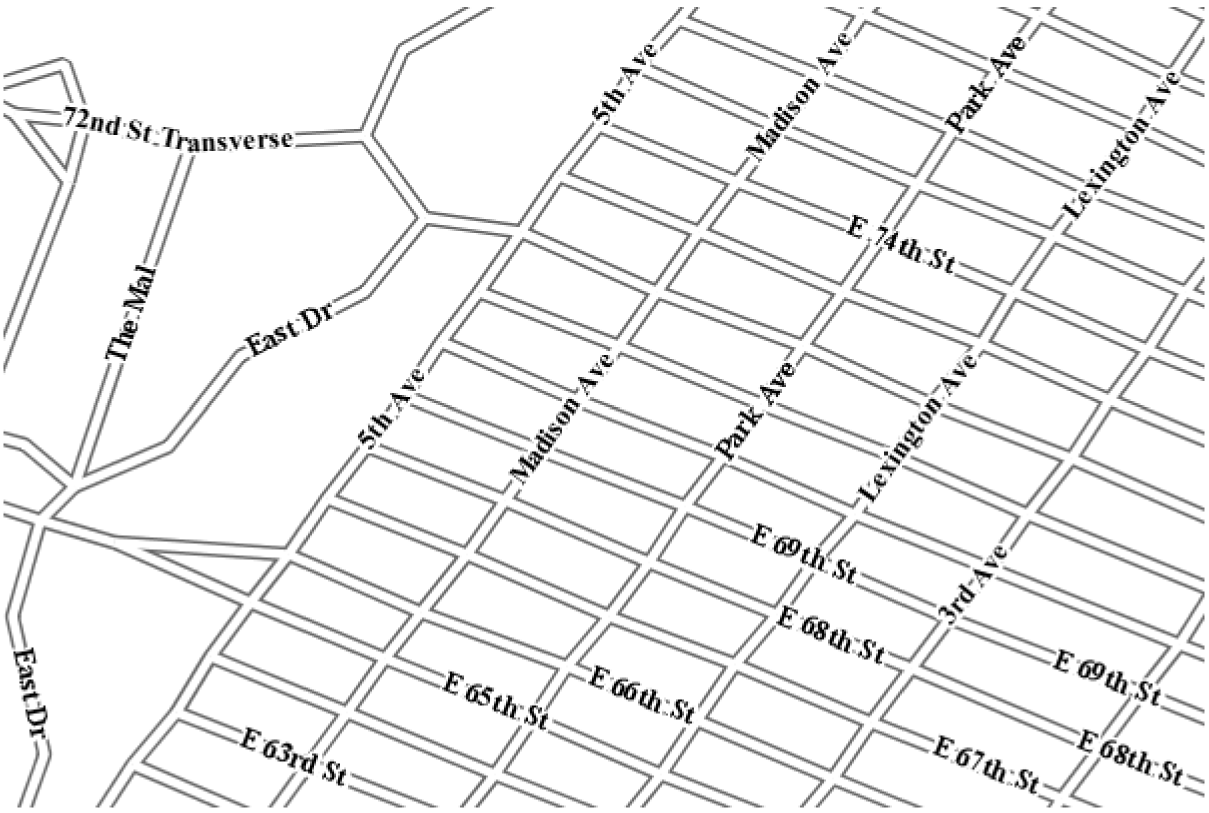
```
<FeatureTypeStyle>
... <!-- First feature type style, skipped for brevity -->
</FeatureTypeStyle>
<FeatureTypeStyle>
  <FeatureTypeName>Feature</FeatureTypeName>
  <Rule><!-- thin line drawn second -->
    <MaxScaleDenominator>32000</MaxScaleDenominator>
    <LineSymbolizer>
      <Stroke>
        <CssParameter name="stroke">#FFFFFF</CssParameter>
        <CssParameter name="stroke-width">4</CssParameter>
      </Stroke>
    </LineSymbolizer>
    <TextSymbolizer>
      <Label><ogc:PropertyName>NAME</ogc:PropertyName></Label>
      <Font>
        <CssParameter name="font-family">Times New Roman</CssParameter>
        <CssParameter name="font-style">Normal</CssParameter>
        <CssParameter name="font-size">14</CssParameter>
        <CssParameter name="font-weight">bold</CssParameter>
      </Font>
      <LabelPlacement>
        <LinePlacement>
        </LinePlacement>
      </LabelPlacement>
      <Halo>
        <Radius>
          <ogc:Literal>2</ogc:Literal>
        </Radius>
        <Fill>
          <CssParameter name="fill">#FFFFFF</CssParameter>
        </Fill>
      </Halo>
    </TextSymbolizer>
  </Rule>
</FeatureTypeStyle>
```

The result is labeled, but has issues with some labels along curved streets, and it's easy to notice the avenues are labeled too often whilst the streets crossing them not enough:



Some vendor parameters can be added to the TextSymbolizer to enable curved labels, to group lines with the same labels into a single, longer line, and to control label repetition (roads3):

```
<TextSymbolizer>
    <Label><ogc:PropertyName>NAME</ogc:PropertyName></Label>
    <Font>
      <CssParameter name="font-family">Times New Roman</CssParameter>
      <CssParameter name="font-style">Normal</CssParameter>
      <CssParameter name="font-size">14</CssParameter>
      <CssParameter name="font-weight">bold</CssParameter>
    </Font>
    <LabelPlacement>
      <LinePlacement>
      </LinePlacement>
    </LabelPlacement>
    <Halo>
      <Radius>
        <ogc:Literal>2</ogc:Literal>
      </Radius>
      <Fill>
        <CssParameter name="fill">#FFFFFF</CssParameter>
      </Fill>
    </Halo>
    <VendorOption name="followLine">true</VendorOption>
    <VendorOption name="group">true</VendorOption>
    <VendorOption name="repeat">200</VendorOption>
    <VendorOption name="maxDisplacement">50</VendorOption>
</TextSymbolizer>
```



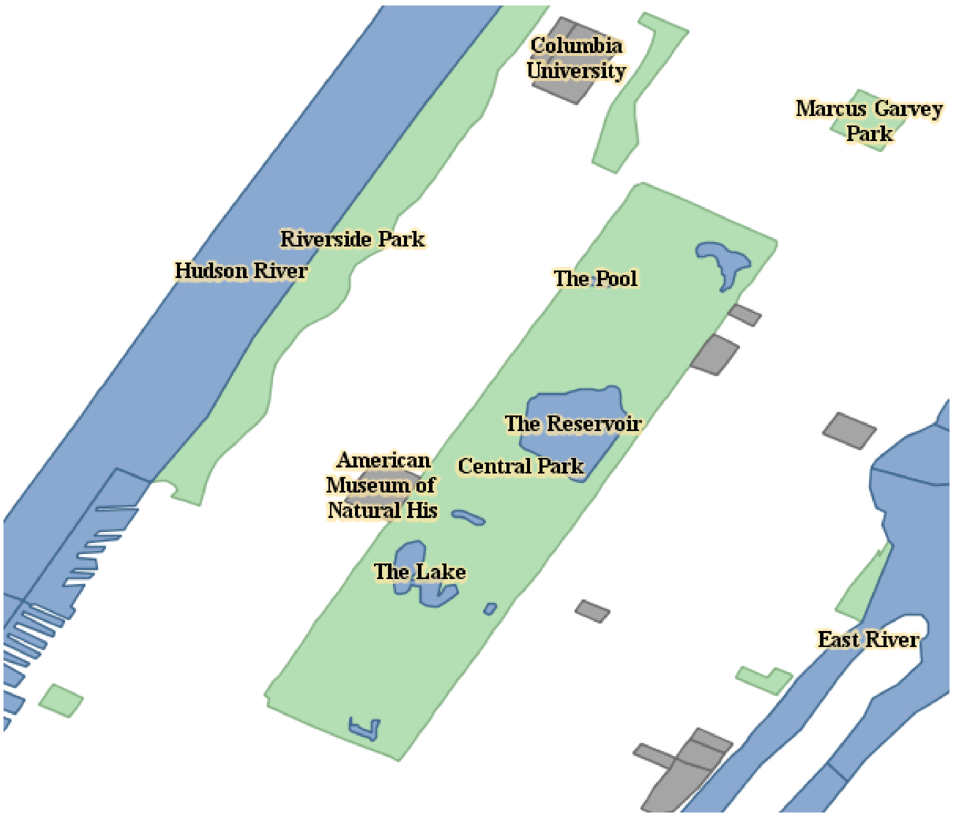
5.4.2 Auto wrapping labels

The landmarks layer used in the thematic map has attributes that can be used for labeling. GeoServer will automatically avoid labeling the feature is the label is too big compared the the polygon being labeled, but still, a significant number of those labels are too long: a way to wrap them is needed.

One way to approach this problem is to pre-process the data and insert hard newlines into the data. Alternatively, recent versions of GeoServer have a vendor parameter that can auto-wrap labels whose length is excessive. The following style snipped, built on top of landmarks1.sld, add a text symbolizer with such option (landmarks2.sld):

```
      <FeatureTypeStyle>
        <Rule>
          <TextSymbolizer>
            <Label>
              <ogc:PropertyName>LANAME</ogc:PropertyName>
            </Label>
            <Font>
              <CssParameter name="font-family">Times New Roman
              </CssParameter>
              <CssParameter name="font-style">Normal</CssParameter>
              <CssParameter name="font-size">14</CssParameter>
              <CssParameter name="font-weight">bold</CssParameter>
            </Font>
            <LabelPlacement>
              <PointPlacement>
                <AnchorPoint>
                  <AnchorPointX>0.5</AnchorPointX>
                  <AnchorPointY>0.5</AnchorPointY>
                </AnchorPoint>
              </PointPlacement>
            </LabelPlacement>
            <Halo>
              <Radius>
                <ogc:Literal>2</ogc:Literal>
              </Radius>
              <Fill>
                <CssParameter name="fill">#FDE5A5</CssParameter>
                <CssParameter name="fill-opacity">0.75</CssParameter>
              </Fill>
            </Halo>
            <Fill>
              <CssParameter name="fill">#000000</CssParameter>
            </Fill>
            <VendorOption name="group">true</VendorOption>
            <VendorOption name="autoWrap">100</VendorOption>
          </TextSymbolizer>
        </Rule>
      </FeatureTypeStyle>
```

The style makes GeoServer wrap labels at 100 pixels width automatically. It also makes it group the polygons so that a block of them with the same label will be labeled just once:

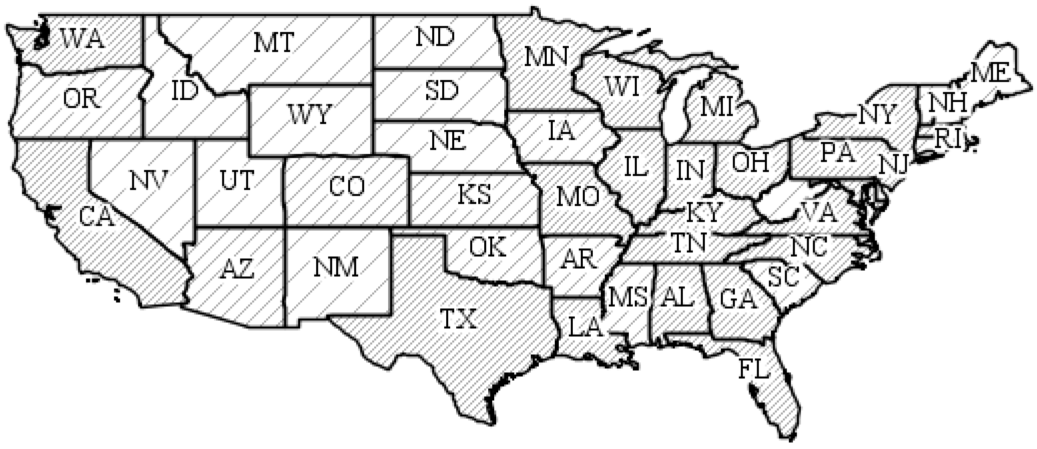


5.5 Thematic mapping with hatch density

Thematic mapping is usually performed using color hues or shades. Yet, it's also possible to communicate numerical progression by graphical density, from looser to thicker, using hatches.

Hatches are not a first level citizen in SLD, they can be produced by creating an external graphic that tiles nicely. In GeoServer and recent version of uDig it's also possible to use a custom Mark name, *shape://slash*. The advantage in using a Mark is that one can control pattern density, stroke thickness and color without the need to create many image samples.

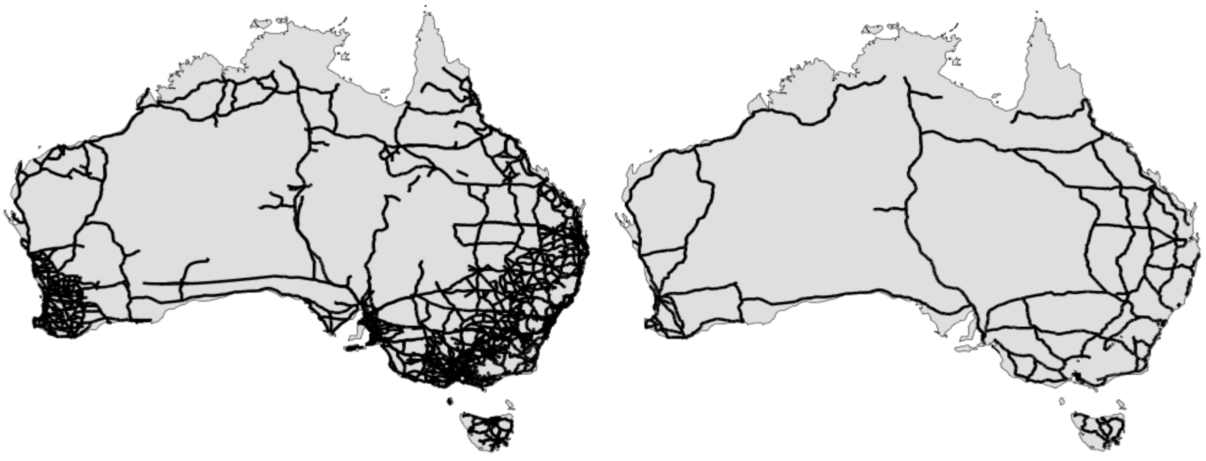
```
      <FeatureTypeStyle>
        <Rule>
          <Title>&lt; 2M</Title>
          <ogc:Filter>
            <ogc:PropertyIsLessThan>
              <ogc:PropertyName>PERSONS</ogc:PropertyName>
              <ogc:Literal>2000000</ogc:Literal>
            </ogc:PropertyIsLessThan>
          </ogc:Filter>
          <PolygonSymbolizer>
            <Fill>
              <GraphicFill>
                <Graphic>
                  <Mark>
                    <WellKnownName>shape://slash</WellKnownName>
                    <Stroke>
                      <CssParameter name="stroke">0xAAAAAA</CssParameter>
                    </Stroke>
                  </Mark>
                  <Size>16</Size>
                </Graphic>
              </GraphicFill>
            </Fill>
          </PolygonSymbolizer>
        </Rule>
        <Rule>
          <Title>2M - 4M</Title>
          <ogc:Filter>
            <ogc:PropertyIsBetween>
              <ogc:PropertyName>PERSONS</ogc:PropertyName>
              <ogc:LowerBoundary>
                <ogc:Literal>2000000</ogc:Literal>
              </ogc:LowerBoundary>
              <ogc:UpperBoundary>
                <ogc:Literal>4000000</ogc:Literal>
              </ogc:UpperBoundary>
            </ogc:PropertyIsBetween>
          </ogc:Filter>
          <PolygonSymbolizer>
            <Fill>
              <GraphicFill>
                <Graphic>
                  <Mark>
                    <WellKnownName>shape://slash</WellKnownName>
                    <Stroke>
                      <CssParameter name="stroke">0xAAAAAA</CssParameter>
                    </Stroke>
                  </Mark>
                  <Size>8</Size>
                </Graphic>
              </GraphicFill>
            </Fill>
          </PolygonSymbolizer>
        </Rule>
        <Rule>
          <Title>&gt; 4M</Title>
          <!-- like a linesymbolizer but with a fill too -->
          <ogc:Filter>
            <ogc:PropertyIsGreaterThan>
              <ogc:PropertyName>PERSONS</ogc:PropertyName>
              <ogc:Literal>4000000</ogc:Literal>
            </ogc:PropertyIsGreaterThan>
          </ogc:Filter>
          <PolygonSymbolizer>
            <Fill>
              <GraphicFill>
                <Graphic>
                  <Mark>
                    <WellKnownName>shape://slash</WellKnownName>
                    <Stroke>
                      <CssParameter name="stroke">0xAAAAAA</CssParameter>
                    </Stroke>
                  </Mark>
                  <Size>4</Size>
                </Graphic>
              </GraphicFill>
            </Fill>
          </PolygonSymbolizer>
        </Rule>
        <Rule>
          <Title>Boundary</Title>
          <LineSymbolizer>
            <Stroke />
          </LineSymbolizer>
          <TextSymbolizer>
            <Label>
              <ogc:PropertyName>STATE_ABBR</ogc:PropertyName>
            </Label>
            <Font>
              <CssParameter name="font-family">Times New Roman
              </CssParameter>
              <CssParameter name="font-style">Normal</CssParameter>
              <CssParameter name="font-size">14</CssParameter>
            </Font>
            <LabelPlacement>
              <PointPlacement>
                <AnchorPoint>
                  <AnchorPointX>0.5</AnchorPointX>
                  <AnchorPointY>0.5</AnchorPointY>
                </AnchorPoint>
              </PointPlacement>
            </LabelPlacement>
            <Halo>
              <Radius>2</Radius>
              <Fill>
                <CssParameter name="fill">0xFFFFFF</CssParameter>
              </Fill>
            </Halo>
          </TextSymbolizer>
        </Rule>
      </FeatureTypeStyle>
```



6 Performance Considerations

6.1 Draw Less

One of the things that often slows down rendering is simply drawing more information then is needed. Taking the road dataset and drawing all the roads does not make a lot of sense when zoomed really far out (there is more detail then is useful). Additionally, small polygons may actually not be visible at all.



You ca reduce this to only show the main “trunk” roads using a small bit of Style Layer Descriptor syntax; adding a “filter” to the Rule used to display roads.

```
            <ogc:Filter>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>highway</ogc:PropertyName>
                    <ogc:Literal>trunk</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Filter>
```

Another thing you can do is choose what zoom level a Rule will use; in the example above we can have the “trunk” roads draw when the user is zoom in to 1:10,000,000 level.

```
            <sld:MaxScaleDenominator>1.0E7</sld:MaxScaleDenominator>
```

6.2 Swap between Layers for the Same Information

You can combine the above two techniques; declare two layers against the same dataset:

- One layer with a Style that draws at a lower level of detail when zoomed out
- Another layer with a Style that draws everything when zoomed in

You can also group these two layers and present them as a single WMS layer. This results in a seamless user experience while still letting you manage information at different levels of details.

Alternative; create a Style with two rules.

6.3 Avoid Expensive Styling Options

The amount of control provided by the Style Layer Descriptor format is staggering allowing you to accomplish many amazing effects. Some of these styling options are more expensive in performance terms than others.

- **Transparency:** Working with any transparency at all will basically double your rendering time.

  Make sure that each symbolizer you use has opacity set to 1.

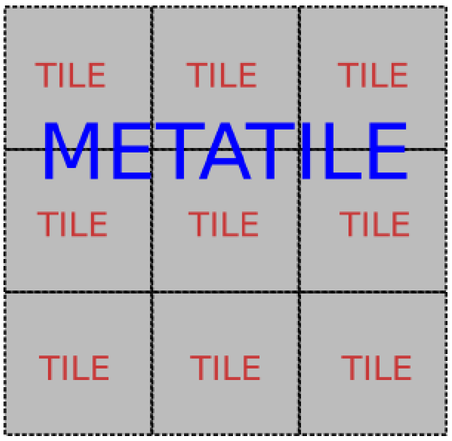
  If you are only looking to "lighten" your colors; rather than see through filled shapes to see other layers consider changing the saturation on a feature by feature basis.

  If you are looking to see line work or outlines of polygons under the current one – consider drawing just the outlines using one Rule, and the various Fill symbolizers in another.
- **Labels are Really Expensive:** Generating labels is a really expensive proposition; For example GeoServer has to collect all the possible labels and then shuffle them into position (depending on the priorities you set).

  There is no real alternative to text – so we will need to focus on minimizing the time it takes to shuffle labels into position. If possible try and use the Rule techniques to limit the number of features you generate labels for; and make use of label priority settings to assist in the selection process.
- **Halo is Very Expensive:** Rendering these labels is even more expensive if you add a Halo.

6.4 Are you Tile Caching?

The need to consider the end-users of your maps was discussed earlier, including what the viewing area of the maps is going to be. A small viewable area (say 200x300 pixels) impacts how the maps should be styled. If you plan to cache the maps you are styling in a tile cache like GeoWebCache, it is more than likely that the cache will build lots of small tiles, usually 256 by 256.



Even though these tiles may be used in a larger viewing area than the tiles, rending small tiles affects the map in some subtle ways. The most prominent of which is that labels won't be added to the map that cross the edge of a rendered map. For this reason most map tiling software support whats called metatiling. The tile cache will request a large map and chop it up into smaller tiles. This incurs a performance benefit as well as solving the styling problem.

In short, as long as you are aware of metatiling, you shouldn't have to do anything in your SLD's to cater for TileCache.

A concise discussion of the benefits and drawbacks of metatiling can be found at http://geowebcache.org/trac/wiki/metatiles.

Further Reading

SLD Cook Book

The SLD Cook Book, available from GeoServer is a collection of SLD "recepies" for creating various types of map styles. This page offers examples of various symbolizer techniques.

http://docs.geoserver.org/trunk/en/user/styling/sld-cookbook/index.html#sld-cook-book