Producing Custom Maps with Google Maps API

John D. Coryat, USNaviguide LLC

This discussion is aimed at those who understand the basic workings of Google Maps API, have reasonably good skills in Javascript and have used server side languages and techniques.

Presented by:

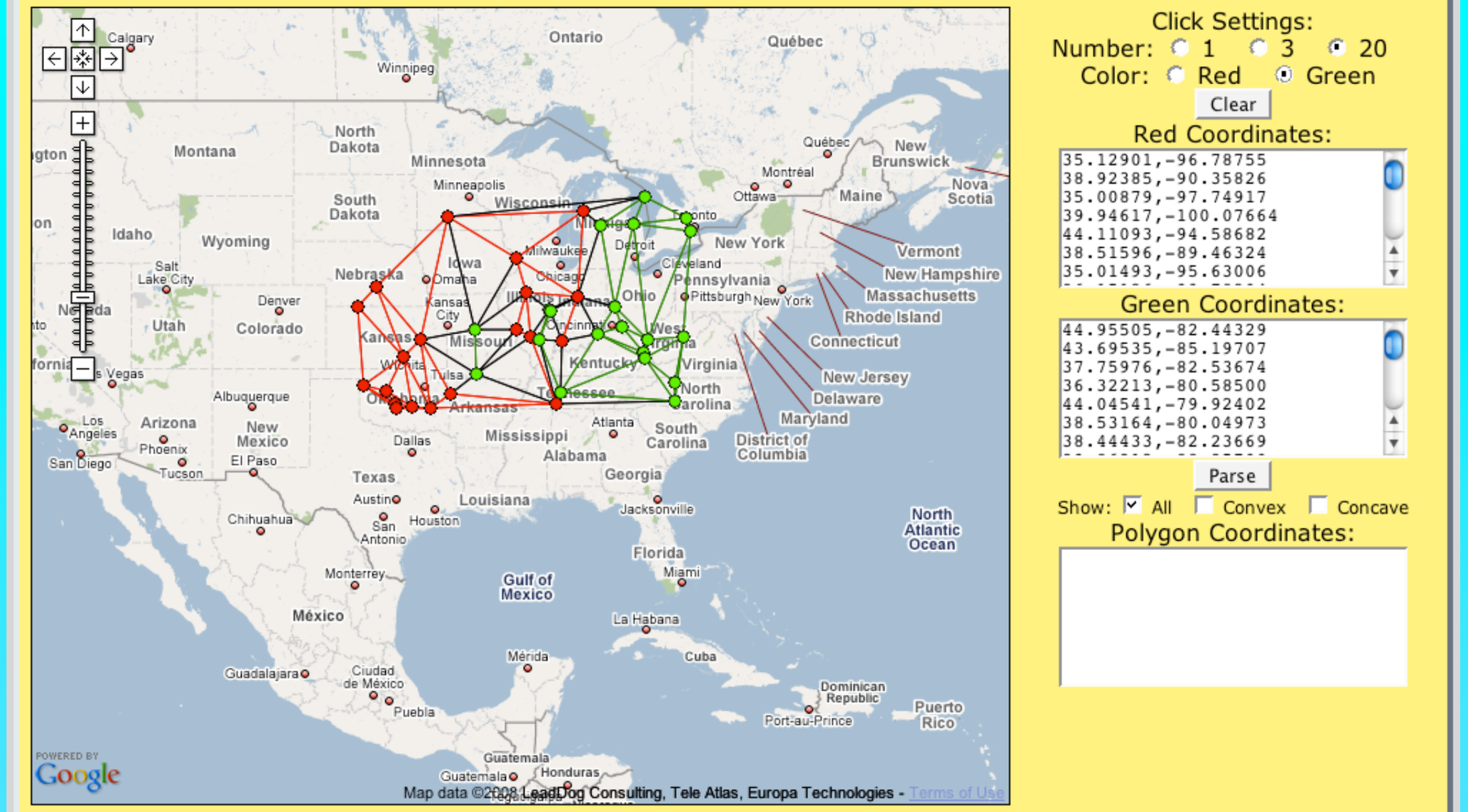
John D. Coryat
USNaviguide LLC

http://www.usnaviguide.com

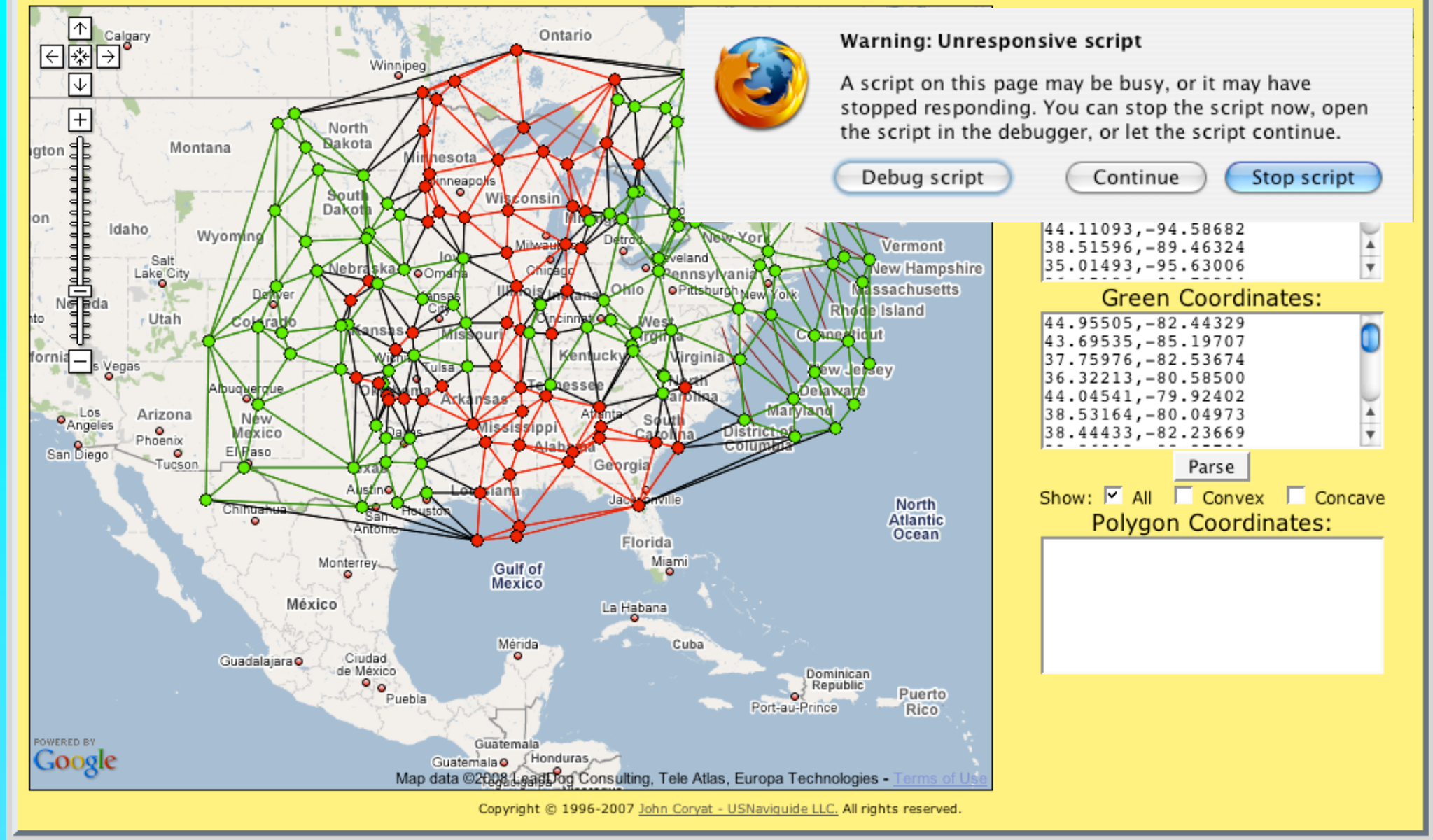
http://maps.huge.info

Markers and Polylines

http://code.google.com/apis/maps/documentation/examples/polyline-geodesic.html



Using markers and polylines for maps can be useful and productive… if you don’t have too many



The maximum number of markers and polylines change depending on the browser and computer running the page. Too many and this happens.

Markers and Polylines

Conclusion:

Markers and polylines work fine for smaller, less data intensive applications.

For anything more detailed, image overlays will be a more workable solution.

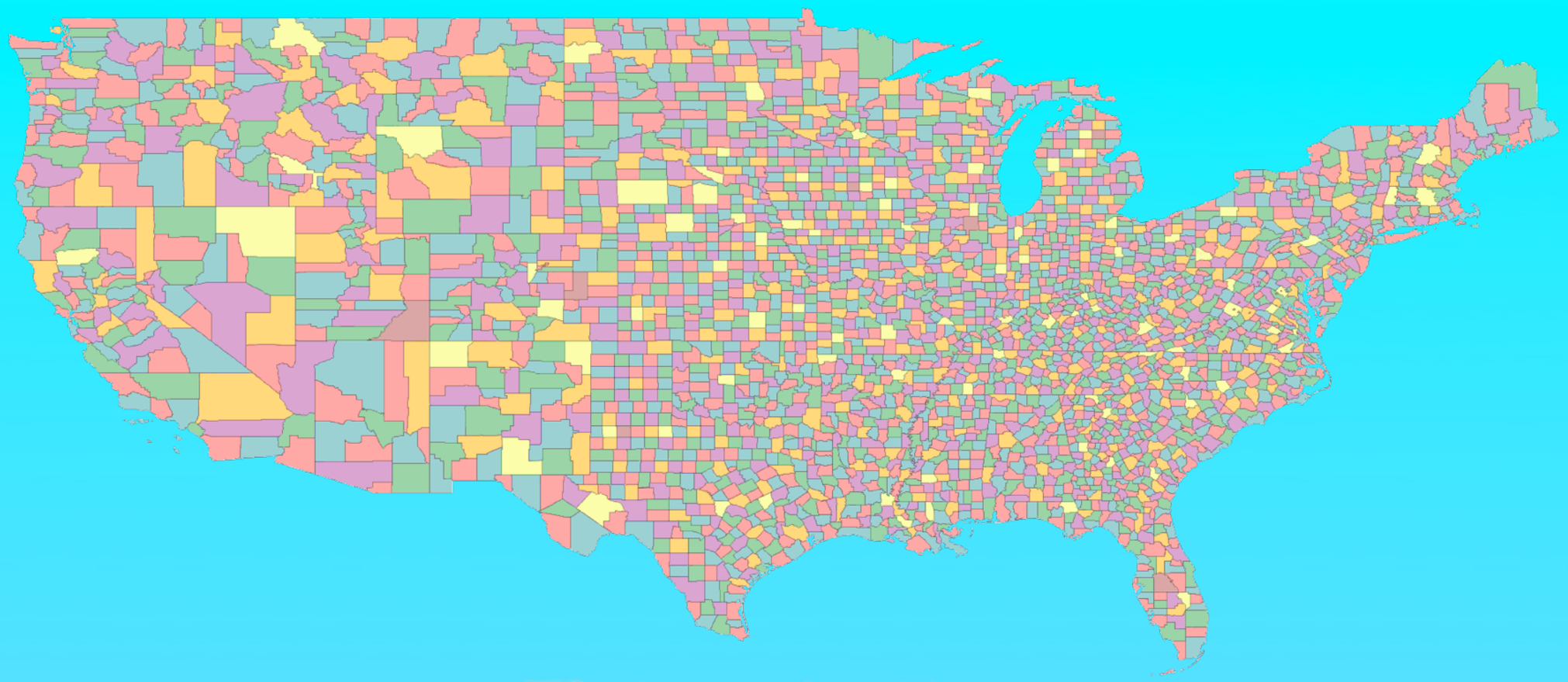
Map Image Overlays

Whole Image Overlays

and

Tile Overlays

GGroundOverlay



Flat projection
Image dimensions less important
No opacity options
Performance penalty

GGroundOverlay

Unprojected image

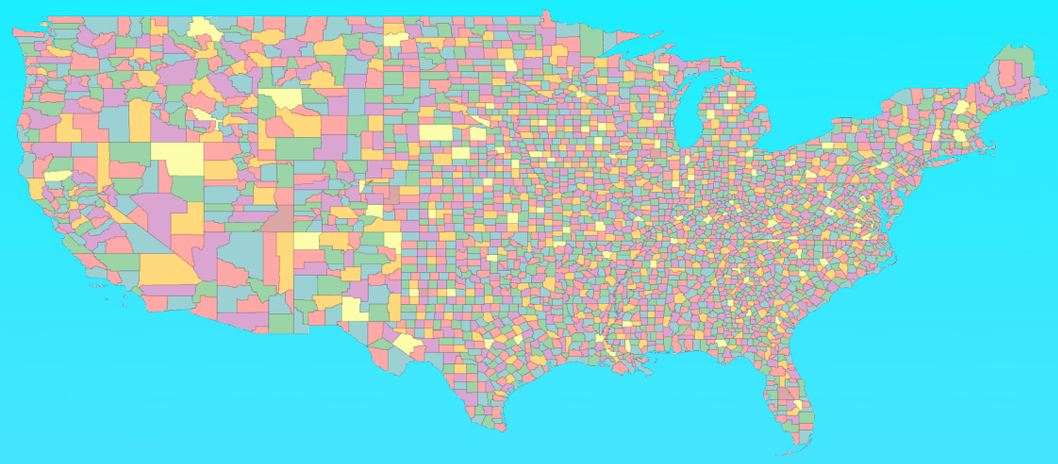


Image stretched and fitted into map

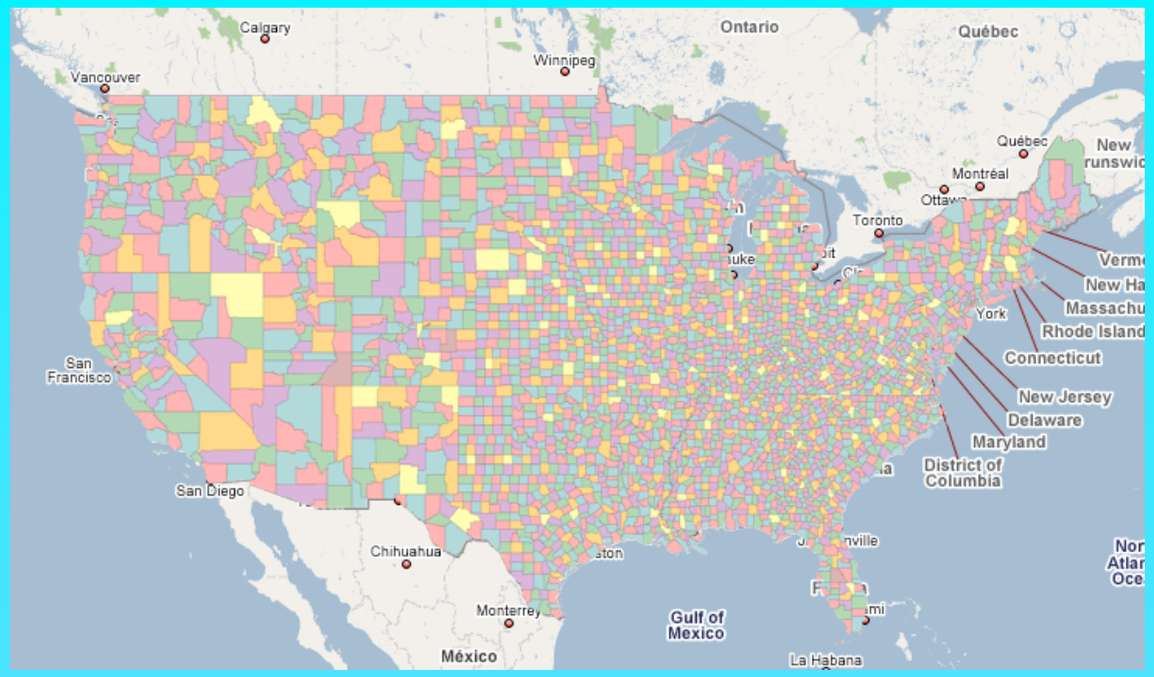


Image requires clever and elaborate Javascript to be stretched into the Mercator projection.

GGroundOverlay Demonstration:

GGroundOverlay Fitter
http://www.usnaviguide.com/ws-2008-02/ggroundoverlayfitter.htm

GGroundOverlay Fitted Example
http://www.usnaviguide.com/ws-2008-02/ggroundoverlay_example.htm

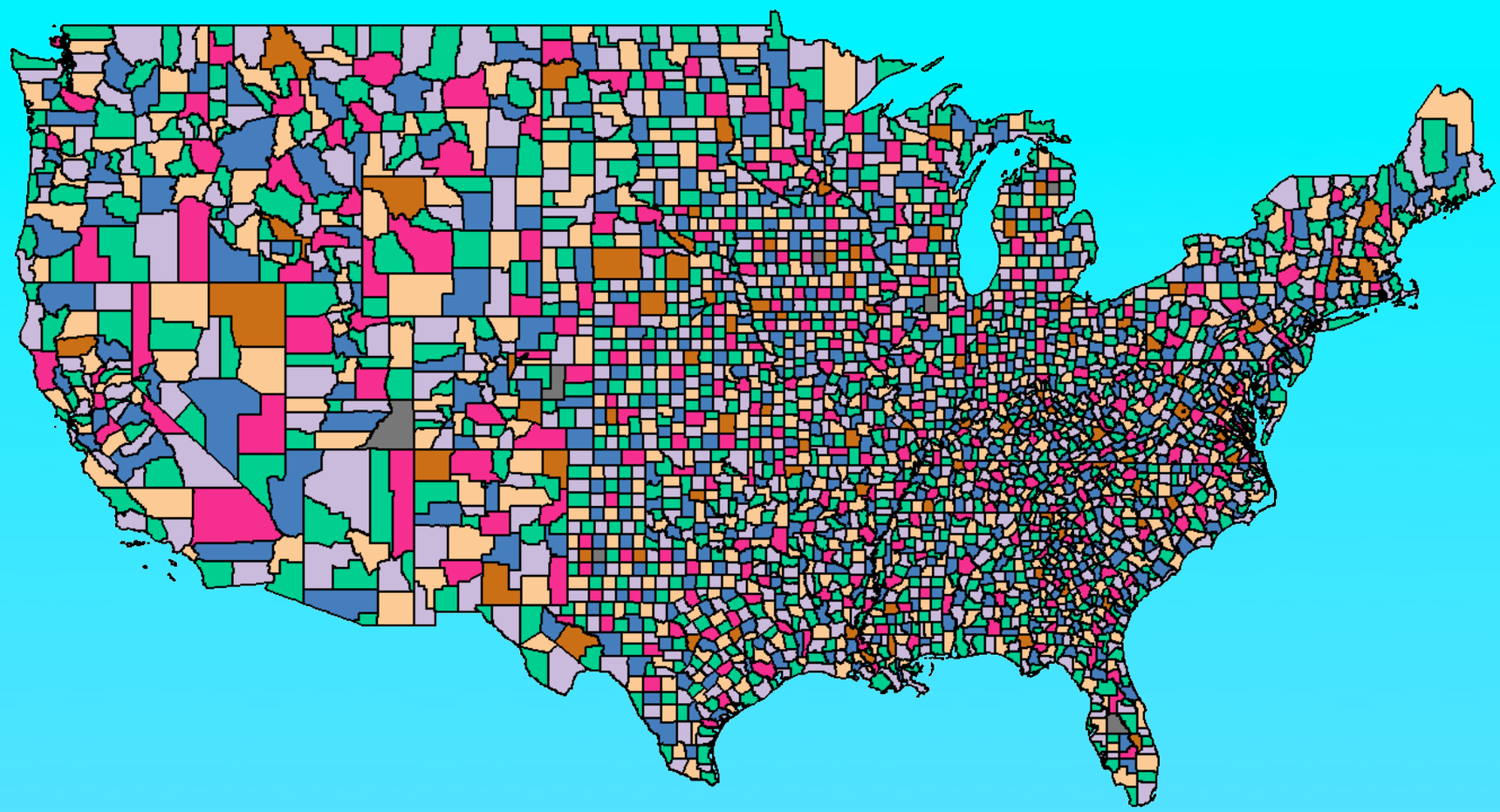
US County Flat Projected Image
http://www.usnaviguide.com/ws-2008-02/images/us_counties.png

GGroundOverlay

Conclusion:

GGroundOverlay works well for unprojected images, it's easy to implement but suffers from performance issues and limited options.

ProjectedOverlay



Mercator Projected Images
Image Dimensions critical
Opacity Options
Fast and Efficient

ProjectedOverlay

Mercator projected image

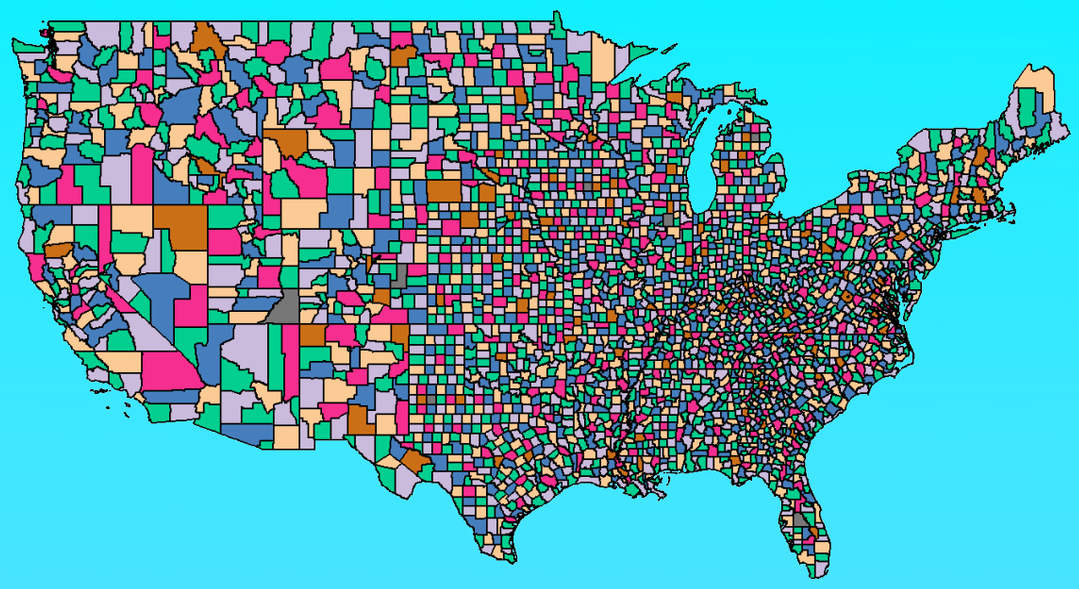
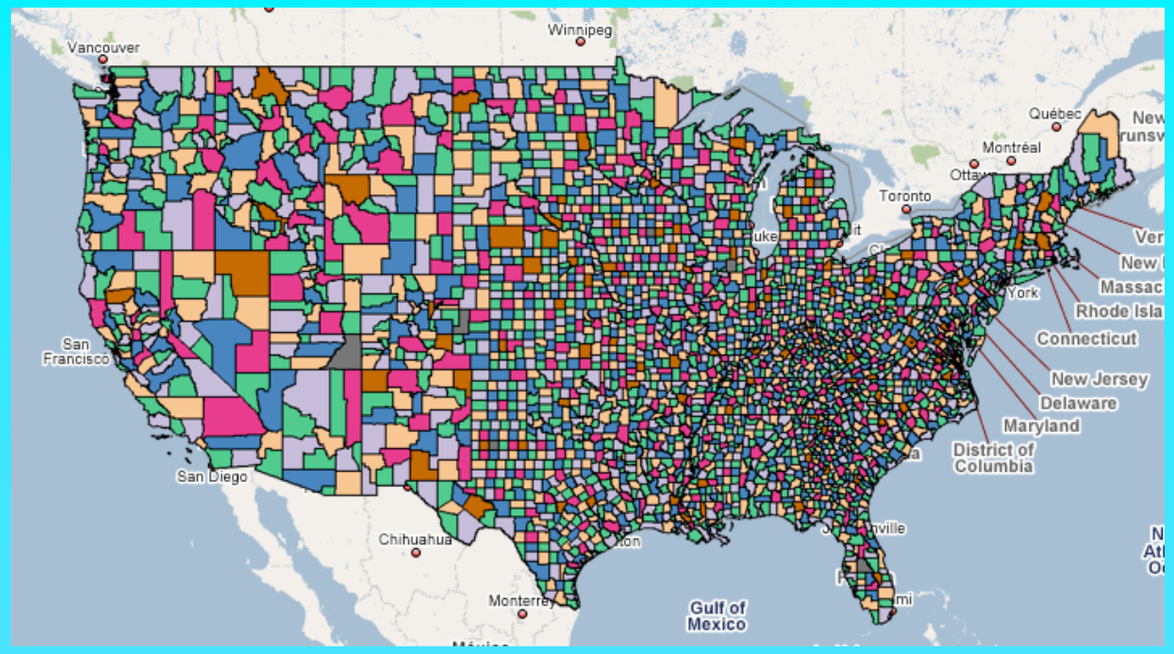


Image fitted into map



Very simple Javascript with fairly good performance.
Opacity options allow some flexibility.
Requires image to be fit closely to the map.
Images should fit the map viewport

ProjectedOverlay Demonstration:

ProjectedOverlay Example
http://www.usnaviguide.com/ws-2008-02/projectedoverlay_example.htm

US County Mercator Projected Image
http://www.usnaviguide.com/ws-2008-02/images/us_counties_projected.png

US County Overlay Created from Data on Demand
http://www.usnaviguide.com/ws-2008-02/countyoverlay.htm

US County Polygon Data (PostgreSQL dump format)
http://www.usnaviguide.com/ws-2008-02/data/counf.zip

ProjectedOverlay

Conclusion:

ProjectedOverlay images work well for areas that consist of a single map viewport and a single zoom setting. Excellent for program created images. Applications that require larger coverage need a more robust solution.

Tile Overlays

Extremely Efficient

Runs easily on all map capable browsers

Coverage from zoom 0 to 17 and over

Capable of displaying large areas

Somewhat complex to generate

More difficult to understand

Requires a robust server

Tile Overlays

Tile Structure

The tiling system consists of a series of images with a dimension of 256x256 pixels.

Each successive zoom level divides the previous zoom level's images into four new images, resulting in a map that displays one forth the area at twice the resolution of the previous level.

Zoom level 0 is the lowest level, there is no theoretical upper zoom level limit.

Tile Overlays

Tile Structure

Zoom System

Numbering Scheme

Pixel Calculations

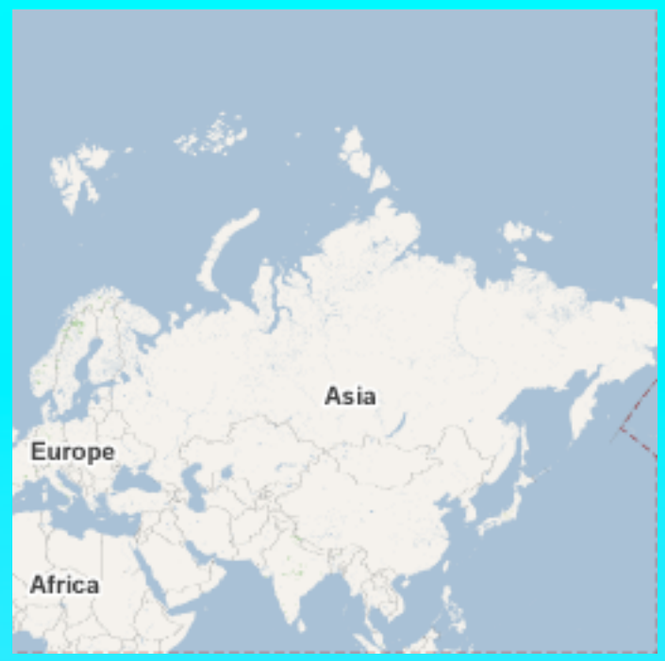
Tile Structure

Pixels:
Top (y): 0
Left (x): 0
Bottom: 255
Right: 255

Tile No.:
x:0 y:0

The world as one tile: zoom 0 (4**0)

Tile No.:
x:0 y:0
Pixels:
Top (y): 0
Left (x): 0
Bottom: 255
Right: 255



Tile No.:
x:1 y:0
Pixels:
Top (y): 0
Left (x): 256
Bottom: 255
Right: 511

Tile No.:
x:0 y:1
Pixels:
Top (y): 256
Left (x): 0
Bottom: 511
Right: 255

Tile No.:
x:1 y:1
Pixels:
Top (y): 256
Left (x): 256
Bottom: 511
Right: 511

The world as four tiles: zoom 1 (4**1)

Tile No.:
x:0 y:0
Pixels:
Top (y): 0
Left (x): 0
Bottom: 255
Right: 255

Tile No.:
x:3 y:0
Pixels:
Top (y): 0
Left (x): 768
Bottom: 255
Right: 1023

Tile No.:
x:0 y:3
Pixels:
Top (y): 768
Left (x): 0
Bottom: 1023
Right: 255

Tile No.:
x:3 y:3
Pixels:
Top (y):768
Left (x): 768
Bottom: 1023
Right: 1023

The world as 16 tiles: zoom 2 (4**2)

Tile Structure

Number of tiles per zoom level = 4**zoom

Number of pixels per zoom level = 4**(zoom + 8)

Zoom	Equation	Tiles	Equation	Pixels
0	4 ** 0	1	4**(0+8)	65536
1	4 ** 1	4	4**(1+8)	262144
2	4 ** 2	16	4**(2+8)	1 mil.
3	4 ** 3	64	4**(3+8)	4 mil.
4	4 ** 4	256	4**(4+8)	16 mil.
5	4 ** 5	1024	4**(5+8)	67 mil.
17	4 ** 17	17+ mil.	4**(17+8)	A lot!

Tile Structure Demonstration:

Demonstrate Tile Structure

http://www.usnaviguide.com/ws-2008-02/demotilestructure.htm

Tile Overlays

Tiles cut from images

Static data tiles

Dynamic data tiles

Tiles cut from images

Uses:

Obsolete and historical maps

Aerial and panoramic photos

Circuit boards, plans, engineering drawings

Other documents, books, magazines, photo albums

Advantages:

Easily integrated, great user interface

Disadvantages:

Pixilation, Mercator stretching, image geolocation errors

Tiles cut from images

Resources:

MapCruncher Beta for Microsoft Virtual Earth

Great alignment options, limited to non-commercial use, non-Google tile numbering

Automatic Tile Cutter for Photoshop - Mapki.com

Requires Photoshop CS or better, alignment web tool available

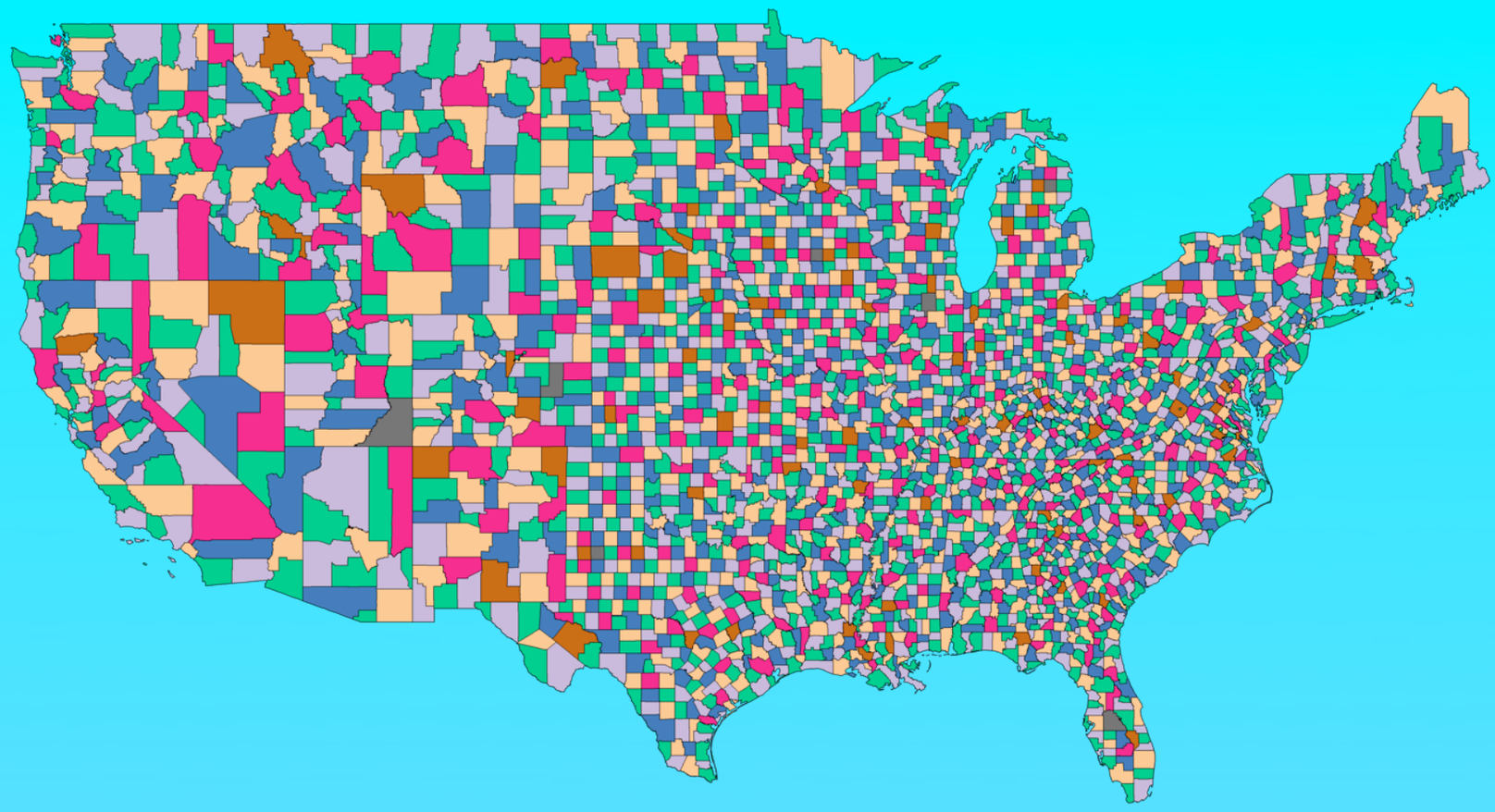
Unix Command Line Tile Cutter - crazedmonkey.com

Runs under Linux/Unix/Mac OSX, uses ImageMagick, GPL, no image alignment tool

Perl tile cutter - USNaviguide.com

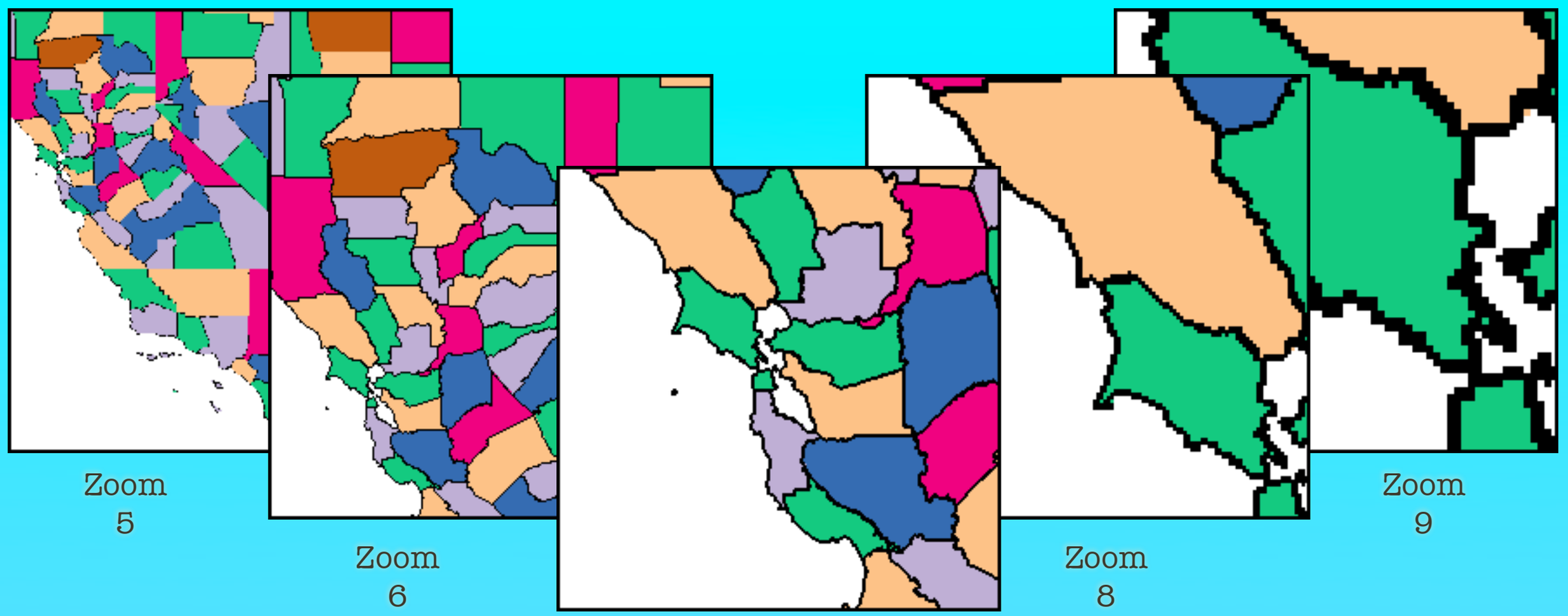
Runs under Linux/Unix/Mac OSX, uses GD, GPL, no image alignment tool

Tiles cut from images



Source image: 5462 x 2920 zoom 7

Tiles cut from images



Zoom 5

Zoom 6

Zoom 7

Zoom 8

Zoom 9

Source image: 5462 x 2920 zoom 7 lower and higher zooms pixilated

http://www.usnaviguide.com/ws-2008-02/countyimagetiles.htm

Tiles cut from images

Demonstration:

US County Tile Overlay from an Image
http://www.usnaviguide.com/ws-2008-02/countyimagetiles.htm

Program to Cut Tiles from a Mercator Projected Image
Command Line Execution: tilecutter.pl <zoom> <South>,<West> <North>,<East> <Source>

Tiles cut from images

Conclusion:

Tiles cut from images are useful for many purposes, but suffer when extending to more than a few zoom levels due to pixilation.

Difficult to align to the map.

Complex to use over large areas.

Static data tiles

Uses:

Displaying points of interest, heat maps, borders, other thematic data

Advantages:

Allows creation of nearly perfect tiles

Fast browser response regardless content

Great for polygons, markers and lines

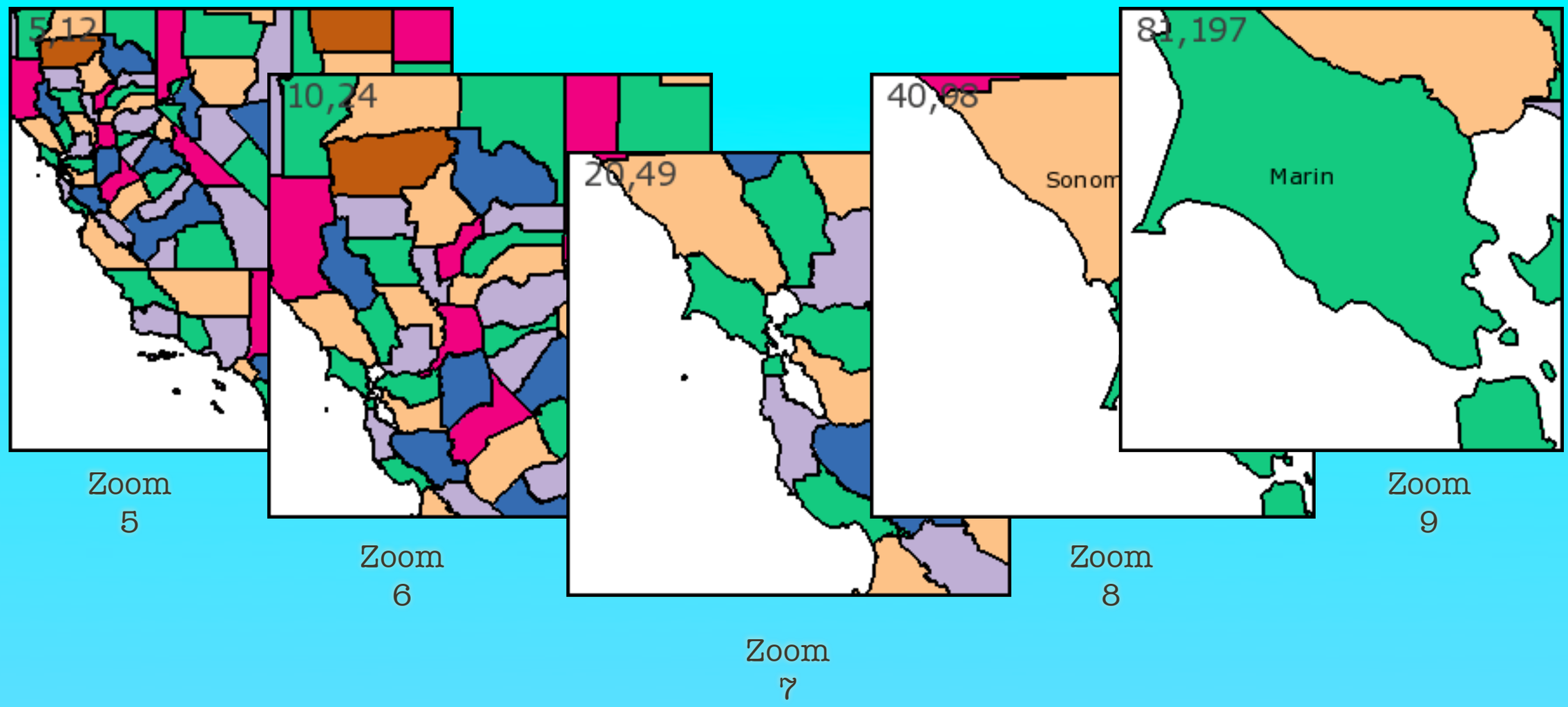
Disadvantages:

Requires significant server side programming experience

Tiles generated on a schedule, not great for time sensitive apps

Difficult to use with client side options

Static data tiles



Tiles from data offer better quality images

http://www.usnaviguide.com/ws-2008-02/countytiles.htm

Static data tiles

Demonstration:

US County Tile Overlay
http://www.usnaviguide.com/ws-2008-02/countytiles.htm

Program to Generate Tiles
Command Line Execution: countytiles.pl <zoom> <South>,<West> <North>,<East>

US County Polygon Data (PostgreSQL dump format)
http://www.usnaviguide.com/ws-2008-02/data/counf.zip

Static data tiles

Conclusion:

Static tiles created from data are extremely fast and efficient, can be made to work with any zoom level and any area. They don’t suffer from pixilation and are easily aligned to the map. Since they require building in advance, time sensitive and user driven requirements are difficult to address.

Dynamic data tiles

Uses:

Simulating markers, User driven data, time sensitive data

Advantages:

Tiles custom tailored to the user

Solve the “too many markers” problem

Disadvantages:

Requires significant server side programming experience

Requires significant server side processing power

Dynamic data tiles

Demonstration:

Data Tile Layer Generated “on the fly”
http://www.usnaviguide.com/ws-2008-02/demotilecookies.htm

US County Polygon Data (PostgreSQL dump format)
http://www.usnaviguide.com/ws-2008-02/data/counf.zip

Program to generate “bigcity” from Geoname.org US.zip
Command line execution: bigcitytilename.pl

Geonames.org US.zip data as “bigcity” file
http://www.usnaviguide.com/ws-2008-02/data/bigcity.zip

Dynamic data tiles

Conclusion:

Dynamic tiles created “on the fly” offer great functionality at a stiff price - increased time to process and heavy load on the server. The benefits of dynamic tiles over static ones have to be weighed for each application.

Producing Custom Maps with Google Maps API

Questions?

Getting Help

Google Maps API Discussion Group

http://groups.google.com/groups/Google-Maps-API

Google Mapki

http://mapki.com

Google Search

http://www.google.com

Software and Data used in this discussion

Perl 5.8 with DBI, GD and CGI modules
USNaviguide_Google_Relpix Perl module
USNaviguide_Google_Tiles Perl module
PostgreSQL 8.2 with PostGIS Extension
Apache 2.0 webserver
ProjectedImage.js Javascript

US Census Cartographic Boundary County Shapefiles
Geonames.org's US.zip data file

Download examples and data used in this discussion:
http://www.usnaviguide.com/ws-2008-02/download.zip

Special thanks to Marcelo Montagna for help
with some examples used in this discussion.