Paper 025-2010

Google Maps and SAS/GRAPH®

Darrell Massengill, SAS Institute Inc., Cary, NC

ABSTRACT

Google Maps have become one of the most widely recognized means of handling maps and location data. This paper introduces a tool and shows you how to create your own Google Maps from within SAS® and SAS/GRAPH. In addition, we will briefly discuss Google Maps and the legal and technical limitations and implications of using these maps.

INTRODUCTION

Numerous SAS users have asked for a way to put their SAS location data on a Google Map so it can be displayed on the Web. In answer to this request, I created a tool that will generate the files for displaying choropleth maps on a Google Map. In addition, SAS/GRAPH Annotate data can be used to create polygons, lines, and markers on the map. This new tool is called the SAS Google Map Generator (GMG). Figure 1 is an example of a map that can be created.

In order to understand the GMG tool, you must first understand Google Maps and the legal and technical limitations of using these maps.

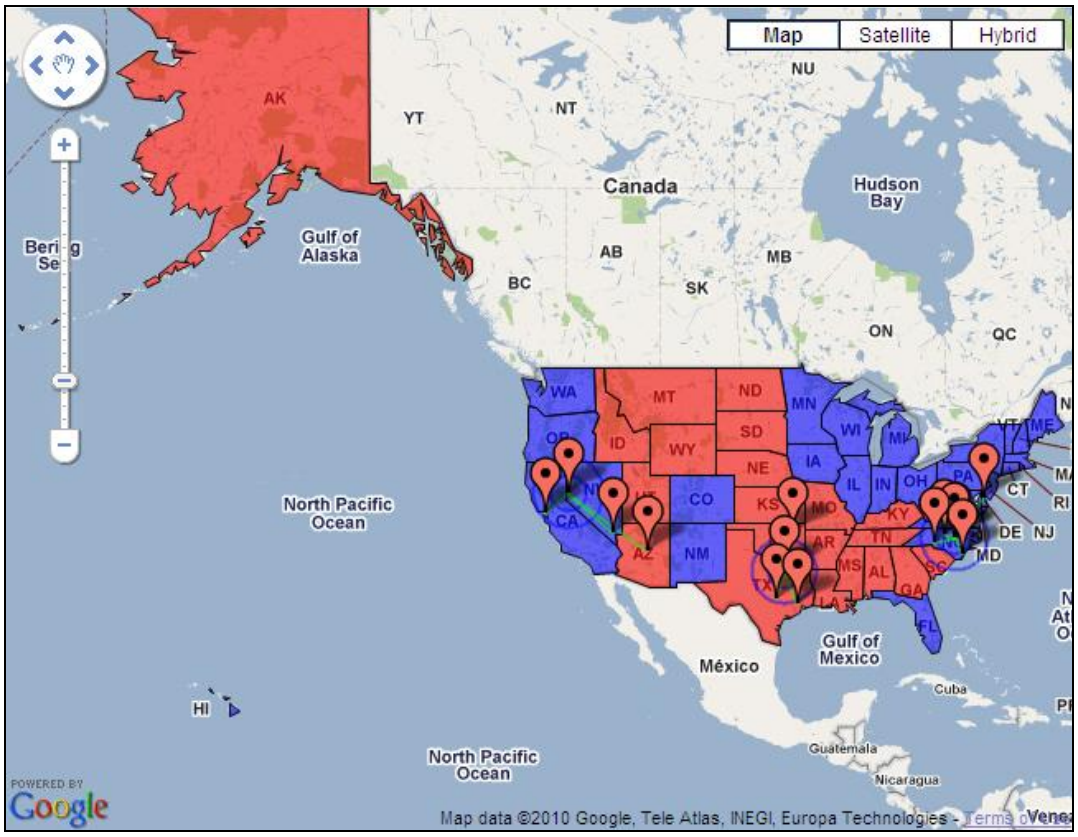

Figure 1.

GOOGLE MAPS

Since their introduction in 2005, Google Maps have been a popular way of delivering maps on the Web. The ubiquitous use of Google Maps has helped to make many people comfortable with mapping technology by making maps easy to access and use. Although most people are familiar with Google Maps, there are some details that people might not know. First, there are limitations regarding the use of Google Maps that are important to understand. Second, when creating your own Google Maps, you must register your Web site and get your own Google key. This key is required to use the SAS Google Map Generator. Finally, you need to understand the Google Map user interface and the structure of the Google map files.

TERMS OF USE

When creating your own Google Maps, you must follow the Google Terms of Use/Terms of Service. These are located at http://code.google.com/apis/maps/terms.html. Here are a couple of important things to note:

1. Google Maps must be freely available. You cannot charge to access the maps.
2. Google Maps must be publically available. You cannot operate them behind a firewall or on an internal network.

Google Maps API Premier is a commercial product that can be licensed for situations where these restrictions on the free Google Maps are not acceptable.

GOOGLE KEY

Your maps must have a key from Google. The key is registered for your Web site's URL address. Your maps can be displayed only from the Web site that you register. You get your key from http://code.google.com/apis/maps/signup.html. When getting this key, you agree to comply with the Terms of Use mentioned previously. The SAS Google Map Generator tool requires that you have this key.

UNDERSTANDING GOOGLE MAPS

In order to create a Google Map, you need understand a little about how the map works. Three areas will be covered: the map user interface, the map HTML file, and map overlay data.

Most people are familiar with the Google Map user interface, but might not know what the various parts are called. The user interface consists of a number of controls that appear on the map itself. It is important to understand these controls because the GMG tool can set these. The controls allow you to zoom, navigate, and otherwise use the map. They can be changed and turned on or off. Figure 2 labels the controls on the map.

- The Map Type control allows the user to select between different types of maps being displayed.
- The Default Type is the Map Type that is selected for viewing.
- The Navigate control allows the user to move the map view around. By default, the user can also pan or drag the map view.
- The Zoom control allows the user to zoom the map.
- The Scale control shows the scale of the current view.

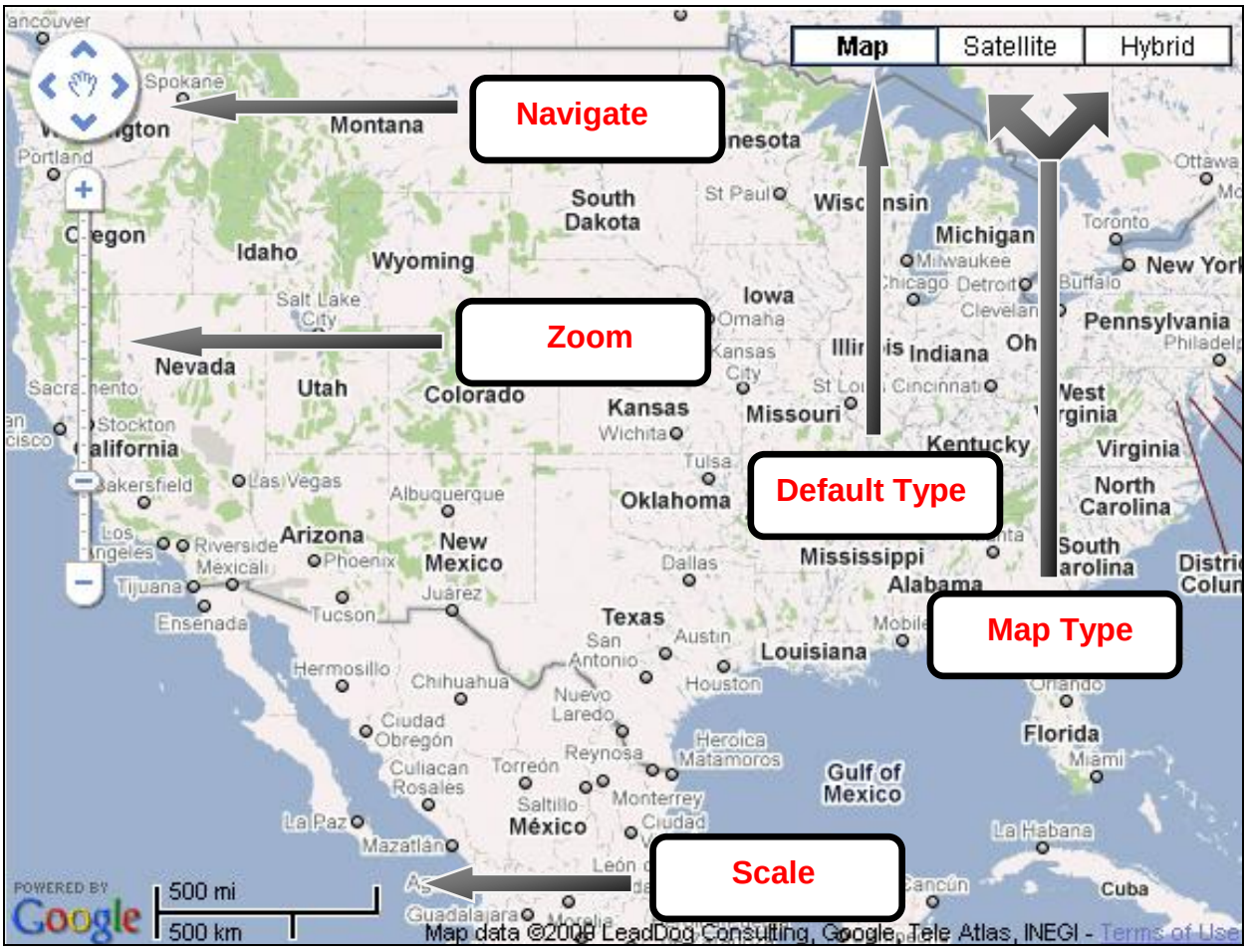

Figure 2.

The Google Maps HTML file determines how the map looks and behaves. This HTML file looks like a standard XHTML Web page. However, it contains special code to initialize the Google map and run the Google Maps API. The controls are set and changed with the Google Maps API. Figure 3 shows the basic sections of the file. The SAS Google Map Generator adds additional JavaScript functions and additional Google Maps API statements.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
     <html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:v="urn:schemas-microsoft-com:vml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Map</title>
    <link href="../base.css" rel="stylesheet" type="text/css"/>

    <script src="http://maps.google.com/maps?file=api&amp;&v=2.x
            &key=ABC_YOUR_KEY_HERE_DEF&sensor=false"
            type="text/javascript"></script>

    <script type="text/javascript">
    var map;

    function loadMap() {
      if (GBrowserIsCompatible()) {
        map = new GMap2(document.getElementById("map"));
        map.addControl(new GLargeMapControl3D());
        map.addControl(new GMapTypeControl());
        map.setMapType(G_NORMAL_MAP);
        map.setCenter(newGLatLng(37.4419, -122.1419),13);
       }
     }
    </script>

  </head>
  <body onload="loadMap()" onunload="GUnload()">
        <div id="map" style="width: 576px; height: 436px"></div>
  </body>
</html>
```

#1

#2

#3

#4

Figure 3.

There are four sections in the HTML file. Section 1 is the standard XHTML code. Section 2 is special initialization code for Google Maps code. The second line beginning with `&key` should be on the previous line, but was split for printing. The key value should represent your Google key. (The key in Figure 3 is not valid.) Section 3 shows JavaScript code containing the Google Map API. Note that the code adds controls to the map user interface. Additional JavaScript code or files containing JavaScript code could be inserted before or after this section. Section 4 contains the body code that defines the HTML layout. This example code sets the height and width of the map. The `onload` event causes the `loadMap` function to be called when the Web page is loaded and the map is drawn. The body section can also contain code for drawing push buttons and other items on the Web page.

Google Maps support several types of overlay data. The two used by the GMG tool are KML and GPOLY data. KML is a markup language that can be used with Google Maps and Google Earth. Google Maps only supports a subset of the full KML specification. KLM cannot support the same types of events and interactions that are available with GPOLY data. GPOLY data is generated with the Google Maps API and GPolygon, GPolyline, and Gmarker, so it is possible to have more control of the overlay items.

SAS GOOGLE MAP GENERATOR

The SAS Google Map Generator will convert SAS/GRAPH map data sets to Google Maps polygons, convert SAS/GRAPH Annotate data sets to Google polygons, lines, and markers, and generate the HTML file and all necessary JavaScript code to display these on the Google Map. This tool only generates the files; the user must move these to these files to their external Web site for displaying.

The SAS Google Map Generator is a SAS/AF SCL application that is run by SAS. It can be run as a batch program or as a SAS Stored Process, or it can be run interactively. SAS/AF is not required to run this application. This tool can take a slightly modified SAS map data set, an Annotate data set, or both as input. It creates either KML output

or code to directly create Google Map polygons, lines, and markers. These are referred to as GPOLY by the SAS Google Map Generator.

The generator can optionally generate the HTML and JavaScript files necessary to display the KML and GPOLY files. Options used with the GMG tool can change and modify the controls that appear on the map. Some required, pre-generated JavaScript files are packaged with the generator. Users can specify their custom JavaScript code too.

HOW IT WORKS

The SAS Google Map Generator converts a modified SAS map data set or an Annotate data set into either a KML file or a GPOLY JavaScript file. Data sets are passed in and the generator creates the files necessary to draw the map. Since the tool is written in SCL, parameters and options must be set with macro variables. There are only four required macro variables, but there are many more that allow you to fine tune your output. Some sample code will help in understanding how it works.

The options are set with four macro variables, and then a PROC DISPLAY is called to run the SCL code. For example, to create a map with polygons and annotation, you would specify the following:

```
%let _GOOGLEKEY="__GetYourOwnGoogleKey__";
%let _MAPDATA = sasuser.mymap;
%let _ANNODATA = sasuser.annodata;
%let _OUTPUTFILE = c:/googlemaps/sample.html;

proc display c=google.googlemap.creategooglemap.scl; run;
```

This example would create all files at C:/googlemaps. The HTML file would be named sample.html. The other files created would all be based on the filename of 'sample', which is specified with the _OUTPUTFILE macro. The map data file has an 'm' appended to the name, the Annotate file has an 'a' appended to the name, and a map bounds file has a 'b' appended to the name. In this example, the Annotate data would be in samplea.js, the map data would be in samplem.js, and a bounds file for centering the map would be created as sampleb.js. The HTML file created includes these files and code to use the data in the files.

Syntax

The following options are all macro variable names that are set before the call to PROC DISPLAY. Some options are required, some are strongly recommended, and others are optional. The required options are those for the Google key, the input data sets, and the output file. Note that because these are macros variables, the value will remain set and you will need to clear them between different types of maps.

_**GOOGLEKEY** is the key value from Google when you registered your Web site.

_**ANNODATA** is the name of the SAS data set containing the Annotate data. The data set details are discussed later.

_**MAPDATA** is the name of the SAS data set containing the modified map data. The data set details are discussed later.

_**OUTPUTFILE** is the file path and filename for the files being created. The HTML file path and name are specified even if you do not create an HTML file. By default, all other files created will also be at this location and will take on the same filename prefix. As noted above, 'a', 'm', or 'b' is appended to the name along with the proper file extension. Ideally, this path will point to a location that exists on your external Web server.

Note that at least one of _ANNODATA or _MAPDATA must be specified, but both are not required.

The 'strongly recommended' options allow you to customize the information in your processing.

_**OUTPUTDATAPATH** is the path that the data files are written to instead of the path specified by _OUTPUTFILE. This is only the path and not the filename. The path must contain the ending pathname separator character for the operating system (for example, the ending slash in C:\mydir\). This path is not verified.

_**WEBDATAPATH** is the Web path or partial path that is used at run time to locate the data files. This is the location written to the HTML file. The path used by _OUTPUTFILE or _OUTPUTDATAPATH is a local file path and is used to generate the output files. These might be different from the Web location that is needed when the map is run. The path specified must contain the ending pathname separator character for the operating system (for example, the ending slash in mydir\ or http://www.sas.com/mydir/). This path is not verified.

_**WEBJSPATH** is the Web path or partial path that is used at run time to locate the pre-created JavaScript files that are shipped with the GMG tool. This is the location written to the HTML file. If _WEBJSPATH is not specified, these JavaScript files are assumed to be in the same location as the HTML file. The path specified must contain the ending pathname separator character for the operating system (for example, the ending slash in mydir\ or http://www.sas.com/mydir/). This path is not verified.

_**USERJS** specifies the path and file that contains user-written JavaScript and is used at run time. The path specified should be the Web-based location.

_**BODYCODE** specifies a file that can contain the body code for the HTML file. This replaces the #4 section shown in Figure 3. This location is a file-creation-time location. It is used in generating the files.

_**OUTPUTTYPE** specifies the type of output created by the generator. The values include the following:

KML specifies that KML data output and the HTML file are generated.

KML_ONLY specifies that only the KML data output is generated. This is useful when you need to update the SAS data, but already have an existing map.

KML_CUSTOM is like KML, but uses the CUSTOMfunc.js file as described later.

GPOLY specifies that Google Map GPolygon, GPolyline, and GMarker data are generated. Also the HTML file is created. This is the default type if _OUTPUTTYPE is not specified.

GPOLY_EVENT is like GPOLY, but enables event handling on polygons.

GPOLY_CUSTOM is like GPOLY, but uses the CUSTOMfunc.js file as described later.

GPOLY_ONLY specifies that the GPOLY data output is generated and not the HTML file. This is useful when you need to update the SAS data, but already have an existing map.

Note that there are two types of file paths used with these options. One path is used by the SAS Google Map Generator to generate the map. These are local machine paths where you can read or write local files from SAS. The second type of path is a Web-based address. These can be a full http:// address, or they can be a partial address relative to some http:// address. The Web-based addresses cannot be validated, and the user must ensure they are specified correctly.

The following list of options allows you to customize the appearance of the map:

_**MAPTYPE** turns the Map Type control on or off. Accepts ON or OFF. ON is the default.

_**MAPDEFAULT** sets the Map Type that is selected when the map first appears. Accepts STREET, HYBRID, SATELLITE, or TERRAIN. STREET is the default.

_**ZOOM** turns the Zoom and Navigate controls on or off. Accepts ON or OFF. ON is the default.

_**NAVIGATE** turns the Navigate control on or off. Accepts ON or OFF. ON is the default.

_**SCALE** turns the Scale control on or off. Accepts ON or OFF. ON is the default.

_**DRAG** turns the ability to drag or pan the map on or off. Accepts ON or OFF. ON is the default.

_**MAPSIZEX** specifies the height of the map in number of pixels. Accepts an integer. Default is 500. This value is ignored if the user specifies a _BODYCODE file.

_**MAPSIZEY** specifies the width of the map in number of pixels. Accepts an integer. Default is 650. This value is ignored if the user specifies a _BODYCODE file.

_**CENTERLAT** specifies the map center latitude in degrees. Accepts a latitude . The default is calculated based on data drawn on the map.

_**CENTERLON** specifies the map center longitude in degrees. Accepts a longitude. The default is calculated based on data drawn on the map.

_**ZOOMLEVEL** specifies the initial map zoom level. Accepts an integer. The default is calculated based on data drawn on the map.

Special files

There are a number of files that ship with the SAS Google Map Generator. These are used by the HTML file and JavaScript code. They should not be modified. This list might change or grow with later releases. References to these files are inserted into the HTML file as needed. The _WEBJSPATH option sets the run-time location of these files. If this option is not specified, the files must be in the same location as the HTML file.

SASpolysubclass.js – This JavaScript file enables events for GPOLY data.

SASfunc.js – This JavaScript file creates the map polygons from the GPOLY map data (without event handling).

SASfunca.js – This JavaScript file creates the annotation from the GPOLY Annotate data (without event handling).

SASfuncE.js – This JavaScript file creates the map polygons from GPOLY map data (with event handling).

SASfuncEa.js – This JavaScript file creates the annotation from the GPOLY Annotate data (with event handling).

SASKML.js – This JavaScript file handles maps and annotation from KML data.

User Files

There are several files that are intended for the user to use and modify. Some of these files might contain example code that can be changed and modified.

CUSTOMfunc.js – This file can be created if the user wants to use his or her own custom JavaScript code instead of any of the preceding files. This file will be inserted into the HTML file instead of any of the special files previously listed. This will be used when GPOLY_CUSTOM or KML_CUSTOM is specified. Note that the location of this file is also determined by the _WEBJSPATH setting.

USER.js – This file contains example JavaScript code for the file specified with the _USERJS option. The JavaScript in this file can be used to tailor your map output and functionality.

BODYcode.js – This file contains example code for the BODY section of the HTML file. The file is specified with the _BODYCODE option. This code sets the layout of your map, adds buttons and other features, and can drive custom HTML in the _USERJS file.

Data Sets

There are two possible input SAS data sets. One contains the map polygon data and the other contains the Annotate data. Extra processing of both data sets is required before they can be processed by the SAS Google Map Generator. Both of these data sets can have additional variables with additional information that the GMG will use. Some of the extra information is required.

A SAS map data set normally contains variables named X and Y, and an id variable and optionally a variable named SEGMENT. The generator requires that you set a variable name ID to contain the map id information. In addition, you must pass a variable named COLOR to indicate the color of the polygon. The generator does not take your response data and categorize your map results the way PROC GMAP does. You must categorize your data before sending it to the generator and decide what color you want the polygons to be. In addition, you can optionally pass a variable for TRANPARENCY, DESCRIPTION, and HTML. Following are the specifics of each variable:

ID is a variable containing the ID variable information. For example, if the data set ID variable is named ‘state’, then you must either rename ‘state’ to ‘ID’ or you must set ID to state (i.e., ID=state;) (Required variable)

COLOR is a character string containing a SAS color. A few of the basic named colors are supported (RED, GREEN, BLUE, and so on.) but it is safest to use the SAS/GRAPH CX color notation (CXFF0000 is red). Unknown named colors are mapped to gray. (Required variable)

TRANSPARENCY is a numeric value representing the transparency of the polygon fill color. The number must be between 0 and 1 (inclusive). If this variable isn’t specified or is specified incorrectly, then 0.5 is used.

DESCRIPTION is a character string with descriptive text about the polygon. This information is available for use by JavaScript functions (tooltips, for example).

HTML is a character string with an HTML link. This information is available for use by the JavaScript functions. (For example, you can drill to this link when you click on the map).

In addition to setting these variable values, it is important to reduce the number of points in the map data set polygons. Google and KML have limits on the sizes of these files. In addition, the more points you have, the slower

the map will draw. PROC GREDUCE can be used to do this reduction. It is recommended that you download the %REDUCE macro code (PIXEL_REDUCE.sas) from SAS Maps Online Web site to do the reduction. The macro allows you to specify a display resolution in pixels, and the map will be reduced to look best at that size. Finally, Google Maps needs the X and Y coordinates in degrees and to be unprojected. If the map has the data in radians, then it must be converted to degrees.

The following code fragment illustrates processing the map data set before passing it on to the SAS Google Map Generator. The code does the following:

1. Convert the X and Y values from radians to degrees.
2. Reduce the number of points in each polygon. Note that the %REDUCE macro must be downloaded from the SAS Maps Online Web page (http://support.sas.com/mapsonline). The macro reduces the map based on the size you intend to display the map. You can specify a smaller size to reduce it more.
3. Remove any two point polygons created by the reduction.
4. Create and set the following variables: ID, COLOR, TRANSPARENCY, DESCRIPTION, and HTML. COLOR is set according to the values in the response data set.

```
/* Convert the X and Y coordinates from Radians to Degrees */
data mymap;
  set maps.states;
  x=-x*45/atan(1);
  y= y*45/atan(1);
run;

/* Reduce the map data with special macro */
/* This must be downloaded.  See notes.   */
 %include lclpgms(reduce_pixel);
/* arguments: Input ds, output ds, ID var, xpixels, ypixels */
%reduce(mymap, mymap, state, 500, 650 );

/* Remove any tiny polygons.  Also, subset map and add ID variable */
proc summary data=mymap sum;  by state segment;
    output out=work.sumstates;
run;
data mymap; merge mymap work.sumstates; by state segment; run;
data mymap; set mymap; length id $ 10;
    id = fipstate(state); /* ID required by generator */
    if (_freq_ > 2 and id ne 'PR') then output;
run;
proc sort; by id; run;

/* Set the required and optional variables needed by generator */
/* Add and set Description, Transparency, and HTML              */
/* Set Color based on response data.  The map is colored red or*/
/* blue based on which Presidential candidate won the state -  */
/* Obama or McCain.                                            */
filename response 'election.dat'; /*set to your location*/
data election; infile response;
   length id $ 10;  input id Obama Mccain;
run;
proc sort; by id; run;
data mymap; merge mymap election ; by id;
   length description $ 1024;
   transparency = 0.6;  /* set this value for all polygons */
   /* set each state polygon to the state government HTML link */
   html="http://"||trim(left(id))||".gov";
   /* set color based on who won the state */
   if Mccain > Obama then do;
      color="CXFF0000"; /*red state*/
      description="McCain won";
   end;
   else do;
      color = "CX0000FF";
      description = "Obama won";
```

```
    end;
run;
```

The Annotate data set is the second type of data that can be passed in. Not all Annotate functions are supported. Some of the variables and values are not supported. Some cannot be supported because Google Maps do not have equivalent functionality, such as labeling. Some are not implemented, but might be added in the future. And finally, some functions are supported only in a limited fashion. Also, only XSYS and YSYS of 2 and HSYS of 3 are supported for the Annotate coordinate system. The Annotate data set must contain all standard Annotate variables. There are four additional, optional variables:

ID is a name for the Annotate item. This is useful for items like markers. If a name is not specified, a name will be created. It is recommended that you create your own meaningful ID.

TRANSPARENCY is a numeric value representing the transparency of the polygon fill color. The number must be between 0 and 1 (inclusive). If this variable isn't specified or is specified incorrectly, then 0.5 is used.

DESCRIPTION is a character string with descriptive text about the polygon. This information is available for use by JavaScript functions (tooltips or KML balloons, for example).

HTML is a character string with an HTML link. This information is available for use by the JavaScript functions. (For example, you can drill to this link when you click on the map). In KML, this would be added to the information in the popup balloon.

The following code fragment is used in the examples. After the Annotate data set is created, the ID variable is set to the customer name. A description and HTML variable are also set.

```
data points;
  set points;
  id=customer;
  description="Click on the link";
  html="http://www."||left(trim(statecode))||".gov";
run;
```

The complete program is found with the example files in CreateData.sas. The program creates map polygons of the U.S. states that are colored by the 2008 presidential election results. The Annotate data shows three store/shipping locations and some customers that ordered from the stores. Lines connect the customers to the store location they purchased from. The circles around the store location represent 100-mile and 200-mile distances from the store to determine which customers fall outside those ranges.

Generated Files

The SAS Google Map Generator will create several files, depending on the options you use. Note that letters at the end of the name indicate the contents of the file: 'm' is for map data, 'a' is for Annotate data, and 'b' is for bounds data. If you specified the base name for the HTML file to be simple.html, the generated files will be as follows:

simple.html contains the HTML and JavaScript to display the map.

simplem.js contains the map data for the GPOLY options.

simplem.kml contains the map data for the KML options.

simplea.js contains the Annotate data for the GPOLY options.

simplea.kml contains the Annotate data for the KML options.

simpleb.js –contains the map bounds based on the input data sets. This is used to center the map and set the zoom.

These files must be moved to your external Web site to be run. _WEBDATAPATH sets the run-time location of these files. If this option is not specified, these files must exist in the same location as the HTML file.

Limitations

The Annotate functions and their options are not fully supported. The support of Annotate functions will change with updates. Currently the following functions are at least partially supported: POLY, POLYCONT, MOVE, DRAW, POINT, SYMBOL, and PIE (360 degrees). POINT creates the default Google Map markers on the map instead of

dots. There are also differences in how the SIZE variable is processed. Google Maps can zoom and PROC GMAP cannot. The size cannot be relative to the displayed area with Google Maps.

INSTALLING AND SETTING IT UP

The SAS Google Map Generator, all associated files, and all examples are available for download in a ZIP file. The file should be unzipped into a working directory on a machine where you have access to SAS. The README.txt file describes how to set up the files. The SCL application is shipped as a PROC CPORT .cpo file. You must run PROC CIMPORT to convert this to a native catalog file. Example data sets shipped as .dpo files must also be converted with PROC CIMPORT.

After you run the generator and create your files, you must move the newly created files and the JavaScript files provided by SAS to your external Web location so you can run the map. Ideally, you will have a local path that points to your Web location. If not, you must physically move the files. It is important that you copy all files to the proper locations, including the JavaScript files provided by SAS.

EXAMPLES

The examples illustrate a number of ways of using the SAS Google Map Generator.

The complete SAS Google Map Generator code and the example code are available for download from http://support.sas.com/rnd/papers. Search for the title of this paper (Google Maps and SAS/GRAPH) on that Web page. The link for downloading is under the title. Future updated versions of the tool and examples will be made available for download on the SAS Maps Online Web site at http://support.sas.com/mapsonline. After installing these files, the examples will be in the directory named 'examples'.

For all of these examples, you must first register your own Google key and install the SAS Google Map Generator and all of the related files. Then you must set the libref GOOGLE to point to the location of the generator. The CreateData.sas program must be run to generate the data needed by the examples. The OptionsReset.sas program should be run between examples to reset the macro variables.

SIMPLE CASE

This example is the simplest case example. It uses the default settings for most of the arguments and uses the minimal number of options. This program will produce GPOLY output for both the map polygons and the Annotate data. You must register your own _GOOGLEKEY as described at the beginning of the paper and use that key in the program. _MAPDATA specifies the modified map data set, and _ANNODATA specifies the Annotate map data set. These are created by running the CreateData.sas program first. _OUTPUTFILE specifies the location where the files will be created. PROC DISPLAY runs the actual generator program. This program creates the output shown in Figure 1.

Here is the syntax:

```
%let _GOOGLEKEY="__GetYourOwnGoogleKey__";
%let _MAPDATA = work.mymap;
%let _ANNODATA = work.anno;
%let _OUTPUTFILE = c:/googlemaps/simple.html;

Proc display c=google.googlemap.creategooglemap.scl; run;
```

This example will create four files in C:/googlemaps. The main HTML file is created as simple.html. This file uses all of the other files and creates the actual map. The map polygon file is simplem.js, the Annotate file is simplea.js and the bounds file (for centering the map) is simpleb.js. Note that the Google Terms of Use requires that all of these files be moved to your external Web site to be used. For this example, all files must be in the same location. You must also copy all of the SAS*.js files provided by SAS to the same location in order to run the map.

See example1.sas in the downloaded files for the complete example.

GENERATING ONLY ANNOTATE DATA AS KML

This example builds on the preceding simple example. In this example, we want the output to be KML instead of GPOLY. In addition, we want only the Annotate data displayed. Note that we specify _OUTPUTTYPE and set it to KML for the output type. In addition, note that we do not specify a map data set with _MAPDATA. You must run OptionsReset.sas between these programs, or the _MAPDATA macro variable will remain set. Figure 4 shows the

output from this program. Note that you can click on a marker and see the balloon with the name, description, and HTML variables from the Annotate data set.

Here is the syntax:

```
%let _GOOGLEKEY="__GetYourOwnGoogleKey__";
%let _ANNODATA = work.anno;
%let _OUTPUTFILE = c:/googlemaps/annokml.html;
%let _OUTPUTTYPE=KML;

Proc display c=google.googlemap.creategooglemap.scl; run;
```

This example will create three files in C:/googlemaps. The main HTML file is created as annokml.html. The Annotate file is annokmla.kml, and the bounds file (for centering the map) is annokmlb.js. Note that all of these files must be moved to your external Web site to be used. For this example, all files must be in the same location. You must also copy all of the SAS*.js files provided by SAS to the same location in order to run the map.

See example2.sas in the downloaded files for the complete example.

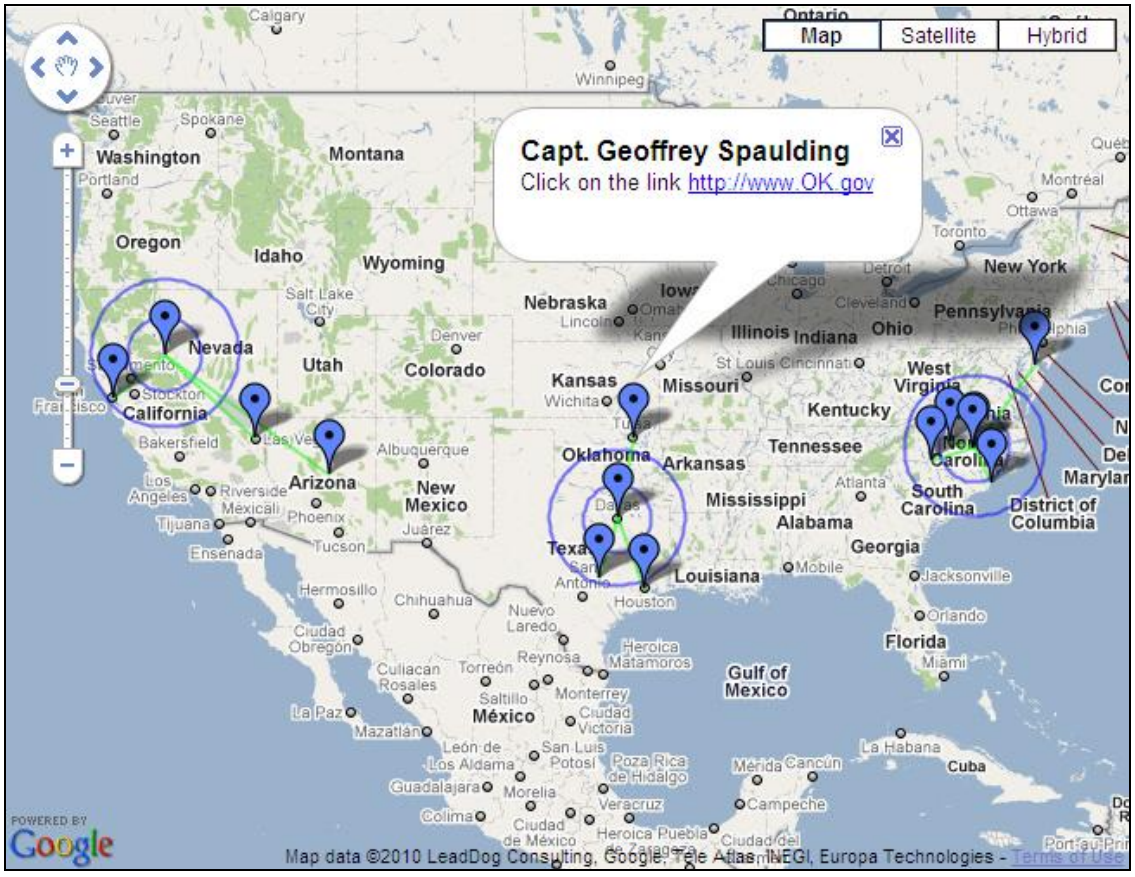

Figure 4.

GENERATING ONLY MAP DATA AS GPOLY

This example builds on the simple example and is very similar to the KML example. In this example, we want the output to be GPOLY. In addition, we want only the map data displayed. Note that we specify _OUTPUTTYPE and set it to GPOLY for the output type. In addition, note that we specify a map data set with _MAPDATA, but do not specify an Annotate data set with _ANNODATA. You must run OptionsReset.sas between these programs to reset the macro variables. Figure 5 shows the output from this program.

Here is the syntax:

```
%let _GOOGLEKEY="__GetYourOwnGoogleKey__";
```

```
%let _MAPDATA = work.mymap;
%let _OUTPUTFILE = c:/googlemaps/mapgpoly.html;
%let _OUTPUTTYPE=GPOLY;

Proc display c=google.googlemap.creategooglemap.scl; run;
```

This example will create three files in C:/googlemaps. The main HTML file is created as mapgpoly.html. The map file is mapgpolym.js, and the bounds file (for centering the map) is mapgpolyb.js. Note that all of these files must be moved to your external Web site to be used. For this example, all files must be in the same location. You must also copy all of the SAS*.js files provided by SAS to the same location in order to run the map.

See example3.sas in the downloaded files for the complete example.

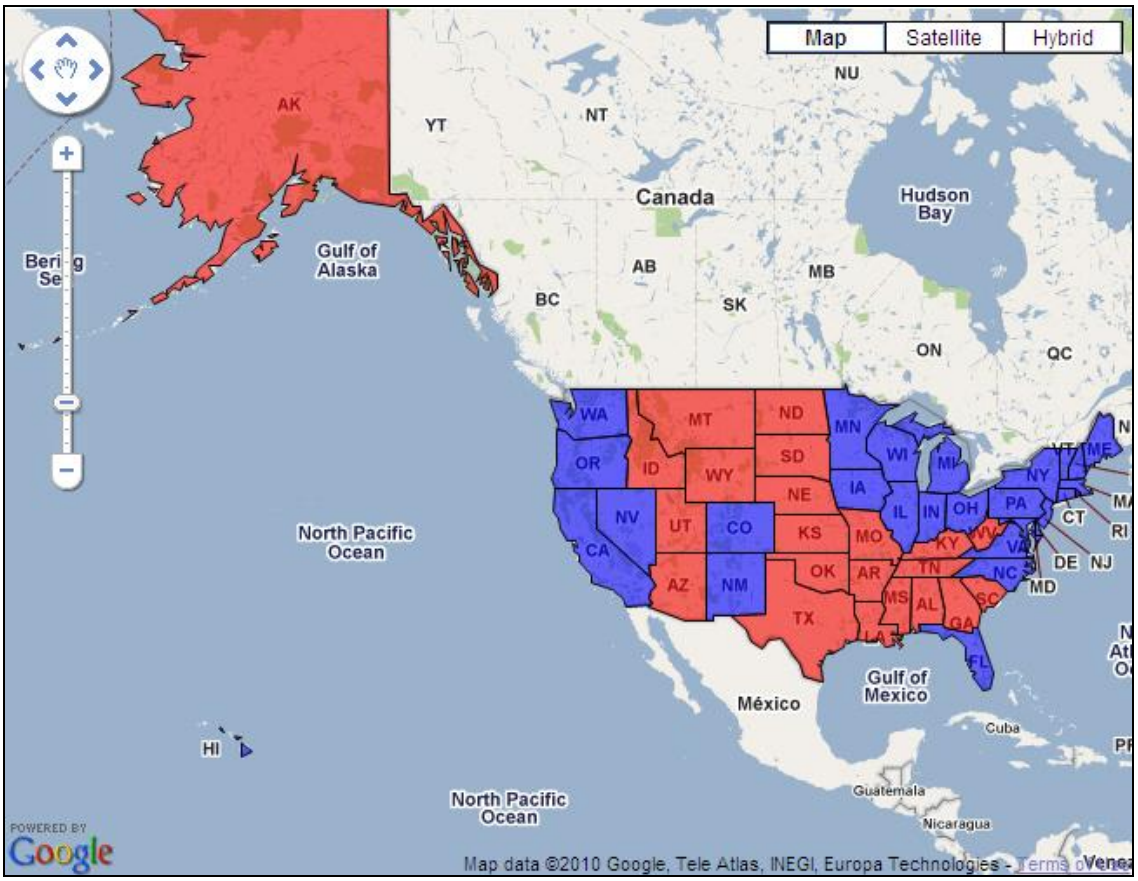

Figure 5.

JUST THE DATA

This example allows you to create only the data and not the HTML file. This is useful if you have already created your HTML and have the map looking the way you want. Now you only need to update the data periodically. The data can be KML or GPOLY output, and it can be map data, Annotate data or both. This example will re-create only the GPOLY map data that can be used with the previous example. Note that we specify _OUTPUTTYPE and set it to GPOLY_ONLY for the output type. To use the new map data with a previously created HTML file, your _OUTPUTFILE filename must match the original HTML file. Note that it still specifies an HTML filename, even though none will be created. You must run OptionsReset.sas between these programs to reset the macro variables.

Here is the syntax:

```
%let _GOOGLEKEY="__GetYourOwnGoogleKey__";
%let _MAPDATA = work.mymap;
%let _OUTPUTFILE = c:/googlemaps/mapgpoly.html;
%let _OUTPUTTYPE=GPOLY_ONLY;
```

```
Proc display c=google.googlemap.creategooglemap.scl; run;
```

This example will create two files in C:/googlemaps. The map file is mapgpolym.js and the bounds file (for centering the map) is mapgpolyb.js. Note that both of these files must be moved to your external Web site to be used. For this example, all files must be in the same location.

See example4.sas in the downloaded files for the complete example.

ADVANCED EXAMPLE

The final example uses many different options to change the behavior of the map, the appearance of the map, and the location of the files. The program will create both map and Annotate GPOLY files. The organization of the files is modified with _OUTPUTFILE, _OUTPUTDATAPATH, _WEBDATAPATH, and _WEBJSPATH. User JavaScript functions are added with the _USERJS option, and user-specified body code is added to the HTML file with the _BODYCODE option. The appearance of the map is changed by turning the Scale on (bottom left corner) with the _SCALE option, turning the Map Type control off (top right) with the _MAPTYPE= option, and setting the map to display with a default type of TERRAIN with the _MAPDEFAULT option. This map is also interactive. There are tooltips displayed over the state polygons, and the opacity of the state color is changed when you hover over it. The user JavaScript and the user body code define buttons on the Web page that allow a click on the polygons to drill down or to select multiple polygons. The multiple polygons selected can be used to do some work. This example code will open a state government Web page on a drill-down and will open multiple state government Web pages when you click **SelectSubmit**. Finally, you can turn off and on the Annotate layer and the polygon layer. You must run OptionsReset.sas between these programs to reset the macro variables. Figures 7 through 9 show the output from this program.

Here is the syntax:

```
%let _GOOGLEKEY="__GetYourOwnGoogleKey__";
%let _MAPDATA = work.mymap;
%let _ANNODATA = work.anno;
%let _OUTPUTFILE = c:\googlemaps\advanced.html;
%let _OUTPUTTYPE=GPOLY_EVENT;
%let _OUTPUTDATAPATH = c:\googlemaps\data\;
%let _WEBDATAPATH=data\;
%let _WEBJSPATH =SASjs\;
%let _USERJS=user\USER.js;
%let _BODYCODE=c:\googlemaps\user\BODYcode.txt;
%let _SCALE=ON;
%let _MAPTYPE=OFF;
%let _MAPDEFAULT=TERRAIN;

Proc display c=google.googlemap.creategooglemap.scl; run;
```

The options specify a particular file structure for the files created and for the run-time files. Figure 6 shows the file structure. The options used to specify the directory path are listed on the right. The boxes that point to the right contain run-time options that specify directory paths used by the HTML file, and the boxes that point to the left contain options that specify directory paths used directly by the SAS Google Map Generator.

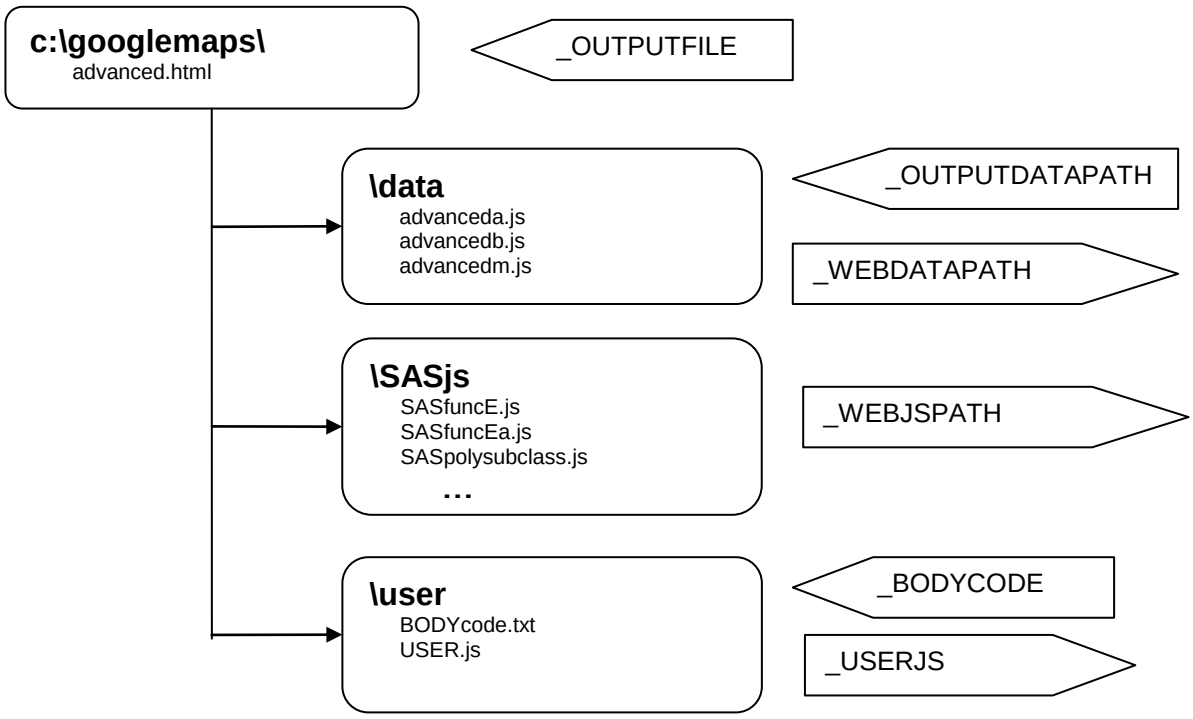

Figure 6.

There are four files created. The main file, advanced.html, is created at C:\googlemaps as specified by the _OUTPUTFILE option. Three files are created in the data subdirectory: advanceda.js, advancedb.js, and advancedm.js. The _OUTPUTDATAPATH option specifies the location where these data files are written. At run time, these files also need to be in a subdirectory named 'data', as specified by the _WEBDATAPATH option. The SASjs subdirectory is needed at run time, as specified by the _WEBJSPATH option. This directory contains all the JavaScript files provided by SAS. The 'user' subdirectory is needed at creation time and run time. The _BODYCODE option specifies the location of the BODYcode.txt file to be used at creation time. The _USERJS option specifies the location of any user-created JavaScript that is used at run time. Using these options allows you to organize your data and map related files on your Web site. To run this map, the entire tree structure from 'C:\googlemaps and below must be moved to the external Web site.

Figure 7 shows the initial state when the map first appears. Notice that the Scale control appears in the lower left corner and that the Map Type control is missing from the upper right corner. In addition, notice that the map is a TERRAIN map.

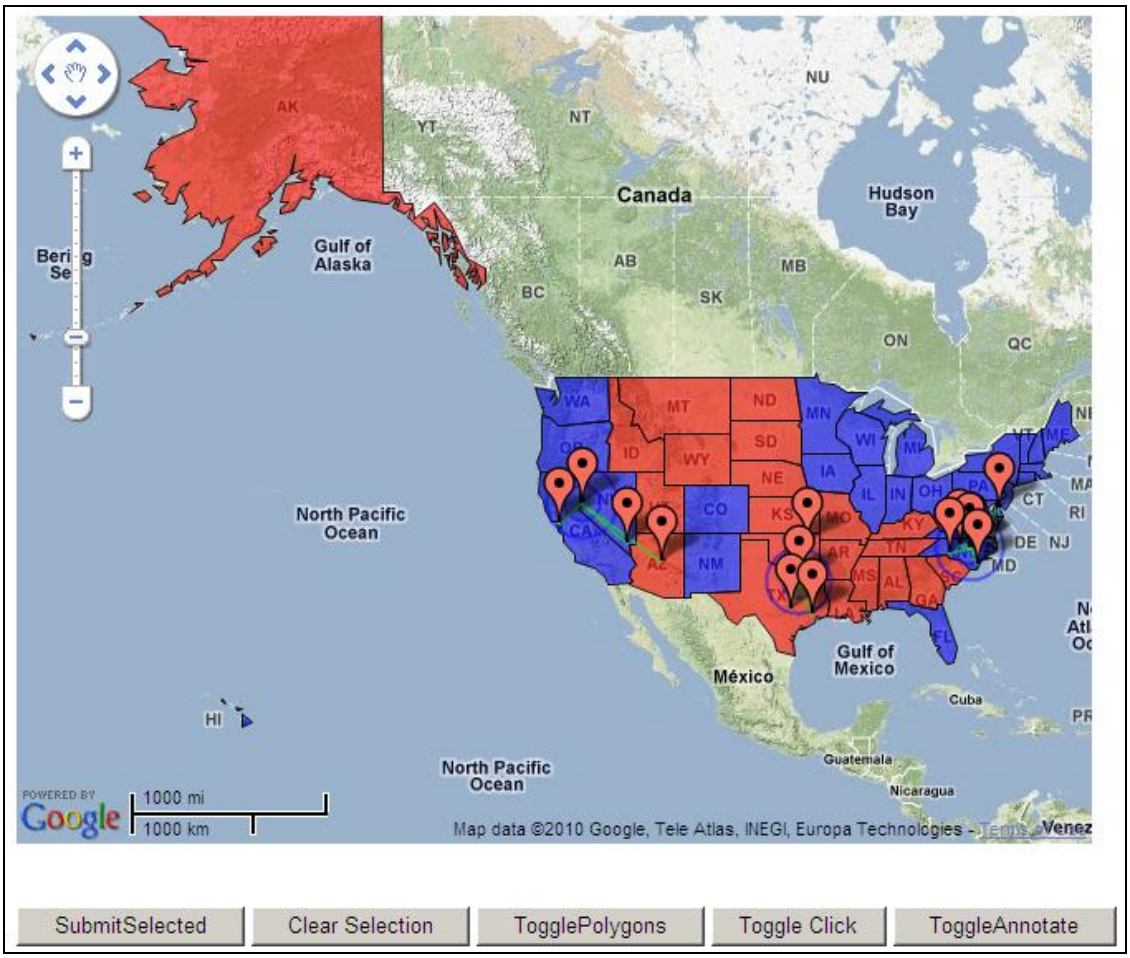

Figure 7.

Figure 8 shows the interactive nature of the map. It illustrates how you can select multiple states. Clicking the **SubmitSelected** button will run user JavaScript code that will open a Web browser for each selected state with the state government Web page.

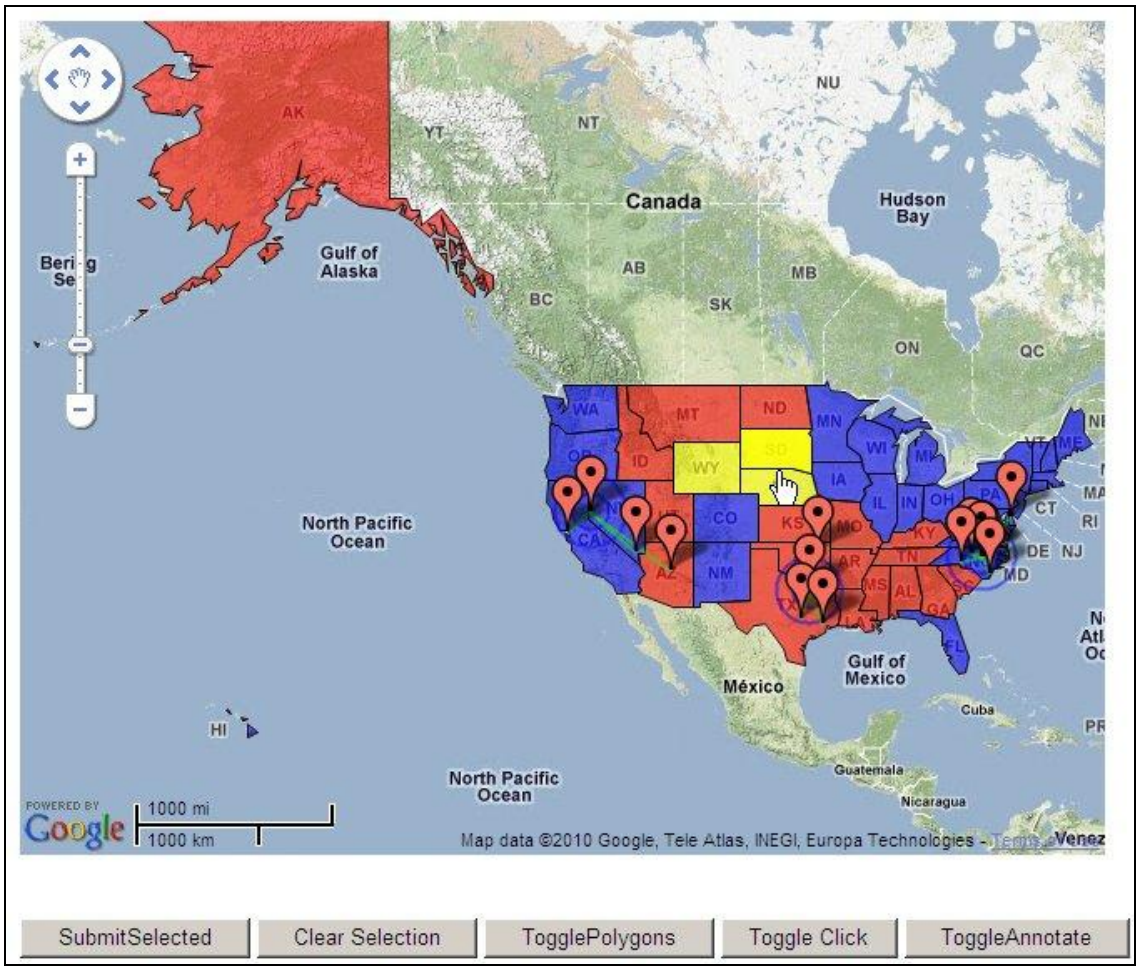

Figure 8.

Figure 9 illustrates clicking the **TogglePolygons** button. The state polygons are turned off. Clicking again will redraw the polygons.

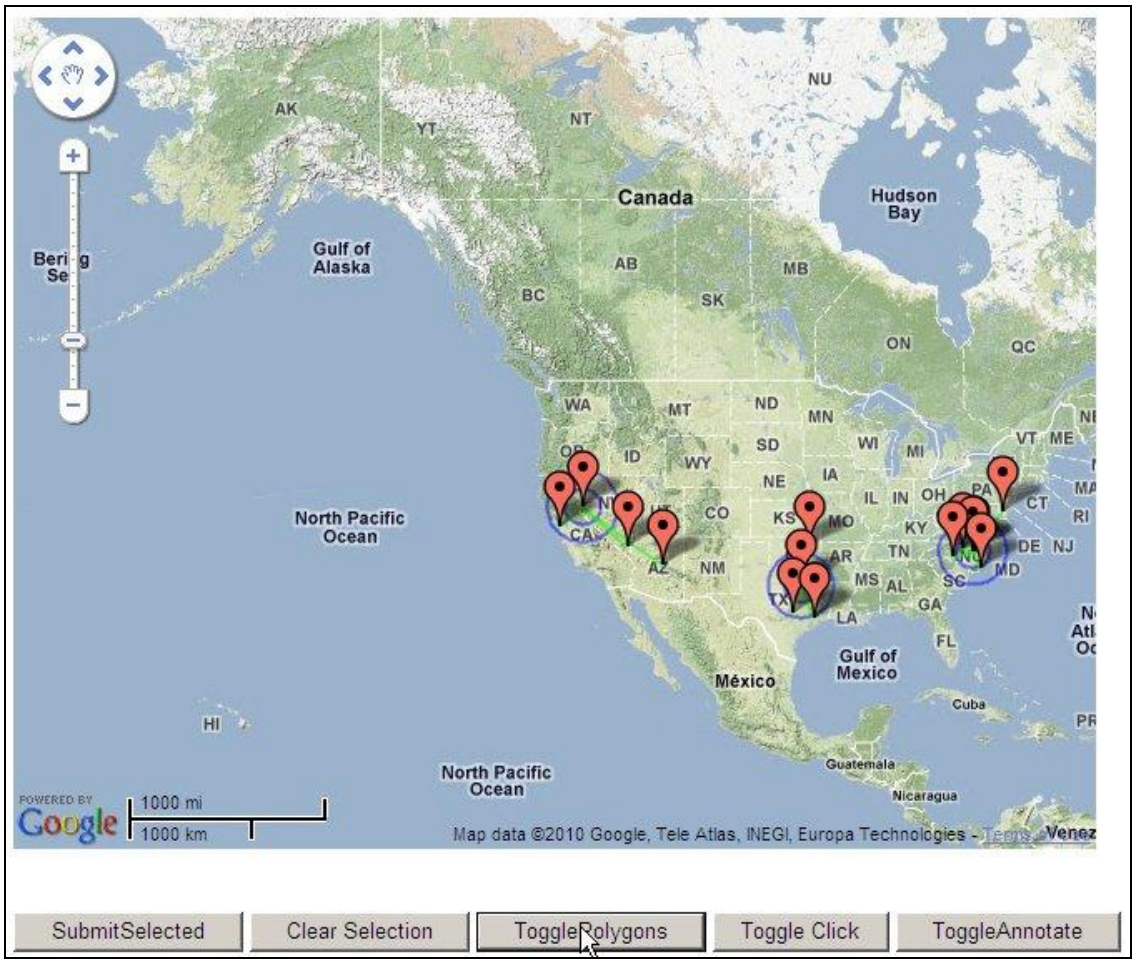

Figure 9.

See example5.sas in the downloaded files for the complete example.

CONCLUSION

The SAS Google Map Generator is a powerful and easy-to-use tool for creating a Google Map with SAS data. The tool has a great deal of flexibility to allow you to tailor your maps in order to meet your Google mapping needs.

REFERENCES

SAS Institute Inc. 2010. SAS/GRAPH 9.2 documentation in the Knowledge Base. http://support.sas.com/documentation/onlinedoc/graph/index.html.

RESOURCES

"Google Maps and SAS/GRAPH." SAS Global Forum 2010 paper and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

"PROC GEOCODE: Now with Street-Level Geocoding." SAS Global Forum 2010 paper and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

"PROC GEOCODE: Creating Map Locations from Your Data." SAS Global Forum 2009 paper and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

"Tips and Tricks IV: More SAS/Graph Map Secrets." SAS Global Forum 2009 paper and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers.

"Tips and Tricks III: More Unique SAS/GRAPH Maps." SUGI 30 SAS Presents handout and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

"Tips and Tricks II: Getting the most from your SAS/GRAPH maps." SUGI 29 SAS Presents handout and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

"SAS Mapping: Technologies, Techniques, Tips and Tricks." SUGI 28 SAS Presents handout and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

SAS Global Forum Proceedings: http://support.sas.com/events/sasglobalforum/previous/index.html.

SAS Maps Online Web Site: http://support.sas.com/mapsonline.

SAS Customer Support: http://support.sas.com/.

SAS Online Product Documentation in the Knowledge Base: http://support.sas.com/documentation/index.html.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Darrell Massengill
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Darrell.Massengill@sas.com

Please send any comments and suggestions on the SAS Google Map Generator to mapping@sas.com.