"Pushing SAS/AF® and FRAME Entries in MVS to the Limit: The USEPA's AIRS Graphics System"

Thomas E. Link, U.S. Environmental Protection Agency, RTP, NC
M. Arthur Alexander III, SAS Institute, Cary, NC

First Of All, What Is "AIRS"?

The U.S. Environmental Protection Agency (USEPA) depends on information -- lots of it -- to carry out its mission to protect and improve the nation's air quality. EPA's main collection of computerized information about air pollution is AIRS, the Aerometric Information Retrieval System. To paraphrase the late Carl Sagan (who probably wished that he had never said it), the AIRS database contains literally billions and billions of values. These myriad numbers and words characterize the state of our air.

The values in the AIRS database tell how much pollution is released into the air by industry, commerce, and just plain folks. They tell the ambient concentration of pollutants in the air we breathe, and they tell what administrative and legal actions have been taken to fix pollution problems. AIRS gives the 50 states and many local environmental agencies direct access to all the AIRS data, and direct control of their own data. This means the states do not need to develop their own computer systems for reporting air pollution data to EPA. By analyzing the values in AIRS, EPA and the states can determine the "health" of our air, where the problems are, and how well remedial actions are working. All of these are good things.

But finding the right values among the billions, and sifting through them to find out what they mean can be daunting. AIRS has a vast array of interactive and batch data reporting capabilities. They range in complexity from easy-to-read, succinct reports of summary data, to unformatted data files containing thousands of characters in every one of its thousands of records. The AIRS database reports are designed to provide the detailed air pollution data that environmental analysts and statisticians need, but the reports can produce some very tall stacks of computer printout. Sometimes they have to generate one pile of printout just to ask the next question, which may lead to yet another pile of printout. (We use the term "printout" figuratively; many analysts view the reports on a terminal screen instead of converting forests into paper. But the point is the same: there's a LOT of chaff among the grain.)

And Why Does It Need Graphics?

Those individuals who use AIRS infrequently or began recently may find the infrastructure of AIRS a bit confusing, even overwhelming. Finding the right path through the many "rooms" and "gardens" of AIRS can be challenging. People who are whizzes at Lotus 1-2-3 and FreeLance may never have heard of these "strange" mainframe terms like NATURAL, TSO, CICS, VTAM, and LOGMODE. Is there help for these poor souls?

Yes, there is! AIRS Graphics (AG) can help both novice and experienced AIRS database users find the proverbial needle in the haystack.

Inexperienced database users use AG to browse through the database visually looking for certain kinds of monitors and plants with certain characteristics. The experienced user uses AG to quickly pinpoint a certain type or range of data and then returns to the AIRS database proper to request the most detailed data listing or extraction available. In both cases would you rather look at 1000 pages of computer printout when it arrives tomorrow, or create a national map now, in 5 minutes, which depicts where 4500 Volatile Organic Compound emission sources are located? You decide.

Why Did We Start Building the System with SAS?

The goal of the AIRS Graphics tool is to provide AIRS users with the ability to access and inspect selected AIRS data graphically, online and on demand. Since we live in the real world, our design/programming team immediately recognized that there were many ways to provide this capability to our 2000 user network; and most were not available to us. AIRS Graphics was developed for our users, and in a development environment of various software, hardware and programming constraints. We asked our users and ourselves lots of questions. Some of the questions were about:

<u>Hardware:</u> Is the existing distribution of hardware sufficient to justify an interactive graphics display system? Graphics devices are expensive resources, and may not be readily available below the level of national and regional office. If the hardware is not in place below the regional level, then the system is

basically going to emerge as an Executive Information System for national and regional management.

Of course, this graphics system is intended to serve for years to come, and it is likely that as older terminals go through their life cycle and are replaced, the concentration of graphics terminals will increase. This would make for an interactive system with a steadily increasing utilization rate over time.

The presence of graphics display terminals, even at today's level, argues for the use of an interactive system. At the same time, the presence of offices with no graphics terminals makes a case for a batch capability. Perhaps an interactive system with batch capabilities will be desirable so that all offices can benefit.

Software: In designing the software for AIRS Graphics, we wanted an interface that would allow those who knew little about computers to interact freely with the data. We knew that we would need the flexibility to construct custom menus and generate complex graphics. We also had huge databases of varying formats that needed to be manipulated in real time, since we wanted to allow the user various parameter selections to subset the data used to produce the finished graphic. Since we were designing a system for the future, we wanted a system with the ability to evolve as computer graphics hardware improves. After surveying the capabilities of products such as ARCINFO, Telegraf, IBM GDDM, and SAS on various platforms, we determined that the SAS system fulfilled our requirements best and was readily available on the current hardware platform.

The modules necessary for the AIRS Graphics system were SAS/AF, SAS/GRAPH and the Base SAS product. SAS/AF is a powerful, customizable menuing facility that contains a complete programming language called SCL (Screen Control Language). With SAS/AF the programmer can custom design a menu screen with complete control over screen colors and the placement of data fields and informational labels. SAS/GRAPH has a variety of map and chart-based, data-driven graphics procedures that are easily customized and enhanced. It also supports virtually any graphics device available in today's market. The Base product has all the data handling and manipulation tools necessary to interact with the existing databases. The combination of these products, being part of the same overall system, integrates seamlessly to allow the programmer maximum flexibility in generating a system tailored to the needs of the end user.

What The Users Want: It was assumed that they really wanted a high-powered, interactive system. While this is probably true of the regional and national offices, it may or may not be true of the lower level offices. Many managers may prefer to work with batch hardcopies, especially those who still feel uncomfortable at a terminal. There are always times when hardcopies are essential, so even an interactive display system should have the capability of generating hardcopy outputs when necessary.

However, we have here the same factor that was present in the hardware discussion, namely the force of change. Managers are getting increasingly comfortable in front of the terminal and that trend is certain to continue. The emerging generation of management is likely to demand on-line capability, and may not be satisfied with a batch system designed to meet the needs of a less-demanding group.

System Needs Vs. Desires: This is an issue common to most major projects. Resource considerations dictate decisions, and AIRS Graphics was no exception. We identified the minimum acceptable capabilities of a graphics system and began work. As the system went into production, users asked for changes and enhancements. Our philosophy has been to provide the program additions and enhancements that will benefit and be used by the largest percentage of our user community.

Has AIRS Graphics worked for users?

Four years since it's initial May 1993 production date, with more than 1000 registered users of AG worldwide, who have submitted more than a million data queries and produced over 100,000 actual graphic plots--we think, yes--it has and is still working. We’re still receiving comments and suggestions for improvements, new bells and whistles.

How Do You Keep The Music Playing?

Users always want NEW, EASIER, FASTER--they want "Point 'N' Click"--even in a mainframe environment. Within the constraints of our closed operating environment, and the USEPA's nationally supported software packages, how could we provide this additional functionality?

AIRS Graphics "Point 'N' Click"

To Frame Or Not To Frame? -- Version 6.08 of the SAS System introduced SAS/AF FRAME Entries. Many SAS developers mistakenly believe that FRAME Entries do not work on MVS. This misconception is the result of two fundamental problems. First, to build a FRAME entry, the programmer's display device must be capable of displaying vector graphics. If attempted on a character-based device, the frame build display will not open. Second, the earliest releases of version 6.08 (before TS415) had several annoying refresh problems, which made extensive use of frames unacceptable. Even developers who used vector graphics capable devices, but who created frame entries with certain object combinations in TS405 or TS410, might have abandoned frame entries believing them to be incompatible with MVS.

Even with the early problems, the capabilities that FRAME entries could add to our AF program based system were too good to ignore. In a frame entry, we would be able to place extended tables anywhere on the screen, not just at the bottom as in program entries. In addition, multiple extended tables could be used on the same screen. Another giant leap over the program entry was the ability to display graphical output on the same screen as text elements such as entry fields and extended tables. These two possibilities alone, made us interested in exploring the usage of frame entries in spite of the initial problems.

Our very first use of a FRAME entry in the AIRS Graphics system came when we decided to introduce the option of allowing users to make multiple county selections as a data subset item. It is easier for the user to visualize the desired county cluster subset by selecting the counties from a map than from a list of county names. This first frame displayed a table of county names alongside a map of the counties displayed in the list. The user still selected the subset from the text list, but did not have to guess as to the geographic clustering of the output selected. If only we could let the user click on the map itself to make selections instead of typing in county numbers or selecting items from a list.

Where No Hotspot Has Gone Before

Screen objects that return information about the object when the object is 'selected' are called hotspots. The SAS/Graph Output Object allows the programmer to create 'segment hotspots' for polygons in a static map. So, you could create a selectable U.S. map of the states and store it in a catalog. Next you could interactively define a hotspot for each state polygon on the map. Finally, if you never change the map, when a user selects a state from the map, you can programmatically return the name of the hotspot using the _get_value_ method. But what if the map is generated dynamically and has the potential to change depending on the user's selection? What if you want to add annotation, such as markers for specific sites? Clearly static hotspots will not suffice here. If we have the ability to create dynamic maps, we need the ability to generate dynamic hotspots as well.

The FRAME Graphics Object has this ability for graphs and charts. Geographic Information Systems (GIS) have this ability for maps. Why couldn't we do it for maps created with PROC GMAP and displayed in the SAS/Graph Output Object? The input to PROC GMAP is a response data set and a map data set. Somehow, GMAP combines these two sources to produce an output stream, which SAS can then display as a map on which the data information is overlaid. Since the unknown in this process was the means by which SAS combined the two files into one, and since the resulting file had meaning only in the context of generating the graphical output, we had to approach the process as a 'black box'. We had input which we could study in detail. We had output for which we had very little information. Was there anything in the output stream file, which could be used to tie the polygon information back to the data which created it?

It took many hours of experimentation, of backing up and coming back at the problem from different angles, but with persistence and some key bits of information supplied by Bill Powers and Jeff Cartier of SAS Institute, we were able to crack the code. We have already mentioned the _get_value_ method, which gives us hotspot information, but we have no hotspots. The only other SAS/Graph Output Object method, which gives us any information when the object is selected, is _get_info_. Frankly, at first glance, _get_info_ didn't seem to help much. When a point on a graphic is selected, the _get_info_ method returns an SCL list, which contains the following information related to the selected point on the graphic:

Item	Description
X	the X coordinate of the selection in pixels
Y	the Y coordinate of the selection in pixels
TEXT	the text of the graph segment, if the selected point contains text
SPOT	a list defining a selected hotspot, if the selected point is a hotspot
SPOTID	the number of the graph segment within the graph (beginning with 1)
SPOTTYPE	a number identifying the type of graph segment selected

Since the X and Y values are related to the graph's position on the screen, those values are of little value to us. The TEXT value could be of some value if we wanted to use a label of some kind. SPOT is related to segment hotspots, and is of no value since we don't have any. SPOTTYPE could help us programmatically be sure that a polygon was selected. All that is left is SPOTID, the number of the graph segment within the graph, beginning with 1. This seems to imply that graphical output is a sequential flow of graphics segments beginning at 1. If the output segment has some predictable and controllable ordering, it might be possible to tie the order back to the data used to generate the output.

To make a long story short, it worked. The items in the output stream are predictable. For GMAP output, if a graphic has a border, the border is segment #1, followed by titles, footnotes, etc. and then the map polygons, always in the same order. So by strictly controlling the number of elements which precede the map polygons, and skipping over those segments (spotids) we can sequentially identify the map polygons from the SPOTID returned by _get_info_. If we make sure that the data has the same ordering as the map polygon data, we can use the segment value to lookup the corresponding data value from the response data set. From the standpoint of United States maps, the spotid of any map polygon can be tied back to the FIPS State and FIPS County codes. Once these values are known, any data that is indexed by state and county is then available to the programmer, all by clicking on the desired area.

We were excited and feeling pretty smart. However, human nature being what it is, we wanted more. What if we had markers on the map? If we could select a single marker and identify the specific site related to that marker, we would be approaching the functionality of a GIS (and in our own minds approaching the status of genius). Could we add annotated markers to our graph without upsetting the predictability of our output data file? Again, the answer is 'Yes'. The annotated markers are drawn in order immediately after the last map polygon is generated. So, by subtracting the title, footnote, etc. segments AND the map polygon segments from the SPOTID, the sequential ordering of the annotated markers could be predicted. The annotate data set is in the same order as the marker segments, so the first annotated marker references the first record in the annotate data set.

We were on a roll. We already knew that the Graphics Object would give us dynamic hotspotting for graphical data. We also knew that individual fields and rows were selectable for extended table data. With the addition of this method for dynamic hotspot simulation on maps, the pieces for the new AIRS Graphics "Point 'N' Click" system was beginning to fall into place.

"Point 'N' Click" Design

The system would be composed of three main data reporting areas for summary data: tabular form, bar charts and choropleth maps. In additional to national, state and county scales, the USEPA groups states into ten geographic regions. It therefore made sense to construct the three data display paths in a hierarchical fashion with this geographic relationship in mind. The user, starting at the national level, by selecting a row from a table, a bar on a chart, or a polygon on a map could 'drill down' to the next lower geographic level using the same display type.

The AIRS data, which we would be using, consists primarily of two databases: one for plant emission data and the other for monitoring data. Within each of these databases are values for seven pollutants called 'Criteria Pollutants'. We wanted the user to be able to select a database, pick any or all of these pollutants and display the data in either of the three display formats. We also wanted the flexibility to switch to any geographic level and the ability to change from tables to maps to graphs from any point in the system without losing the selected data subset. This meant that we had to be able to jump from a frame in a given hierarchy to any other hierarchy without having to return to the previous frame or to some main menu in between.

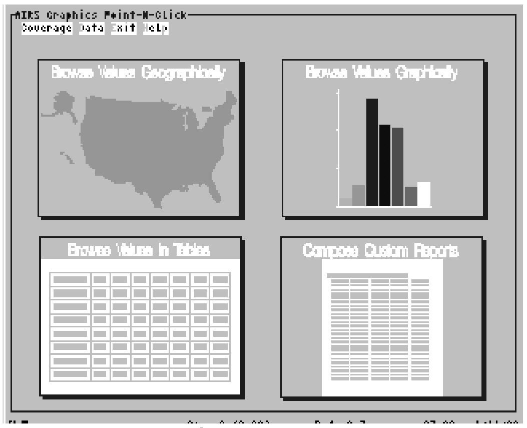
The ability to select a data subset from any screen in the system was accomplished by adding a system of pull down menus to each frame. Pmenu selections would invoke frames, which would allow database selection, pollutant selection, geographic coverage selection and display format selection from lists, radio boxes and check boxes. The final selections would be stored in the Global Environment List and thereby be available to all frames at any point in the process.

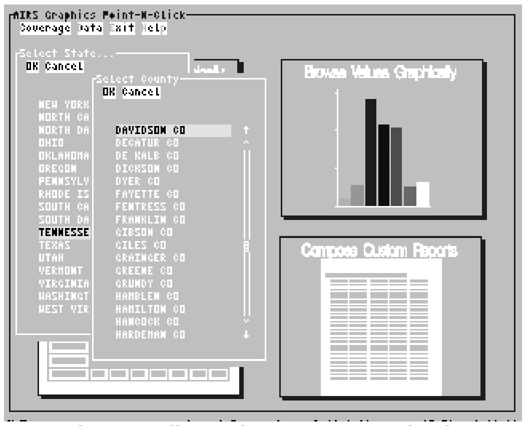
Since we wanted to be able to jump to any frame from any other frame in the system, closing down each frame before we jumped, *Call Display* would not do. *Call GOTO* seemed the likely mechanism, but *Call GOTO* did not allow parameter passing. Reason number 2 for using the Global Environment List.

In addition to pmenus, each frame would have radio buttons for single-click reselection of pollutant subset and update of the display. Icons would likewise allow a single-click jump from map to chart to table to map using the same geographical and data subset. Selection of a table row, a graph bar or a map polygon would take the users to the next geographic subset. Just before the jump to the next frame, the altered subset is written to the Global Environment List. When the next frame is invoked, the Global

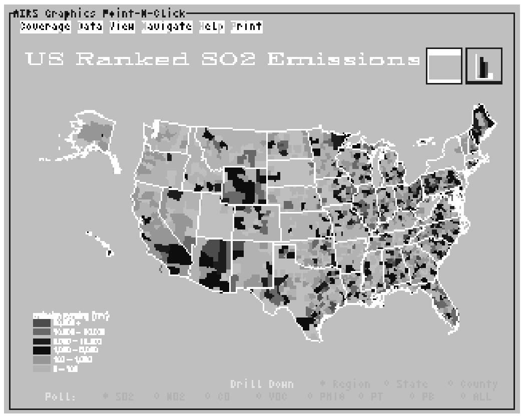
Environment List is read and the necessary where clauses constructed to display the representative data subset.

The result is a system with Graphical User Interface running on the mainframe. The user can browse AIRS plant and monitor data, subset it, jump from one display type to another, from a national level all the way to a single site without ever touching the keyboard; one click at a time.

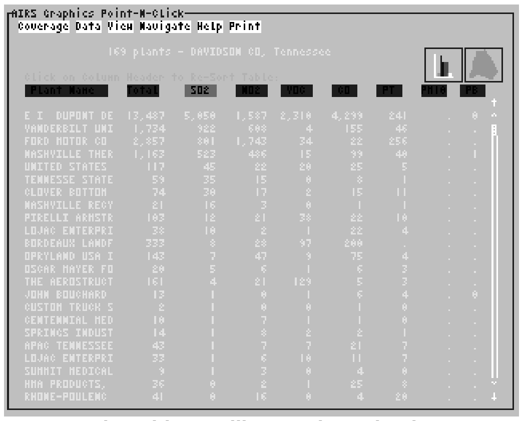

AIRS Graphics Point-N-Click
Coverage Data Exit Help
Browse Values Geographically
Browse Values Graphically
Browse Values in Tables
Compose Custom Reports
Main Menu Screen



Main Menu Illustrating Coverage Selection



US Map – Drill Down by Selecting Any Polygon

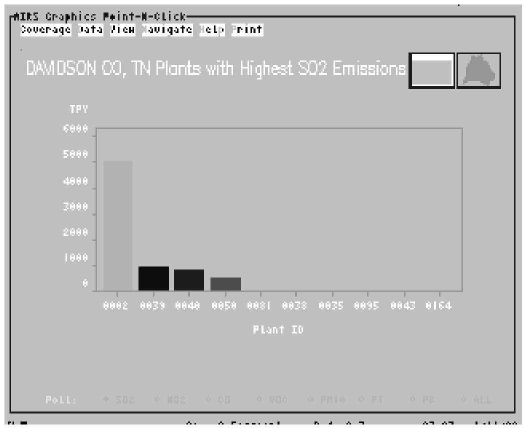

AIRS Graphics Point-N-Click
Coverage Data View Navigate Help Print

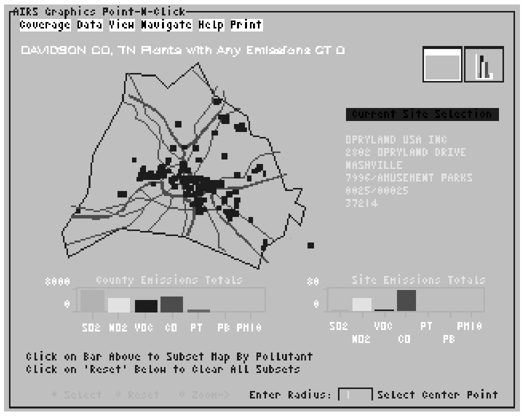
169 plants – DAVIDSON CO, Tennessee

Click on Column Header to Re-Sort Table:

Plant Name	Total	SO2	NO2	VOC	CO	PT	PM10	PB
E I DUPONT DE	13,487	5,050	1,587	2,310	4,299	241	.	0
VANDERBILT UNI	1,734	922	608	4	155	46	.	.
FORD MOTOR CO	2,857	801	1,743	34	22	256	.	.
NASHVILLE THER	1,163	523	486	15	99	40	.	1
UNITED STATES	117	45	22	20	25	5	.	.
TENNESSE STATE	59	35	15	0	8	1	.	.
CLOVER BOTTOM	74	30	17	2	15	11	.	.
NASHVILLE RECY	21	16	3	0	1	1	.	.
PIRELLI ARMSTR	103	12	21	38	22	10	.	.
LOJAC ENTERPRI	38	10	2	1	22	4	.	.
BORDEAUX LANDF	333	8	28	97	200	.	.	.
OPRYLAND USA I	143	7	47	9	75	4	.	.
OSCAR MAYER FO	20	5	6	1	6	3	.	.
THE AEROSTRUCT	161	4	21	129	5	3	.	.
JOHN BOUCHARD	13	1	0	1	6	4	.	0
CUSTOM TRUCK S	2	1	0	0	1	0	.	.
CENTENNIAL MED	10	1	7	1	1	0	.	.
SPRINGS INDUST	14	1	8	2	2	1	.	.
APAC TENNESSEE	43	1	7	7	21	7	.	.
LOJAC ENTERPRI	33	1	6	10	11	7	.	.
SUMMIT MEDICAL	9	1	3	0	4	0	.	.
HMA PRODUCTS,	36	0	2	1	25	8	.	.
RHONE-POULENC	41	0	16	0	4	20	.	.

Example Table – Drill Down by Selecting Row
Change View by Selecting Bar or Map Icon


AIRS Graphics Point-N-Click
Coverage Data View Navigate Help Print
DAVIDSON CO, TN Plants with Highest SO2 Emissions
TPY
6000
5000
4000
3000
2000
1000
0
0002
0039
0040
0050
0081
0038
0035
0095
0043
0164
Plant ID
Poll:
SO2
NO2
CO
VOC
PM10
PT
PB
ALL
Bar Chart View of Same Data
Drill Down by Selecting Bar


AIRS Graphics Point-N-Click
Coverage Data View Navigate Help Print
DAVIDSON CO, TN Plants with Any Emissions GT 0
Current Site Selection
8000
0
County Emissions Totals
SO2 NO2 VOC CO PT PB PM10
80
0
Site Emissions Totals
SO2 NO2 VOC CO PT PB PM10
Select
Reset
Zoom->
Enter Radius:
Select Center Point

OPRYLAND USA INC
2802 OPRYLAND DRIVE
NASHVILLE
7996/AMUSEMENT PARKS
0025/00025
37214

Click on Bar Above to Subset Map By Pollutant
Click on 'Reset' Below to Clear All Subsets

Map View of Same Data
Marker Selection Populates Table with Detail

What Will AIRS Graphics Offer in The Future?

We know that AIRS Graphics doesn't do everything and will not provide the detail or 'horsepower' that some users will need. We also know that allowing a user too free a hand can result in less than meaningful graphs, misleading reports, or reams of unwanted output. We have purposefully limited our users to a selection of graph types and available data. AIRS Graphics doesn't try to be everything to 100% of our users; AG tries to be 75% of what 75% of our users want. The other 25% are what the future of AG is about.

AIRS Graphics will evolve to meet changing needs of the users and new capabilities of the base software. As mentioned earlier AIRS Graphics is currently available to all 50 States. Efforts are underway to assist local air pollution control agencies at the county level with access to AIRS -- potentially 3300 new user organizations. International air pollution data from the World Health Organization will continue to necessitate new international geographic coverages and solutions to specialized mapping problems.

Being resourceful and innovative within the hardware/software constraints of yesterday inspired additional system functionality made possible with "Point 'N' Click". But with new products, faster processors, lower communications costs, and worldwide demand for air pollution data, we're working on new solutions and systems.

The AIRS Graphics Re-engineering Project--now underway--will migrate the entire system to UNIX and allow for Internet access through any Web browser. The new SAS/IntrNet product will figure prominently in this new incarnation of AIRS Graphics, and allow public access to EPA's mother of all air pollution databases--AIRS--in graphic mode, and in ways which were impossible, even six months ago.

References:

Appendix

An example of the algorithm used to simulate dynamic hotspots on the AIRS Graphics “Point ‘N’ Click” Maps is contained in the following FRAME screen along with the SCL code labeled Mapexamp.scl. The frame consists of a SAS/Graph Output Object named ‘map’.

Figure 1 illustrates the frame as it is first displayed, with nothing selected.

Figure 2 illustrates the selection of a blank area within a county polygon.

Figure 3 illustrates the selection of a marker representing a single plant whose coordinates are unique in the data set.

Figure 4 illustrates the selection of a marker representing more than one plant with duplicate coordinates.

Algorithm Notes:

Since the EPA data has unprojected coordinates, we used the unprojected SAS map data from maps.counties. The Density parameter is used to reduce the number of points used in each polygon. A density of 3 and under is usually sufficient at this resolution and helps the map render faster than using all densities.

Parts of the Submit Block in the init section of the code is placed there for illustrative purposes. In a production system, the frequency data related to the map polygons would be run previously and stored in saved SCL lists to improve response time.

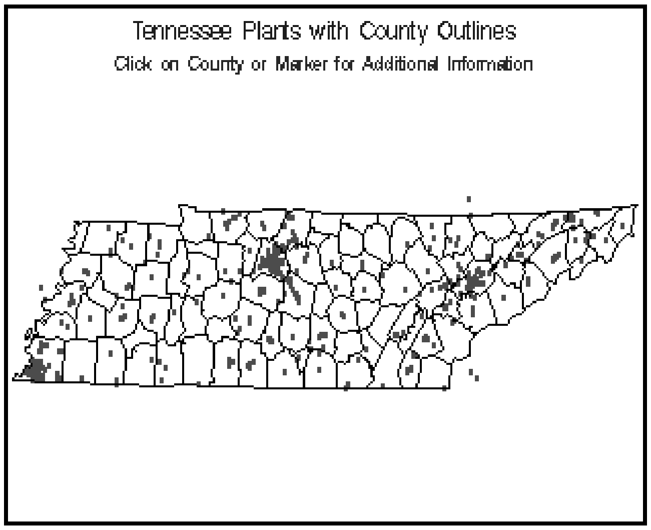

Tennessee Plants with County Outlines
Click on County or Marker for Additional Information

Figure 1.

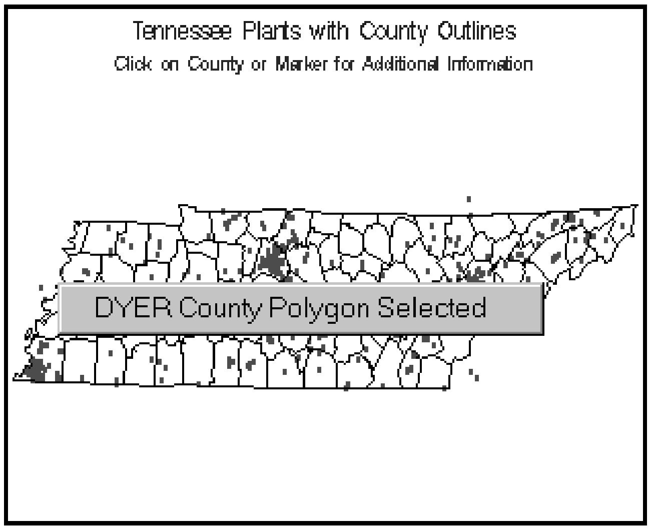

Tennessee Plants with County Outlines
Click on County or Marker for Additional Information
DYER County Polygon Selected

Figure 2.

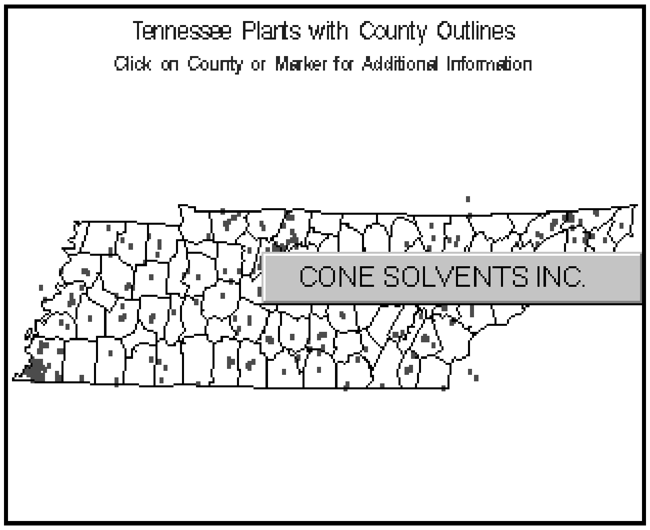

Tennessee Plants with County Outlines
Click on County or Marker for Additional Information
CONE SOLVENTS INC.

Figure 3.

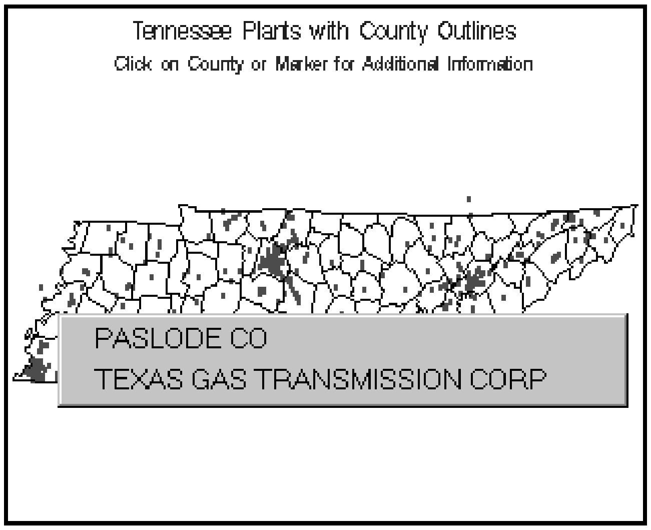

Tennessee Plants with County Outlines
Click on County or Marker for Additional Information
PASLODE CO
TEXAS GAS TRANSMISSION CORP

Figure 4.

Only the plant name is returned in this example. However, it should be evident that any information from the corresponding data set could be returned, as well as, information in related data sets which are keyed by any of the unique data values retrieved from this data set.

In addition, it is a simple task to ‘drill down’ to any county level once the state and county FIPS codes are known. A similar algorithm could be applied to the county data to display the county data and make the resulting markers selectable.

```
MAPSAMP.SCL:

length spotid index dupcoord 8;
init:
  rc=rc;
  dupwhr=dupwhr;
  whrc=whrc;
  submit continue;
    libname sugi 'e:\sugi';
    goptions gunit=pct device=win;
    data tennmap;
     retain dtype 'MAP';
     set maps.counties(where=(state =47 and density le 3));
    run;
/*  determine the number of points per segment for each county */
    proc freq data=tennmap;
      tables county*segment / noprint out=ctysegs;
    run;
/* discard any segments with only one point per segment */
    data ctysegs;
      set ctysegs;
      where count>1;
    run;
/* prepare to remove plants with duplicate coordinates  */
/* from the annotate file and store them in separate    */
/* files to use for lookup later on.                    */
    proc sort data=sugi.plants(where=(state=47))
              out=plbydups;by dupcoord;run;
/* create a primary annotate file (which contains one   */
/* point for each xy coodinate pair in the data set),   */
/* a duplicate site data set, and a data set for plants */
/* records with missing coordinates.                    */
    data annoplt pltdups plnocrds;
     length dtype $4 xsys ysys hsys position when $1
            color function style text $8 size x y 8;
     set plbydups;
     by dupcoord;
     function='symbol';
     style='marker';
     text='U';
     xsys='2';ysys='2';hsys='3';position='5';
     size=1;
     when='B'; * to be able to see the county map lines;
     color='green';
     dtype='PLT';
     if xycoord='N' then output plnocrds;
     else do;
        if dupcoord=0 then output annoplt;
        else do;
          if first.dupcoord then do;
             output annoplt;
             output pltdups;
          end;
          else output pltdups;
        end;
     end;
    run;
/* combine the map data set and the annotate data set   */
/* to project all points together.                      */
    data all;
      set tennmap annoplt;
    run;
/* project the unprojected data                         */
    proc gproject data=all out=allout;
      id state county segment;
    run;
/* replace any previous graphics in the catalog         */
    goptions goutmode=replace nodisplay;

    pattern v=e c=black r=99999;

    title f=swiss h=5 c=blue
          'Tennessee Plants with County Outlines';
    title2 f=swiss h=4 c=Red
          'Click on County or Marker for Additional Information';

/* create the empty map with annotated markers          */
    proc gmap data=allout(where=(dtype='MAP'))
               map=allout(where=(dtype='MAP')) all;
      id state county segment;
      choro county/nolegend name='plantmap'
      annotate=allout(where=(dtype='PLT'));
    run;
    quit;

/* reset goptions to default FRAME entry mode           */
    goptions display goutmode=append;
  endsubmit;

/* store the map polygon information currently stored   */
/* in ctysegs in an scl list for retrieval when areas   */
/* of the map are selected.                             */
  clid=makelist();
  dsid=open('ctysegs');
  do index=1 to attrn(dsid,'nlobs');
    rc=fetchobs(dsid,index);
    clid=insertn(clid,getvarn(dsid,1),-1);
  end;
  dsid=close(dsid);
  call notify('map','_set_graph_','work.gseg.plantmap.grseg');
return;

map:
  call notify('map','_get_info_',infoid);
  displist=makelist();
  spotid=getnitemn(infoid,'spotid');
/* A marker is selected */
  if spotid gt listlen(clid)+2 then do;
    dsid=open('annoplt');
    index=spotid-listlen(clid)-2; * 2 titles;
    rc=fetchobs(dsid,index);
    dupcoord=getvarn(dsid,varnum(dsid,'dupcoord'));
    if (dupcoord) then do;
      dupid=open('pltdups');
      dupwhr=where(dupid,'dupcoord='||put(dupcoord,4.));
      do while(fetch(dupid)=0);
        displist=insertc(displist,
                   getvarc(dupid,varnum(dupid,'plt_name')));
      end;
    end;
    else
      displist=insertc(displist,
                 getvarc(dsid,varnum(dsid,'plt_name')));
    if dupid then dupid=close(dupid);
    if dsid then dsid=close(dsid);
  end;
/* A county polygon is selected */
  else if spotid gt 2 then do;
    county=getitemn(clid,spotid-2);
    ctid=open('maps.cntyname');
    whrc=where(ctid,'state=47 and county='||put(county,3.));
    rc=fetch(ctid);
    cntyname=getvarc(ctid,varnum(ctid,'countynm'));
    displist=insertc(displist,cntyname||' County Polygon
Selected');
    ctid=close(ctid);
  end;
/* A Title is selected */
  else if spotid in (1,2) then
    displist=insertc(displist,
              'Title selected, select a polygon or a marker');
```

```
/* A blank area on map is selected */
  else displist=insertc(displist,
              'You must select a polygon or a marker');

/* display popup and clean up list */
  rc=popmenu(displist);
  displist=dellist(displist);
return;

term:
  clid=dellist(clid);
return;
```

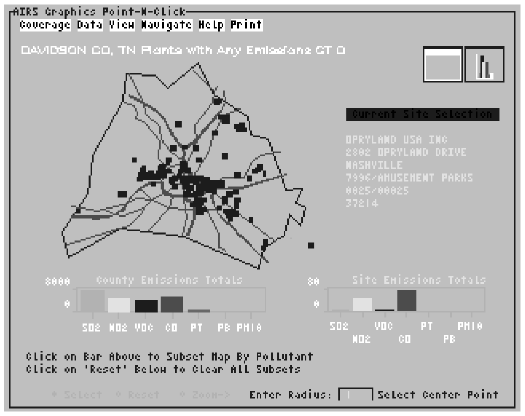
Contents of Plants Data Set:

```
-----Variables Ordered by Position-----

#   Variable   Type  Len   Pos  Format  Label
_________________________________________________________________
1   REGION      Num    5     0    2.    EPA Region
2   STABBR      Char   2     5          State Abbrev.
3   STATE       Num    5     7    3.    FIPS State Code
4   COUNTY      Num    5    12    3.    FIPS County Code
5   PLT_ID      Char   4    17          Plant ID (NEDS)
6   PLT_CDS     Char   5    21          Plant ID (CDS)
7   PLT_NAME    Char  40    26          Plant Name
8   PLT_ADDR    Char  30    66          Plant Street Address
9   PLT_CLS     Char   2    96          Plant Classification
10  PLT_CST     Char   1    98          Plant Compliance Status
11  OPST        Char   1    99          Plant Operating Status
12  SIC         Num    5   100    Z4.   SIC Code
13  SICD        Char  25   105          SIC Description
14  PLT_CITC    Char   5   130          City Code
15  PLT_CITY    Char  30   135          City Name
16  PLT_ZIP     Char   5   165          Plant ZIP Code
17  LON         Num    5   170    7.    Longitude (d-m-s)
18  LAT         Num    5   175    7.    Latitude (d-m-s)
19  X           Num    5   180          Longitude (radians)
20  Y           Num    5   185          Latitude (radians)
21  SIP         Char   1   190          SIP Plant?
22  LARGE       Char   1   191          Large Plant? (Emis.)
23  MERGED      Char   1   192          Merged Plant?
24  XYCOORD     Char   1   193          XY Coords?
25  DUPCOORD    Num    5   194    5.    Dup Coords Flag
26  BADX        Char   1   199          X Coord Bad?
27  BADY        Char   1   200          Y Coord Bad?
28  PLT_ECON    Char  20   201          Emissions Contact
29  DTYPE       Char   4   221          Data Type
30  YEAR        Num    5   225    2.    Emis. Inventory Year
31  MINEMIS     Num    5   230    8.2   Min.Pollutant Emissions
32  MAXEMIS     Num    5   235    8.2   Max. Pollutant Emissions
33  TOTEMIS     Num    5   240    8.2   Total Pollutant Emissions
34  CO          Num    5   245    8.2   CO
35  NO2         Num    5   250    8.2   NO2
36  SO2         Num    5   255    8.2   SO2
37  VOC         Num    5   260    8.2   VOC
38  PT          Num    5   265    8.2   PT
39  PM10        Num    5   270    8.2   PM10
40  PB          Num    5   275    8.3   Lead
```

Duplicate coordinate values have been covered in the example. The only other complication in this overall scheme is a map where markers are so dense that they become stacked. These markers do not have duplicate coordinates, however they are so close together that only the topmost markers are accessible. To deal with this problem, we added a zoom function which allows the user to pick a radius, select a marker for the center point and redraw a circular subset of the data around that point. This has the effect of spreading out the markers and, at some zoom level, will allow the selection of the individual markers even in areas of high density.

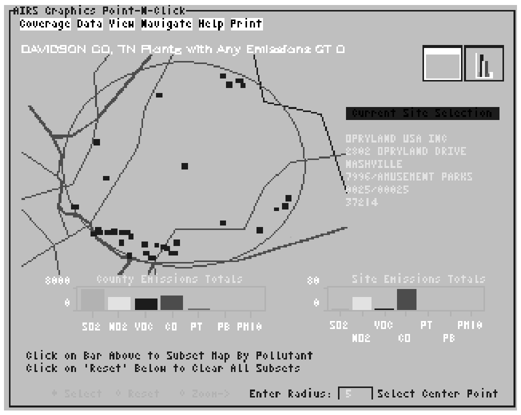
The following screens show a county level map and a 5-mile radius subset around a selected marker:



DAVIDSON CO, TN Plants with Any Emissions GT 0

OPRYLAND USA INC
2802 OPRYLAND DRIVE
NASHVILLE
7996/AMUSEMENT PARKS
0025/00025
37214

Click on Bar Above to Subset Map By Pollutant
Click on 'Reset' Below to Clear All Subsets



DAVIDSON CO, TN Plants with Any Emissions GT 0

OPRYLAND USA INC
2802 OPRYLAND DRIVE
NASHVILLE
7996/AMUSEMENT PARKS
0025/00025
37214

Click on Bar Above to Subset Map By Pollutant
Click on 'Reset' Below to Clear All Subsets