Using SAS/GRAPH® Software to Create Graphs on The Web

Himesh Patel, SAS Institute Inc., Cary, NC

ABSTRACT

Version 7 SAS/GRAPH® software will contain several enhancements that enable users to easily display graphics in Web pages and to create "drill-down" graphs in Web environments. This paper describes these enhancements and presents some examples of their use. Note that these features are still under development; syntax and usage are subject to change prior to Version 7 release.

INTRODUCTION

You can use SAS/GRAPH software in the following ways to produce output on the Web:

1. Using SAS/GRAPH Device Drivers

This method creates GIF files and can also be used to generate a HTML page containing the associated GIF files. This approach is particularly useful in batch applications where you want to generate a large number of graphs that can be viewed latered with a Web browser.

2. Using ODS (Output Delivery System)

This method takes advantage of the Output Delivery System that will be available in Version 7. ODS can create HTML files and can be used to combine output from multiple procedures (graphics and non-graphics) using custom layouts. You can use SAS/GRAPH options with ODS to create graphs with the drill-down feature. This feature is designed for use in interactive environments.

3. Using the WEBOUT= Option to Create SAS/GRAPH Output Data Sets

This method is for users with extensive HTML knowledge who want to write their own Web pages that include SAS/GRAPH output with drill-down capabilities. With this method, SAS/GRAPH creates an output data set containing coordinates that can be used to generate HTML MAP tags. You use a SAS DATA step to process the output file and generate your own HTML files. This gives you complete control over the appearance and structure of your Web pages.

USING SAS/GRAPH DEVICE DRIVERS

SAS/GRAPH provides three different device drivers that can be used to create graphics output for the Web: GIF, HTML and WEBFRAME. The ***GIF*** driver creates a GIF format file that can be referenced in an HTML file. Use of this driver requires that you create your own HTML file containing the references to the GIF files created by the driver. The ***HTML*** driver creates a single HTML file, as well as a GIF file for each graph produced by SAS/GRAPH procedures. The HTML file displays all of the GIF files on a single, scrollable Web page. The ***WEBFRAME*** driver creates a single HTML page containing two frames. Thumbnail versions of all of the graphs are displayed in one frame, and when you click on a thumbnail image, an HTML file containing a full-size image of the graph is displayed in the other frame.

The GIF Driver

This is the simplest method to get the SAS/GRAPH output in your HTML documentation. To create a GIF file, specify the following:

```
filename grafout 'filename.gif';
goptions device=gif gsfname=grafout
  gsfmode=replace;
```

Note that 'grafout' is an arbitrary fileref and can be replaced by any valid SAS fileref. *filename.gif* should be replaced with the name of the GIF file you want to create. Once you create the GIF file, it can be included in the HTML page to be viewed under any Web browser.

The HTML Driver

You can use the HTML device driver to create a series of graphs that can be viewed on a single scrollable page with any viewer. Whereas the GIF driver creates GIF files that must be included in a separately developed Web page, the HTML driver generates both the GIF files and a HTML file that displays them.

To create a series of graphs, specify:

```
filename out 'directory';
goptions device=html gsfname=out
 gsfmode=replace;
```

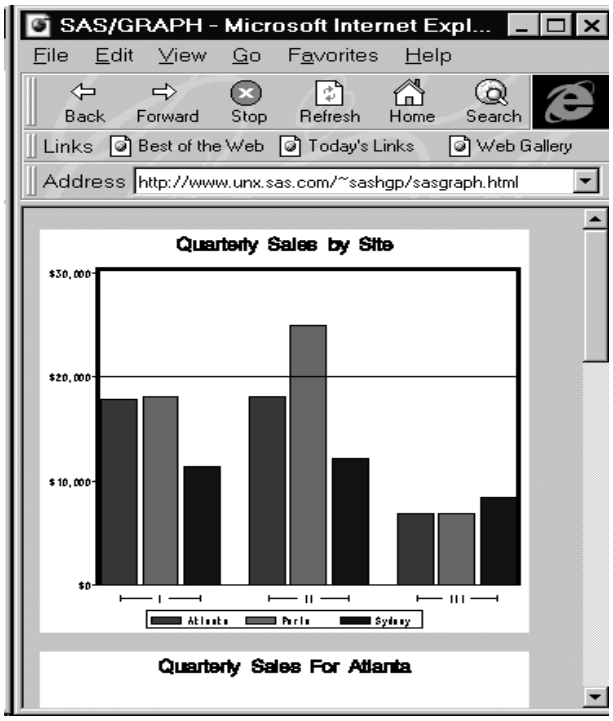
The HTML driver creates a default HTML file called sasgraph.html in the directory pointed to by the FILENAME statement. The driver also creates a GIF file for every graph produced by the procedure.

The following program creates a series of four graphs using the HTML driver. The first two graphs are shown in *Outputs 1* and *2*, on the next page.

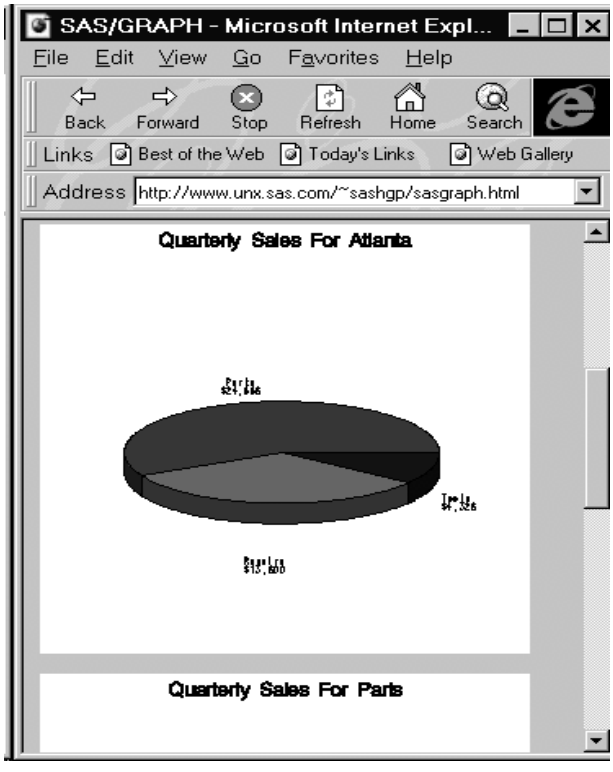
Example 1:

```
data totals;
     length dept $ 7 site $ 8;
     input dept site quarter sales;
     datalines;
Parts   Sydney  1 4043.97
Parts   Atlanta 1 6225.26
Parts   Paris   1 3543.97
Repairs Sydney  1 5592.82
Repairs Atlanta 1 9210.21
Repairs Paris   1 8591.98
Tools   Sydney  1 1775.74
Tools   Atlanta 1 2424.19
Tools   Paris   1 5914.25
Parts   Sydney  2 3723.44
Parts   Atlanta 2 11595.07
Parts   Paris   2 9558.29
Repairs Sydney  2 5505.31
Repairs Atlanta 2 4589.59
```

```
Repairs Paris   2 7538.56
Tools   Sydney  2 2945.17
Tools   Atlanta 2 1903.99
Tools   Paris   2 7868.34
Parts   Sydney  3 8437.96
Parts   Atlanta 3 6847.91
Parts   Paris   3 6789.85
run;

   /* location for output files */
filename out '/tmp';

   /* Specify device related options */
goptions reset=all device=html
        gsfname=out gsfmode=replace;

   /* Define the titles */
title 'Quarterly Sales by Site';

   /* axis characteristics */
axis1 value=none label=none width=2;
axis2 label=none minor=none;

   /* legend characteristics */
legend1 cborder=black label=none;

   /* Create a vertical bar chart */
proc gchart data=totals;
    format quarter roman.;
    format sales dollar8.;
    vbar site / sumvar=sales
                subgroup=site
                group=quarter
                gspace=4
                space=1
                ref=20000
                maxis=axis1
                raxis=axis2
                gaxis=axis2
                caxis=black
                legend=legend1
run;

  /* 3-D pie chart for Atlanta */
title 'Quarterly Sales For Atlanta';
where site="Atlanta";
 pie3d dept / noheading
              sumvar=sales
             name='Atlanta';
run;

   /* 3-D pie chart for Paris */
title 'Quarterly Sales For Paris';
where site="Paris";
  pie3d dept / noheading
               sumvar=sales
               name='Paris';
run;

   /* 3-D pie chart for Sydney */
title 'Quarterly Sales For Sydney';
where site="Sydney";
  pie3d dept / noheading
               sumvar=sales
               name='Sydney';
run;
quit;
```



Output 1: First Graph Produced by HTML Driver



Output 2: Second Graph Produced by HTML Driver

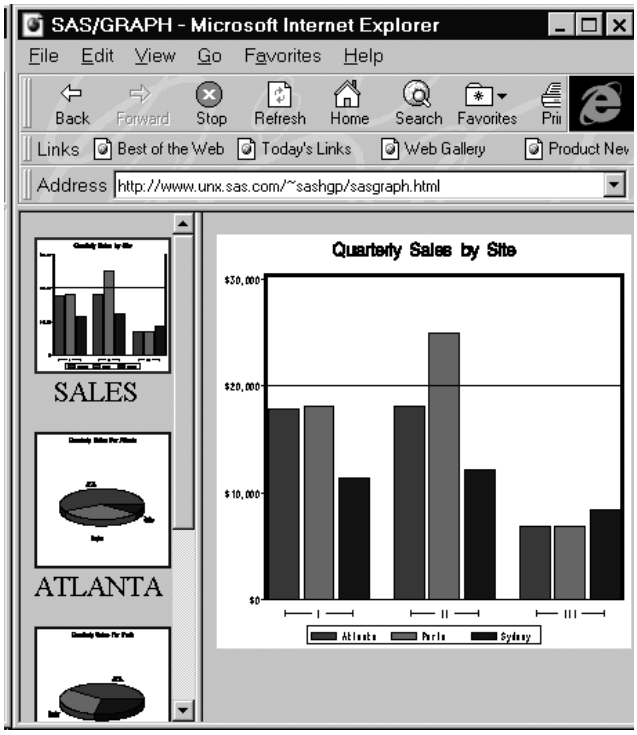
The above program creates a file called sasgraph.html, and five GIF files, one for each graph. The driver formats the HTML file so that all GIF images display in a single, scrollable window.
The GACCESS field in the HTML device entry points to the default location where the GIF files are stored. On most hosts this is your current directory. To redirect the GIF files elsewhere, you can change the GACCESS value by modifying the device entry or by using a GOPTIONS statement.

This HTML driver is particularly useful for batch jobs that create HTML pages on the fly. If you are using the driver interactively, you can modify the driver so that it invokes a Web browser automatically. To do this, you should use PROC GDEVICE to edit the HTML driver entry and specify the path to your browser in the DRVTERM1 parameter:

```
proc gdevice cat=gdevice0.device nofs;
   mod html fileclose=DRIVERTERM
            drvterm1='Browser location';
```

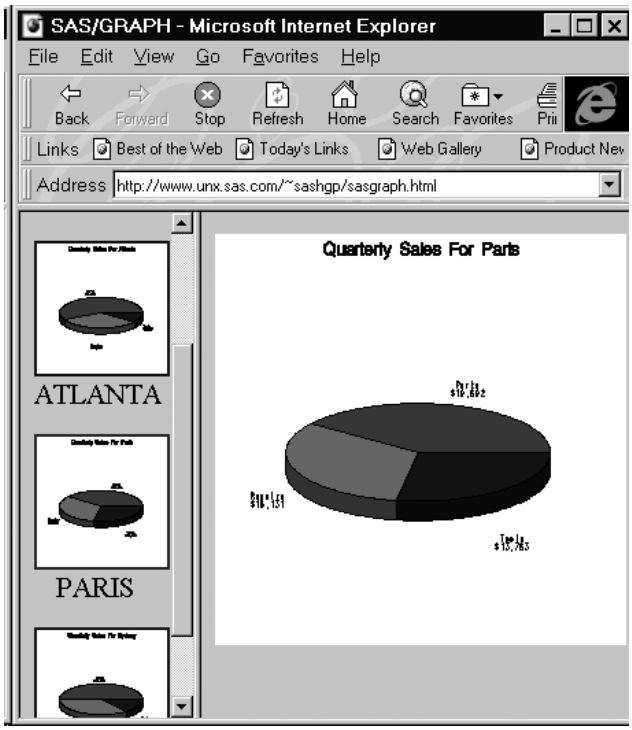
The WEBFRAME Driver

The WEBFRAME device works similarly to the HTML device with the exception of the layout. The WEBFRAME driver creates one or more GIF files and several HTML files that are used to display the GIF images with thumbnail links. By default the layout is broken down in two views. The left side contains the thumbnail size graph and the right side contains a full-size version of the selected thumbnail image. Running the code from Example 1 using the WEBFRAME driver instead of the HTML driver creates *Output* 3 below.



Output 3:
Web Page Produced by WEBFRAME Driver

In the above output, when you click on the thumbnail for Paris, the full-size graph for Paris appears on the right hand side, as shown in *Output 4*. Each full-size graph is displayed as a separate HTML file.



Output 4:
Web Page After Thumbnail Image For 'Paris' Is Clicked

The WEBFRAME driver creates the following files:

- **sasgraph.html*:*** This is the overall Web page which is composed of two frames. The frames contain the thumbnail images created by the driver and the full-size version of the selected thumbnail image. This file is intended to be viewed by a browser or referenced from another HTML page.
- **sasthumb.html*:*** This file displays thumbnails of each full-size GIF image created by the driver in a frame on the left hand side of the browser window. It also links the thumbnails to the full-size GIF images. When the thumbnails are clicked, the full-size image is displayed in the frame on the right side of the window. The name of each image appears just below each thumbnail and corresponds to the name of the GRSEG entry.
- ***GRSEG entry name*.html:** This HTML file displays the full-size image indicated, where *GRSEG entry name* is the base name associated with the full-size image. The HTML file name corresponds to the name of the GRSEG entry and can be controlled by specifying the NAME= option in the procedure.
- ***GRSEG entry name*.gif:** The WEBFRAME driver creates a thumbnail and full-size image pair for each graph produced by the procedure. The full-size image file name corresponds to the name of the GRSEG entry. The thumbnail image file name begins with "t". The remaining characters are the same as those used for the file name for the full-size image. For example, if you generate two graphs with PROC GCHART and do not specify the NAME= option in the procedure, the driver will create GRSEG entries called GCHART and GCHART1, and

will create full-size GIF files called gchart.gif and gchart1.gif, and thumbnail GIF files called tgchart.gif and tgchart1.gif.

By default, the GACCESS field of the WEBFRAME device entry points to the location where the files are created; on most hosts, this will be your current directory. To redirect the files elsewhere, you can change the GACCESS value by modifying the device entry or by using a GOPTIONS statement.
Like the HTML driver, the WEBFRAME driver can be used in batch jobs to create HTML pages on the fly and can also be modified invoke a browser automatically.

OUTPUT DELIVERY SYSTEM (ODS)

ODS allows you to combine output from different procedures and define custom layouts. In addition, the ODS HTML statement allows you to create SAS/GRAPH output with drill-down capability. This method requires some knowledge of HTML syntax and familiarity with the ODS HTML statement.

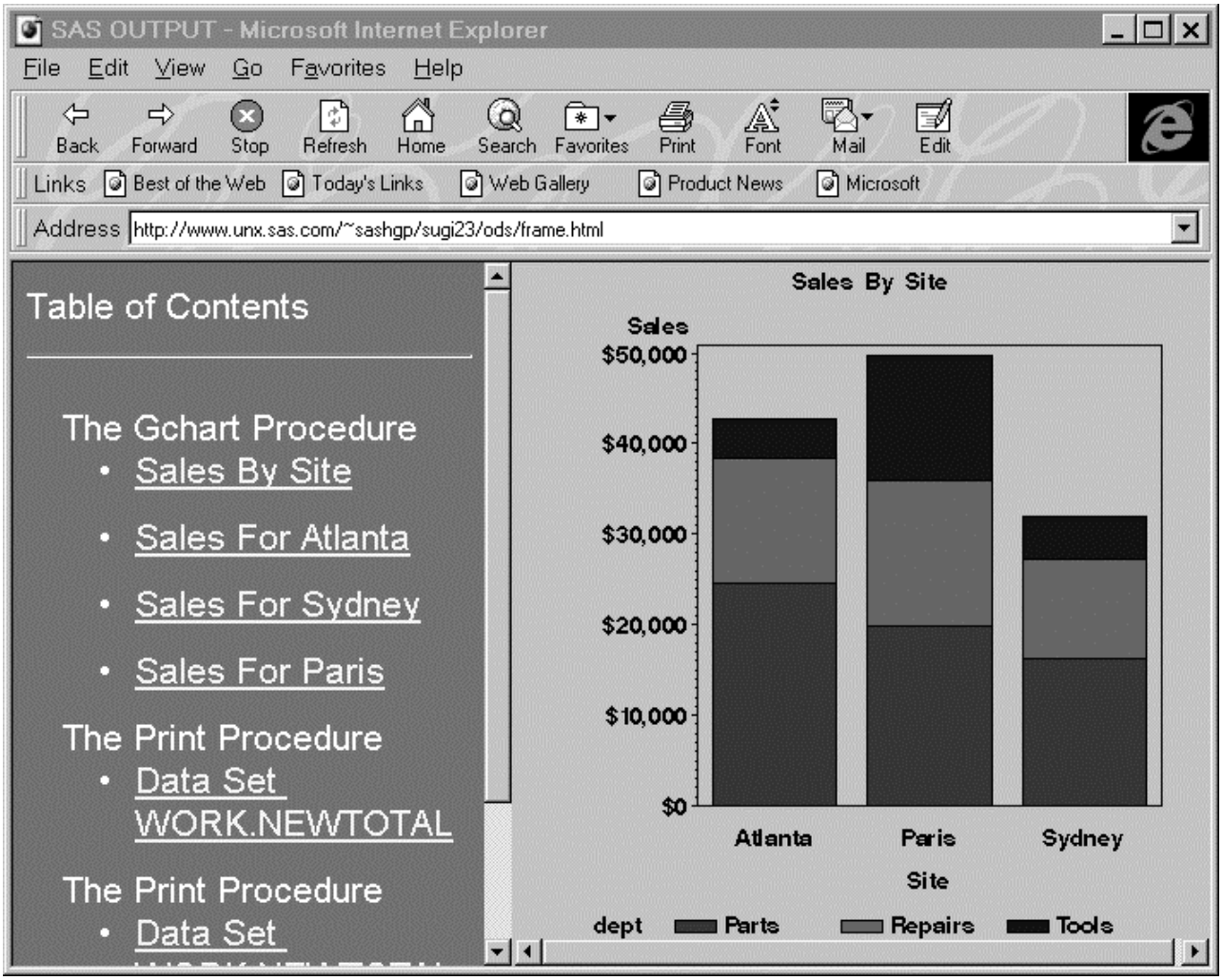
In *Output* 5, below, you can click on the descriptions displayed in the table of contents to see the appropriate graphs or text reports. You can also click on the bars. If you click on the bar for Atlanta, a 3-D pie chart for Atlanta is displayed. Similarly, you can click on the Paris or Sydney bars to display a 3-D pie chart for each of those cities. You can also click on the legend entries for Parts, Repairs or Tools to see a quarterly report produced by PROC PRINT.

Structure of ODS Output

ODS HTML output always creates a "body" file, which is a single HTML document containing the output from one or more procedures. If the SAS program only produces a single piece of output (such as a single graph), the body file is all that is displayed. If more than one piece of output is created, ODS HTML can also create a "frame" file, which is an HTML document containing multiple frames. The frame file can display the body file, a table of contents, and other frames. For example, Output 5 displays a table of contents and a graph (the body file) in two frames on a single Web page.

Creating Graphs With Drill-down Capability

To generate a graph with drill-down capability, you must define the links that connect each drill-down area with the corresponding graph or report. These links are HTML commands that are stored in the HTML body file and point to another location in the body file that is the target of the drill-down. To define these links, you store the HTML commands in a variable in the input data set. When ODS creates the body file, it includes these commands in the file.



Output 5: Output from ODS

The code that follows illustrates how to create a graph with drill-down bars. The first step is to decide what action you want to occur when you click on the bars. Next, assign this action to a new variable that defines the HTML links from those bars. In this example, two new variables (BARDRILLS and LEGENDDRILLS) are added to the data set used in the earlier example. These variables contain the links that will be used when each bar or legend entry is clicked on.

```
data newtotal;
set totals;
length bardrills $ 40 legenddrills $ 40;
if (trim(site)='Atlanta') then
Bardrills='HREF=#Atlanta';
else if (trim(site)='Paris') then
   bardrills='HREF=#Paris';
else if (trim(site)='Sydney') then
   bardrills='HREF=#Sydney';

if (trim(dept)='Parts') then
   legenddrills='HREF=#Parts';
else if (trim(dept)='Repairs') then
   legenddrills='HREF=#Repairs';
else if (trim(dept)='Tools') then
   legenddrills='HREF=#Tools';
run;
```

Note that the values of both the new variables are of the form '*HREF=#<anchor-name>*'. *<Anchor-name>* can be any name that you want to use, but it has to be a unique name for each link. For example, in the above code, for all observations with SITE='Atlanta', the variable BARDRILLS contains the value 'HREF=#Atlanta'. When the bar chart is displayed and you click on any segment of the bar for Atlanta, you link to an anchor labeled 'Atlanta' in the body file. This anchor displays a 3-D pie chart for Atlanta. Later in the program you use the ANCHOR option of the ODS HTML statement to create the 'Atlanta' anchor in the body file with the 3-D pie chart for Atlanta.

Before creating the graphs, you must stop output from going to the GRAPH and OUTPUT windows before it can be directed to an HTML file. You must also specify that SAS/GRAPH use the GIF driver to create graphics output.

```
   /* Start sending output to html */
ods listing close;
goptions reset=all device=gif;
```

Next, use the ODS statement to define the layout of the HTML document and specify the names and locations of the files to be created. ODS has several predefined layouts which can be selected by the PALETTE option. Output 5 illustrates one such layout (D3D), consisting of a main frame which contains two subframes--the table of contents and the output frame (body file). The FILE= option is required, and specifies the HTML body file that contains the graphics and/or text output displayed on the right hand side. The GPATH= option specifies the directory destination for the GIF files generated by the SAS/GRAPH procedure. The CONTENTS= option specifies the name of the HTML file that is displayed in the left sub-frame. The FRAME= option specifies the name of the main HTML frame file that contains the sub-frames.

```
   /* define HTML files and   */
   /* location of GIF files   */
ods html file='chart.html'
         gpath='mydir'
          contents='contents.html'
          frame='frame.html'
          palette=palettes.d3d;
```

By default, graphs produced by the GIF driver have a white background. If the HTML document uses an image map in the background, it can be used as the background for your graphs. To display graphs using a transparent background, specify:

```
goptions transparency;
```

Next, generate the graphs and reports. First create the main vertical bar chart. The GCHART procedure uses the HTML= and HTML_LEGEND= options on the VBAR statement to control the drilldown. HTML= specifies the variable in the data set that defines the drill-down for rectangles and polygons in the graph itself. In the data set NEWTOTAL, the variable BARDRILLS is defined to specify the drill-down action for each site. HTML_LEGEND= specifies the variable that controls the drill-down action for each legend value. In the data set NEWTOTAL, the variable LEGENDDRILLS defines the drill-down action for each department. The DES= option specifies the description of the graph that is displayed in the table of contents in the left frame.

```
   /* Create VBAR chart with drilldown action */
proc gchart;
format sales dollar8.;
vbar site / subgroup=dept
            html=Bardrills
            html_legend=Legenddrills
            des='Sales By Site';
run;
```

The graph that is displayed when the drill-down is selected on the bar depends on the unique anchor name associated with that graph. The anchor name is defined in an ODS statement before each target graph is produced.

In this example, the 3-D pie chart for the selected site is displayed when you drill down on the vertical bar. To do this, create a separate 3-D pie chart for each location, and use the ODS HTML ANCHOR option to add an anchor to the HTML body file to display each pie chart. The code below uses a PROC GCHART with a WHERE clause to generate a 3-D pie chart for Atlanta, and uses the ODS HTML statement to add the anchor 'Atlanta' to the HTML body file. The process is then repeated for Sydney and Paris, respectively.

```
   /* define the Anchor point */
ods html anchor='Atlanta';
where site ? 'Atlanta';
title 'Sales For Atlanta';
pie3d dept / noheading
             sumvar=sales
             des='Sales For Atlanta'
             name='Atlanta';
run;

ods html anchor='Sydney';
where site ? 'Sydney';
title 'Sales For Sydney';
```

```
pie3d dept /  noheading
              sumvar=sales
              des='Sales For Sydney'
              name='Sydney';
run;

ods html anchor='Paris';
where site ? 'Paris';
title 'Sales For Paris';
pie3d dept /  noheading
              sumvar=sales
              des='Sales For Paris'
              name='Paris';
run;
```

The legend drill-down action is based on the anchor value used in the LEGENDDRILLS variable. Before each PROC PRINT step, the ODS HTML ANCHOR option is used to define the unique anchor for each department. The anchor name should match the name used in the LEGENDDRILLS variable for each department. Since the PROC PRINT output is to be ordered by site for each department, PROC SORT is used to sort the data by SITE.

```
proc sort data=newtotal; by site; run;

   /* generate a report for Parts department */
title 'Parts Sold';
ods html anchor='Parts';
proc print data=newtotal
         (where = (dept ? 'Parts'));
var site quarter sales;
run;

   /* generate a report for Repairs department */
title 'Repairs Done';
ods html anchor='Repairs';
proc print data=newtotal
       (where = (dept ? 'Repairs'));
var site quarter sales;
run;

   /* generate a report for Tools department */
title 'Tools Sold';
ods html anchor='Tools';
proc print data=newtotal
         (where = (dept ? 'Tools'));
var site quarter sales;
run;
```

Finally, use the ODS HTML CLOSE statement to close all files opened by the FILE=, FRAME=, and CONTENTS= options and to stop generating HTML output. These files remain open until you close them with the ODS HTML CLOSE statement or you specify a different output file.

```
ods html close;
```

Note that to redirect the output back to its default window, an ODS LISTING statement is required.

A portion of the HTML file created by the above code is shown in Output 7 at the end of the article. The MAP tag defines the drill-down areas on the graph, which are the bar segments and legend boxes. Each AREA definition links to an anchor later in the file. When you click on one of the bars for Atlanta, "HREF=#Atlanta" links to the anchor tag '<A NAME="Atlanta"></A>', which has the 3-D pie chart for Atlanta (atlanta.gif) displayed below it. Similarly, if you click on the legend entry for Parts, "HREF=#Parts" links to the anchor tag '<A NAME="Parts"></A>', which displays a formatted table produced by PROC PRINT.

You can use the ODS HTML statement with most SAS/GRAPH procedure. Drilldown using the HTML= and HTML_LEGEND= options is only supported with the following procedure statements:

PROC GCHART
- VBAR, VBAR3D
- BLOCK
- PIE, PIE3D
- HBAR, HBAR3D
- STAR
- DONUT

PROC GPLOT
- PLOT, PLOT2 (with AREAS option only)

PROC GMAP
- CHORO
- BLOCK
- PRISM

The HTML_LEGEND= option is supported when legends are generated by the above procedures.

USING THE WEBOUT= OPTION TO CREATE SAS/GRAPH OUTPUT DATA SETS

If you have a good working knowledge of HTML, you can use the WEBOUT= option and SAS DATA steps to develop Web pages containing graphs with drill-down capabilities.

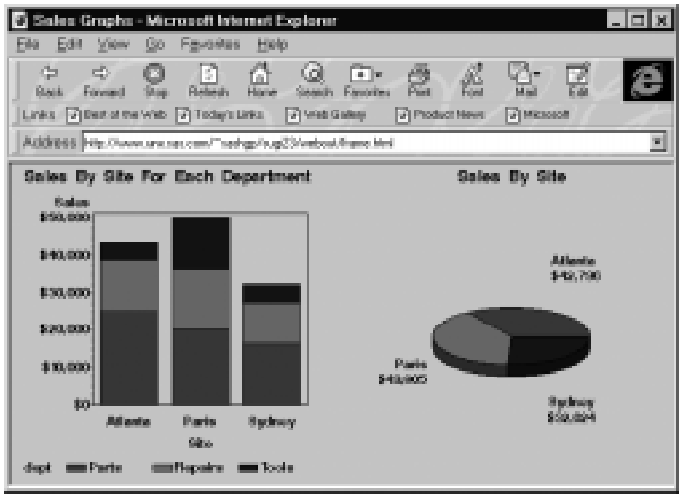
This method combines procedure syntax with SAS DATA steps and macros to generate the HTML output. To generate drill-down output from the GCHART, GPLOT or GMAP procedures, you use the procedure's WEBOUT= option to specify a data set that will store information used to generate HTML pages containing the graphs produced by the procedure. In addition to the WEBOUT= option, you use the HTML= and/or HTML_LEGEND= options to identify the variable whose values create the HTML links. To generate the HTML coding, you use SAS DATA steps to write the HTML tags, and the SAS macro language to dynamically construct the Web pages, based on the output from your SAS program.

This method differs from the ODS method described in the previous section in the following ways:

- With the ODS method, the HTML is actually generated by the ODS system, and all you have to do is supply information about the links. Using the WEBOUT= option to create an output data set requires that you actually write the HTML yourself (although macros are available to facilitate the process).

- With the ODS method, you have limited control over the structure of the HTML that is generated. With the WEBOUT= option, you have complete control over the appearance of the output.

Output 6 shows a Web page that was created using an output data set from SAS/GRAPH created with the WEBOUT= and HTML= options. The page is divided into two frames — the left frame displays a bar chart with drill-down regions, and the right frame displays a graph generated by the drill-down action.



Output 6: Web Page Created Using WEBOUT= Option

The WEBOUT= Data Set

The WEBOUT= option is used on a SAS/GRAPH procedure statement and specifies the name of an output data set in which the procedure stores the outline coordinates for the polygons used in the graph. The output data set has following variables:

GRAPH: a character variable containing the name of the graph. The naming convention is that used for naming GRSEGS, and can be controlled by using the NAME= option in the procedure. If you do not specify the NAME= option, the GRAPH variable will contain the name of the procedure that generated the graph.

LENGTH: a numeric variable containing the length of the LINK variable.

LINK: a character variable containing the value from the input data set variable specified in HTML= or HTML_LEGEND= option. This information is used to specify the action taken when the corresponding polygon or legend entry is clicked.

SHAPE: a character variable containing the name of the shape contained in the graph. SHAPE can have a value of 'RECT' or 'POLY'. For a bar chart SHAPE will have a value of 'RECT' and for a 3-D pie it will have a value of 'POLY'.

NXY: a numeric variable containing the number of points (x/y pairs) in the shape. If SHAPE='RECT', then NXY will have a value of 2, with the lower left points as the first pair and upper right as the second pair. If SHAPE='POLY', NXY contains the actual number of points in the polygon.

X1 *to* X100: numeric variables containing the x coordinates of the shape.

Y1 *to* Y100: numeric variables containing the y coordinates of the shape.

Using the WEBOUT= Option

When using the WEBOUT= option on a PROC statement, you must also use HTML= or HTML_LEGEND= option in the "action" statements (such as PIE, VBAR, HBAR, CHORO, etc.) of the procedure. The HTML= option specifies a variable in the input data set that contains drill-down information for polygons in the graph. The HTML_LEGEND= option specifies a variable containing drill-down information for legend boxes. In the example below, you use the NEWTOTAL data set that was used in the previous example. The data set contains a variable named BARDRILLS, which contains the drill-down specifications for each bar (for example 'HREF=#Atlanta'). In this example, you will generate a bar chart to be displayed in the left frame, and a summary 3-D pie chart in the right frame. When a bar is clicked in the left frame, a drill-down 3-D pie chart for the corresponding location should be displayed in the right frame.

First generate the bar chart that is to be displayed in the left frame. All of the graphs are stored as GIF files in the directory specified in a FILENAME statement.

```
   /* Location to store all the GIF files */
filename image 'sugi23';

goptions reset=all dev=gif
         gsfname=image gsfmode=replace
         ftext=swissb transparency htext=4 pct;

axis1 label=('Site');
axis2 label=('Sales');
legend label=('Dept.');
Title h=5 pct 'Sales By Site For Each
Departments';

libname ex 'sugi23';

   /* Generate picture for left (body) frame */
proc gchart gout=ex.exp2
            webout=htmldat
            data=newtotal;
format sales dollar8.;
vbar site / subgroup=dept
            sumvar=sales
            width=15
            maxis=axis1
            raxis=axis2
            legend=legend1
            html=bardrills
            des='sales by site'
            name='salesbar';
run;
quit;
```

The WEBOUT= option creates a data set called HTMLDAT and stores in it the coordinates and link information associated with the polygons in the chart. The HTML= option specifies the name of the variable (BARDRILLS) that contains the action that will be taken when the corresponding polygon is clicked.

Now generate a summary 3-D pie chart to be displayed initially in the right frame, then a separate pie chart for each location. Note that these charts will not have drill-down ability, so the WEBOUT= and HTML= options are not specified.

```
/* Generate initial chart for right frame */
Title h=5 pct 'Sales By Site';
proc gchart gout=ex.exp2 data=newtotal;
format sales dollar8.;
pie3d site / noheading
```

```
            sumvar=sales
            des='Sales by site'
            name='salespie';
run;

/* Generate picture for each site */
Title h=5 pct 'Sales For Atlanta';
where site ? 'Atlanta';
pie3d dept / noheading
            sumvar=sales
            des='sales for atlanta'
            name='atlanta';
run;

Title h=5 pct 'Sales For Sydney';
where site ? 'Sydney';
pie3d dept / noheading
            sumvar=sales
            des='sales for sydney'
            name='sydney';
run;

Title h=5 pct 'Sales For Paris';
where site ? 'Paris';
pie3d dept / noheading
            sumvar=sales
            des='sales for paris'
            name='paris';
run;
quit;
```

To generate the main frame window containing *salesbar.html* and *salespie.html* you will use a simple SAS DATA step with PUT statements that will write HTML information to an external file. The main frame window is divided into two sub-frame windows. The left hand side frame is referenced as *body*, containing the output from *salesbar.html*, and the right hand side is referenced as *contents*, containing the output from *salespie.html*. Note this type of layout is automatically defined for you if you select the ODS HTML method.

```
   /* create the main frame file */
filename frame '/sugi23/frame.html';
data _null_;
   file frame;
   put '<HTML>';
   put '<HEAD>';
   put '<TITLE>Sales Graphs</TITLE>';
   put '</HEAD>';
   put '<FRAMESET FRAMEBORDER=YES'@;
   put ' FRAMESPACING=0 COLS="52%,*">';
   put '<FRAME MARGINWIDTH="0"'@
   put ' MARGINHEIGHT="0"'@;
   put ' SRC="salesbar.html" NAME="body">';
   put '<FRAME MARGINWIDTH="0"'@;
   put ' MARGINHEIGHT="0"'@;
   put ' SRC="pies.html" NAME="contents">';
   put '</FRAMESET>';
   put '</HTML>';
   put '</FONT>';
   put '</BODY>';
   put '</HTML>';
run;
```

The next step is to read in the output data set that was created by the WEBOUT= option, and generate an HTML file (salesbar.html) with appropriate image maps. Creation of this HTML file involves three steps:

- Generating the necessary HTML tags.
- Generating the appropriate image maps from the WEBOUT data set.
- Closing the HTML tags generated in the first step.

To make the example simple, only the necessary HTML tags are used.

```
   /* Generate HTML file with drill-down info */
data _null_;
   set htmldat end=last;
   file 'salesbar.html';

      /* Initialize the HTML file with         *
       * required tags and information for the *
       * first graph.                          */
   if _n_=1 then do;
      put '<HTML>';
      put '<HEAD>';
      put '<TITLE>Sales Chart</TITLE>';
      put '</HEAD>';
      put '<BODY>';
      put '<IMG SRC="salesbar.gif" '@
      put 'USEMAP="#barchart_map">';
      put '<MAP NAME="barchart_map">';
   end;

   array x{100} x1-x100;
   array y{100} y1-y100;

      /* Add image map points based on shape */
   if shape='RECT' then do;
      put '<AREA SHAPE="RECT" '@;
      put  link $varying. length @;
      put ' TARGET="contents"' @;
      put ' COORDS="' @;
      put 'x1 4. ',' y1 4. ',' x2 4. ',' y2 4. '
          ">';
   end;
   else
      if shape='POLY' then do;
         put '<AREA SHAPE="POLY" ' @;
         put  link $varying. length @;
         put ' TARGET="contents"' @;
         put ' COORDS="' @;
         do i=1 to nxy-1;
            put x{i} 4. ',' y{i} 4. ',' @;
         end;
         put x{nxy} 4. ',' y{nxy} 4. @;
         put ' ">';
      end;

      /* End the HTML file    */
   if last then do;
      put '</MAP>';
      put '</BODY>';
      put '</HTML>';
   end;
run;
```

When you drill down on the bar chart, the right frame is updated with the appropriate 3-D pie chart. In the example there are 4 pies--one for each location, plus the summary pie. The macro LINKFILE is executed once for the summary pie and again for each location, and concatenates an anchor and image link onto the end of the pies.html file.

```
%macro linkfile(htmlfl);
      put '<MAP NAME='"&htmlfl"'>';
      put '<IMG SRC=' "&htmlfl" '.gif>';
%mend;

/* Create the html file for Main Frame layout */

data _null_;
      file  "/sugi23/pies.html" mod;
```

```
put ’<HTML>’;
put ’<HEAD>’;
put '<TITLE>' "Pie Charts” '</TITLE>';
put '</HEAD>';
put '<BODY>';

%linkfile(salespie);
%linkfile(atlanta);
%linkfile(sydney);
%linkfile(paris);

put '</BODY>';
put '</HTML>';
run;
```

SAS Institute will provide macros that will generate a basic HTML file from the data set created by the WEBOUT option. These macros will be located in the ANNOMAC library that is included with SAS/GRAPH software.

The WEBOUT option is currently supported for the following procedures:

- GCHART
- GPLOT
- GMAP

CONCLUSION

SAS/GRAPH software provides three methods to produce Web output, each with its own advantages. If you need a simple Web output format with predefined layout, you will probably find using the GIF and HTML device drivers the most convenient approach. If you want to combine output from graphics and non-graphics procedures, and use drill-down capabilities, the ODS method is most appropriate. If you have extensive HTML knowledge and specific formatting requirements, consider the WEBOUT= option.

```
<HTML>
<BODY leftmargin=8 onload="startup()" bgcolor="#C0C0C0" VLINK="#FFFF77" LINK="#FFFFFF">
<font  face="ARIAL" size="3" color="#000000"><A NAME="Tools"></A><P>
<CENTER>
<MAP  NAME="gchart_map">
<AREA SHAPE="POLY" HREF=#Parts  COORDS="217,592,265,592,265,581,217,581 " >
 <AREA SHAPE="POLY" HREF=#Repairs COORDS="371,592,419,592,419,581,371,581 " >
 <AREA SHAPE="POLY" HREF=#Tools COORDS="525,592,573,592,573,581,525,581 " >
<AREA SHAPE="RECT" HREF=#Atlanta COORDS="332,299,400,508 " >
<AREA SHAPE="RECT" HREF=#Atlanta COORDS="332,160,400,299 " >
<AREA SHAPE="RECT" HREF=#Atlanta COORDS="332,21,400,160 " >
<AREA SHAPE="RECT" HREF=#Paris COORDS="416,299,484,508 " >
<AREA SHAPE="RECT" HREF=#Paris COORDS="416,160,484,299 " >
<AREA SHAPE="RECT" HREF=#Paris COORDS="416,21,484,160 " >
<AREA SHAPE="RECT" HREF=#Sydney COORDS="501,299,568,508 " >
<AREA SHAPE="RECT" HREF=#Sydney COORDS="501,160,568,299 " >
<AREA SHAPE="RECT" HREF=#Sydney COORDS="501,21,568,160 " >
</MAP>
<P>
<IMG SRC="gchart.gif">
<A NAME="Atlanta"></A>
<CENTER><IMG SRC="atlanta.gif">
<A NAME="Sydney"></A>
<CENTER><IMG SRC="sydney.gif">
<A NAME="Paris"></A>
<CENTER><IMG SRC="paris.gif">
<P>
<A NAME="Parts"></A>
<P>
<CENTER><font  face="ARIAL" size="3" color="#000000">
<TABLE  border=0 width=100% cellpadding=2 cellspacing=0 rules=NONE frame=VOID bgcolor="#C0C0C0"><TR>
 <TD ALIGN=CENTER bgcolor="#C0C0C0"><font  face="ARIAL" size="5" color="#000000"><b><i>Parts Sold</i></b></font></TD>
</TR></TABLE><P></font></CENTER>
<CENTER><font  face="ARIAL" size="3" color="#000000">
<TABLE  border=9 cellpadding=5 cellspacing=0 rules=BASIC frame=BOX bgcolor="#D8CCC8" bordercolorlight="#DDDDDD"
bordercolordark="#666666"><thead>
<TR>
 <TD ALIGN=CENTER bgcolor="#707070" bordercolorlight="#EEEEEE" bordercolordark="#555555">
<font  face="ARIAL" size="4" color="#FFFFFF"><b>Obs</b></font></TD>
 <TD ALIGN=CENTER bgcolor="#707070" bordercolorlight="#EEEEEE" bordercolordark="#555555">
<font  face="ARIAL" size="4" color="#FFFFFF"><b>site</b></font></TD>
 <TD ALIGN=CENTER bgcolor="#707070" bordercolorlight="#EEEEEE" bordercolordark="#555555">
<font  face="ARIAL" size="4" color="#FFFFFF"><b>quarter</b></font></TD>
 <TD ALIGN=CENTER bgcolor="#707070" bordercolorlight="#EEEEEE" bordercolordark="#555555">
<font  face="ARIAL" size="4" color="#FFFFFF"><b>sales</b></font></TD>
</TR></thead>
<tbody><TR>
 <TD ALIGN=RIGHT bgcolor="#C0C0C0" bordercolorlight="#EEEEEE" bordercolordark="#555555">
<font  face="ARIAL" size="4" color="#000000"><b> 1</b></font></TD>
.
.
.
.
```

Output 7: Portion of an HTML File Created Using ODS

ACKNOWLEDGMENTS

The following staff members of SAS Institute Inc. contributed to the preparation of this paper:

Mike Kalt	Betsy Corning
Jade Walker	Mary Ellen Toebes
Robert Dolan	Howard Houston
Dave Caira	Faith Durham