ABSTRACT

There are a few guidelines you should keep in mind when developing an application that contains SAS/Graph® Graphics Objects. These include making the graph as large as possible, limiting the size of the data, and working within your hardware limitations. With the Orlando version of SAS® you have the added ability of creating 3D bar charts, 3D pie charts, and grouped bar charts. Other additions allow you to place reference lines on your graphs, specify fonts, and make use of extended select. With the Orlando Addon version more 3D types of graphs are available, as well as overlay bar and line charts and the ability to order bar charts by their response values. New printing capabilities are included in both of these versions. It is also easy to run your application in batch mode. This and other common application needs are addressed in the Questions & Answers section that follows the main discussion.

INTRODUCTION

The Graphics Object is the SAS/Graph extension of the SAS/AF® Application Development system. It lets you create and display a variety of plots and charts including horizontal and vertical bar charts, pie charts, scatter plots, and others. You dynamically create these plots and charts by specifying the plot type, data source, and data variables identified either through the attribute window or with the Graphics Class SCL methods.

The Graphics Object also enables data query. Simply by selecting, or clicking on, a bar, pie slice, or data point of interest you can retrieve the data value that was used for its creation. This value can either be displayed in a data view window or retrieved through an SCL method and used for further data analysis, which could include creating another graph based upon this retrieved value.

ORLANDO ENHANCEMENTS

3D Bar Charts and 3D Pie Charts:

The Graphics Object has added 3D Chart types for Bars and Pies. You can easily select these chart types from the attributes window when creating the Graphics Object, or they can be set using the _SET_TYPE_ method.

/* 3D Pie Chart */
call notify('graph', '_SET_TYPE_', 16);

/* 3D Vertical Bar Chart */
call notify('graph', '_SET_TYPE_', 17);

/* 3D Horizontal Bar Chart */
call notify('graph', '_SET_TYPE_', 18);

Grouped Bar Charts

You can now create grouped Bar Charts with the Graphics Object by specifying a group variable for the graph. To facilitate data analysis, the application
Applications Development

developer can use the following SCL code to let the end user switch the specified midpoint variable with the specified group variable.

```
// Swap midpoint variable and group variable */
length midpoint $24 group $24;
call notify('graph', '_GET_INDEP_VARS', midpoint);
call notify('graph', '_GET_GROUP_VAR', group);
call notify('graph', '_SET_INDEP_VAR', group);
call notify('graph', '_SET_GROUP_VAR', midpoint);
```

Figure 2: Jstyle Pie Chart

Labeling Pie Chart Slices and Jstyle Pie Charts

You can label the pie slices with the midpoint variable that the slice represents, along with the value of the statistic used to determine its size. These labels eliminate the Pie Charts' need for displaying a legend. The default setting is now pie slice labeling.

You can also arrange the slices of a Pie Chart using a Japanese-favored ordering style called JSTYLE. This style can be set from the appearance window, or you can specify it with the _SET__JSTYLE_ method in SCL code. JSTYLE ordering arranges the slices in descending order of the statistic value and draws the slices clockwise starting at the 12 o'clock position (90 degrees).

```
/* Set Japanese Style Ordering for Pie Chart */
call notify('graph', '_SET_JSTYLE', 1);
```

Fonts and Colors for Text Elements

You can assign hardware or software fonts to the text elements of a graph. Accompanying this Graphics Object enhancement is the upgraded version of the portable font selector. For the upgrade, the SAS/Graph software fonts have been added to the portable font selector.

Be aware that the top 11 fonts listed in the portable selector (Helvetica, Times, Palatino, etc.) are only portable if the host system has those fonts available. The only truly portable fonts are the SAS/Graph software fonts listed under these top 11 (starting with Brush). If hardware fonts are desired, Helvetica and Times are generally the safest fonts to use since they are available on most systems.

By default, the font selector used will be the host supplied font selector. It will not contain the SAS/Graph software fonts. To access these truly portable fonts, submit the following statement prior to font selection:

```
OPTIONS MULTENVAPPL;
```

There are two ways to access a font selector: through the object's attribute window or with the fontsel() SCL function. The _SET_FONT_LIST_ method can then be used to associate the text on the graph with a selected font. Also, the colors for the text elements can be set using the enhanced _SET_COLOR_ method.

```
/* Bring up the font selector window and have returned font list applied to the graph */
fontlist = makelist();
fontlist = fontsel(fontlist, ' ');
call notify('graph', '_SET_FONT_LIST', 'CHART', fontlist);
```

```
/* Set the color for the graphs text elements */
call notify('graph', '_SET_COLOR', 'TEXT', 'red');
```

```
/* Override the color of Title 1 */
call notify('graph', '_SET_COLOR', 'TITLE', 'blue', 1);
```

Figure 2: Jstyle Pie Chart
Reference Lines

You can add reference lines to the Graphics Object. Two kinds of reference lines can be specified: those placed at major tick mark locations and those placed at specified values.

Using the _SHOW_MAJOR_REF_LINES_ method, you can request reference lines placed at every major tick mark location or using the _SET_REF_LINES_ method you can individually specify reference line placement.

You can set the color, line style, and line thickness on these reference lines using the _SET_COLOR_, _SET_STYLE_, and _SET_SIZE_ methods. The SCL lists work on a last-in-first-out rule which needs to be remembered when assigning these attributes to the reference lines set in the SCL list.

/* create a list of values */
list=makelist(0);
rc=insertn(list, 5);
rc=insertn(list, 7);
rc=insertn(list, 10);

/* associate list with the vertical axis */
call notify('graph', '_SET_REF_LINES_', 'vaxis', list);

/* set color of the reference line at value 10 */
call notify('graph', '_SET_COLOR_', 'vrefline', 'blue', 1);

Extended Select

The _GET_VALUE_ method now has an optional return parameter. When highlighting is enabled, a number is returned through this parameter. Based upon the value of this number, the application developer can tell if the data point was selected using extended select.

Attributes and Appearance Windows Enhanced

The Attributes Window lets you modify the graph's attributes directly or access an Appearance Window in which you can modify additional attributes. The Appearance Window has been significantly enhanced. From the new Appearance Window you can modify your color selections, choose fonts, add titles and footnotes, as well as specify reference line placement and set pie chart options.

See the QUESTIONS & ANSWERS section to learn how to bring up this window at run time.

PC Printing Capabilities

For PC hosts, the _PRINT_ method now uses the settings defined for your system printer. You can access these settings via the SAS File->Print setup... pull down menu item. The _PRINT_ method will then ignore its device parameter and use the printer specified in the 'Print setup' dialog.

ORLANDO ADDON ENHANCEMENTS

Prism, Cylinder, and Hexagon Chart Types

Three new chart types have been added to the Graphics Object: Prism, Cylinder, and Hexagon. These new chart types can be used to take a traditional Bar Chart and convert it to a presentational style graph.

Order Bar Charts by Response Values

You can now specify the bars on your Bar Chart to be ordered by their response values. Either
Applications Development

ASCENDING or DESCENDING ordering can be requested. With DESCENDING order the bars will be displayed with the tallest bar first and each subsequent bar of equal height or shorter. For ASCENDING order the inverse is displayed. These orderings make reading the graph easier and more natural.

To enable this new functionality, use the _SET_SORT_TYPE_ method with an added optional parameter set to RESPONSE.

call notify( 'graph', '_SET_SORT_TYPE_','DESCENDING', 'RESPONSE');

Overlay Bar and Line Charts

A new optional numerical argument for the _SET_GRAPH_TYPE_ and _SET_DEP_VAR_ methods has been added. With this argument you can now specify bar charts with line chart overlays.

You can specify multiple dependent variables to be overlaid on one independent variable. These can be any combination of bar charts overlaid with line charts.

UNIX printing capabilities

The printing capabilities added for the PC hosts in the ORLANDO release are now available for UNIX hosts. You can access this new feature by first going into the printer setup window and defining a specific printer.

The printer setup can be accessed through the SAS File->Print setup... pull down menu item. To disable this setup and return to the default settings, bring up the View->Preferences window and click to raise the 'Use host printing' square.

Beware, if you have not set up a printer and you have not disabled this new functionality your output will be sent to a postscript file in your current directory.

Specifying Column Widths & Chart Scrolling

You can now control the width of the bars on bar charts. This is needed for many reasons; two specific ones are as follows. One, to achieve a consistent look when using more than one graph in your application. This is especially useful when implementing drill down. Two, to allow the application developer to line a table up underneath the graph.

The _SET_COLUMN_WIDTH_ method has been added for setting column width, which also controls the width of the bars.

The _SET_DEPAxis_WIDTH_ method has also been added for setting the dependent axis width. This
allows you to line up a table under the graph.

Without these settings, the width of the bars is determined by equally dividing up the available graphing area. Allowing the setting of these widths leads us into chart scrolling. When there is not enough physical space to display all the bars at the column width requested, scrolling is implemented.

**GENERAL GUIDELINES**

1. **Make the Graphics Object region as large as possible.** This will cut down on many fitting issues such as overlapping text, clipping of legends, and unevenly spaced or missing axis values. It will also allow pie charts to be larger and more easily labeled.

2. **Use software fonts or a small point size on hardware fonts when your graph contains a lot of text.** With Orlando version, the ability to set hardware and SAS/Graph software fonts has been added. See the 'Fonts and Colors for Text Elements' section for details. If you will be printing your graph or porting your application, use one of the software fonts such as SWISS, SIMPLEX, ZAPF, or CENTB.

3. **Limit the amount of data that will be graphed.** The Graphics Object was not designed to substitute for SAS/Graph PROC output. If your application needs to accommodate very large sets of data, you should consider using the SAS/Graph Output Object. However, you will lose interactive access to your data.

4. **Use larger fonts for main titles than for subtitles and footnotes.** More vertical space is reserved for the main titles. The difference in vertical spacing will become obvious only if you specify the same font size for all four titles.

You may notice the legends overwriting footnotes or being clipped. This is because large legends are not allowed to consume all of the graphing space. In some cases you can circumvent this by using small font sizes or larger graphing area.

5. **Place a Graphics Text Object on top of the Graphics Object.** This allows you to position text anywhere on the graph that you would like. Be aware that if your application requires printing, the Graphics Text Objects will not print as part of your graph.

6. **Develop your application with hardware limitations in mind.** The Graphics Object was designed to take full advantage of sophisticated, high-resolution terminals. You can sidestep many resolution and color mapping challenges by keeping in mind the sophistication of the systems your application will run on.

Consider using the following graph frame setting to accommodate a wide range of hardware:

```
    call notify('graph',"_SET_SIZE="FRAME",0,"thin");
```

Hardware considerations should also be made for printed output. If your output device is a pen plotter with limited color capabilities and your application contains Bar Charts or Pie Charts, use the 2D versions instead of 3D. The combined result of color mapping with limited pen colors can cause a "patchwork quilt" effect.

**QUESTIONS & ANSWERS**

**Question 1:** How can I allow the users of my application to bring up the attributes window and customize the graph?

**Answer:** Simply execute the following method when
your application is running.

call notify('graph', '_ATTRIBUTES_DIALOG_');

Question 2: What can I do if I find there are many bars without axis values underneath?

Answer: Some axis values will be disabled when there is not enough physical space for them to fit without overlapping. To work around this use a smaller font or larger graph region. Also, with chart scrolling charts being implemented in the Orlando Addon version this should not occur as often.

Question 3: What can I do if the legend on my Pie or Bar Chart is truncated?

Answer: Enlarge the Graph Object's region to accommodate the complete legend. Specifying a smaller font can also help. See 'General Guidelines' section under item number 2 for details.

Question 4: Can I manipulate my Graphics Object in BATCH mode running without user interaction?

Answer: Yes, it is easy to manipulate your graphs without user interaction. The following example shows how to print two graphs changing only the dependent variable.

Add to your INIT section the method calls to set the graph attributes you desire and then call the _PRINT_ method for each graph that you wish to print.

INIT:
call notify('graph', '_SET_DSNNAME', 'data.one');
call notify('graph', '_SET_INDEP_VAR', 'date');
call notify('graph', '_SET_DEP_VAR', 'jan');
call notify('graph', '_PRINT', 'driveiname');
call notify('graph', '_SET_DEP_VAR', 'feb');
call notify('graph', '_PRINT', 'driveiname');
return;

Question 5: What can I do about my pie slice labels overwriting or writing under the slices?

Answer: The pie labelling method is extremely flexible. If you suspect that your Pie Charts will be consistently problematic with regard to labels (for example, long character names or large statistic values) a global solution would be to use a legend instead of pie labels. You may also find better results by using a smaller font for your labels.

The _SHOW_LABELS_ method can also be used to accommodate label placement.

call notify('graph', '_SHOW_LABELS_','BEST FIT');

The following values can be used as parameters to this method. Be aware that each of these values except for "BEST FIT" will resize the pie in an effort to accommodate the labels.

"BEST FIT" - The best fit that can be made without resizing the pie. Each slice's text is evaluated individually and placed where it will fit. (This is the default.)

"INSIDE OUTSIDE" Midpoint value outside the slice and statistic inside the slice. This parameter decreases the size of the pie to accommodate the outside text.

"INSIDE" Both midpoint and statistic inside the pie. This parameter increases the size of the pie since no text space is reserved for outside labels.

"OUTSIDE" Both midpoint and statistic outside the pie. This parameter decreases the size of the pie.

Truncation of labels may still occur but can be reduced by following the guidelines below:

a) Short midpoint / short statistic - any of the methods will work. If you have a limited graphics object region size, "INSIDE" would be the best one to use.

b) Long midpoint / short statistic - use "INSIDE OUTSIDE"

c) Long midpoint / long statistic - use "OUTSIDE", especially if region size is limited.
d) Short midpoint / long statistic - use "OUTSIDE"

Question 6: My titles and/or footnotes are writing on top of each other or on top of the legend, what can I do?

Answer: This problem has been eliminated with the Orlando Addon version. The titles and footnotes will now scale down to fit within the region. If you are working with an earlier release there are several things you can do to solve this problem. The global solution is to enlarge the Graphics Object region vertically. Specifying a smaller font can also help. To avoid a footnote writing over a legend, use the _SET_JUSTIFY_ method to move the footnote to the left or right.

call notify('graph', '_SET_JUSTIFY', 'left', 'footnote', 1);

Question 7: My default symbol / line / axis / text color isn't the color i want. Can I change it?

Answer: Use the _SET_COLOR_ method to set the colors that you want.

call notify('graph', '_SET_COLOR_', 'FRAME', 'RED');

Question 8: Unexpected font substitutions sometimes occur when I port my application from one platform to another. Can I prevent this?

Answer: Yes, specify software fonts for your text. This substitution only occurs when hardware fonts are specified. If the application is ported to a host that does not have the same hardware font as specified, a substitution will be used. See 'General Guidelines' section under item number 2 for details.

Question 9: How do I explode the OTHER pie slice?

Answer: The _GET_VALUE_ method returns the values that compose the OTHER slice. You could use these values to create a Bar Chart facilitating a graphical view of the OTHER slice as in the example below.

length lab midpoint value $40;
array others[50] $60 ('Midpoint Percentage', ' ');
array mid[3] $20;
array resp[3];

init:
call notify('Bar', '_HIDE_');
call notify('Arrow', '_HIDE_');
return;

pie:
call notify('pie', '_GET_VALUE_', 'listid');
if listen(listid) > 0 then do;
    idtype = getnitemc(listid, 'INDTYPE');
    if idtype='C' then
        idval = getnitemc(listid, 'INDVALUE');
    else
        idnum = getnitemn(listid, 'INDVALUE');
*/---Build Array with Other Slice Values ----*/
if idval='OTHER' or idnum=-99 then do;
    otherlist = getnitem(lislistid, 'DEPVALUE');
    othemum = getnitemn(otherlist, 'NUM');
    do i=1 to othemum;

freq = getnitemn(otherlist, 'FREQ');
freq = putn(freq, '6.6');
pct = getnitemn(otherlist, 'PCT');
pct = putn(pct, '4.4');
if idval='OTHER' then do;
   midc = getnitemc(otherlist, 'MIDPNT');
   string=midc;
   end;
else do;
   midn = getnitemn(otherlist, 'MIDPNT');
   string=midn;
   end;

/*/ --- Create Array for Other Bar Chart ---*/
   mid[i] = string;
   resp[i] = freq;
   end; /* DO */

/*/ --- Create the Other Bar Chart ---*/
call notify('Bar', '_SET_INDEP_VAR_', 'mid');
call notify('Bar', '_SET_DEP_VAR_', 'resp');
call notify('Bar', '_UNHIDE_');
call notify('Arrow', '_UNHIDE_');
call notify('Bar', '_UPDATE_');
end; /* other */

/*/ --- Build Array with non-other info ---*/
extlist = getniteml(listid, 'DEPVAlUE');
freq = getnitemn(otherlist, 'FREQ');
freq = putn(freq, '6.6');
pct = getnitemn(otherlist, 'PCT');
pct = putn(pct, '4.4');
if idtype='C' then do;
   midc = idval;
   string=midc;
   end;
else do;
   midn = idnum;
   string=midn;
   end;

/*/ --- Create String and Line up the Columns---*/
   string=trim(string);
   len=11-length(string);
   do k=1 to len;
      string=string||' '; 
   end;
   string=string||freq;
   len=23-length(string);
   do k=1 to len;
      string=string||' '; 
   end;

return;
Bar:
call notify('Bar', '_HIDE_');
call notify('Arrow', '_HIDE_');
return;

Question 10: How can I use the VARSTAT() to place reference lines on my graph?

Answer: You can store the statistic returned from the SCL varstat() function in an SCL list. This list can be passed as a parameter to the _SET_REF_LINES_ method as such:

/*/ table is a list box object that contains a list of possible statistics to calculate */
call notify('table', '_GET_LAST_SEL_', value, sel, text);

/*/ retrieve the dependent variable from the graph object and request the statistic for it */
call notify('graph', '_GET_DEP_VAR_', response);

/*/ store statistic in a list */
statcode=varstat(dsid, response, text, stat);
list=makelist(3);
rc=insertn(list, stat);

/*/ request reference line placement on graph */
call notify('graph', '_SET_REF_LINES_', 'VAXIS', list );

end; string=string||pct;
others[3]=string;

/*/ --- Fill Popup List ---*/
call notify('Bar', '_HIDE_');
call notify('Arrow', '_HIDE_');
poplist=makelist(3);
do j=1 to 3;
   rc=setitemc(poplist, others[j], j);
   end;
   sel=popmenu(poplist);
   end; /* else other */
end; /* listlen > 0 */
Question 11: I want to use Extend Select to select and exclude data values from my graph. Do you have SCL code to do this?

Answer: Use Extended Select to collect a series of selected values and store these in an SCL list. With the values in this list you can submit a WHERE clause on the data set to exclude these values. The example below shows how to do this.

```sas
length dataset $40 xvalue $20 yvalue $20;
init:
  points = makelist();
  dsid=open('data.sel');
  return;
exit:
  rc=close(dsid);
  return;

graph:
call notify('graph', '_GET_VALUE_', listid, status);
if (status = 0) then do;
  x = getnitemn(listid,'indvalue');
  dep= getnitemn(listid,'depvalue');
  y = getnitemn(dep,'value');
  do j = 1 to listlen(points)/2;
      x1= getnitemn(points, 'X', j);
      y1= getnitemn(points, 'Y', j);
      if x = x1 and y = y1 then do;
          rc = delnitem(points, 'X', j);
          rc = delnitem(points, 'Y', j);
      end;
  end;
else if (status = 1) then do;
  end:
else if (status = 2) then do;
  rc = dellist(points);
  points = makelist();
  link add;
  end;
else if (status = 3) then do;
  rc = dellist(points);
  end;
return;

add:
  len = listlen(points)/2;
  len=len+1;
```

CONTACT

The author will be glad to answer questions and accept suggestions at the following addresses:

Lina P. Land
SAS Institute Inc., R5336
SAS Campus Drive
Cary, NC 27513

Phone: 919-677-8000
Email: saspw@unx.sas.com

SAS, SAS/Graph, and SAS/AF are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.