KEYWORDS

FRAME, Graphics Text Object, SAS/GRAPH Output Object, GREPLAY, template

ABSTRACT

Presentations often require graphics which are formed from more than one panel and at times also include an overall or banner title. Typically GREPLAY is used in conjunction with the Template Facility in SAS/GRAPH® to display combinations of previously generated graphs and titles. This approach works well if the developer has a clear and unchanging definition of the layout from which to build the display template. Changing the layout of the display by altering the positions, size, or number of the panels requires a new template definition and has been known to cause programmer consternation. Experimenting with the layout is usually not an option.

A great deal of flexibility can be gained by using FRAME entries to design the display layout. Objects in FRAME can be created, added, dragged, dropped, and resized easily. This presentation will demonstrate how the SAS/GRAPH® Output Objects can be created and then tied to existing graphical entries by using Screen Control Language, SCL. Discussion will include the generalization of the technique using macros.

INTRODUCTION

Applications that combine two or more graphics onto a single display will usually use the template facility within PROC GREPLAY to allocate the size, shape, and relationships of the panels. Very often one of the panels will contain textual information which serves as an overall title for the combined graphs. Detailed information on the design and use of templates for multiple panel displays including a title panel is presented in Carpenter, 1995, p. 174.

This paper will present an alternate method for working with multiple panel graphical displays by using FRAME entries. The techniques presented here will not be used to replace templates or the use of PROC GREPLAY altogether, however they can be used to provide an interactive display design tool or as a display presentation tool for interactive presentations.

USING GREPLAY AND TEMPLATES

Templates are defined by specifying the coordinates of each of the four corners of each of the panels in the display. If the display consists of four panels with a fifth title panel twenty distinct panel coordinates must be specified to define the template. Changing the size of one of the panels may require changing most if not all of these coordinates. Figure 1 shows a typical display of four graphs with a banner title. Each of the graphs was created using PROC GPLOT and the banner title was created using PROC GSLIDE.
It is fairly easy to set up the template for this type of display since there are four equal size panels and the panel used for the banner title is full size. The template coordinates for the five panels are shown in Figure 2. Notice that the template for the banner title is full size and overlays the other four panels.

The template panels are generally defined in the TEMPLATE DESIGN window of PROC GREPLAY, and although this can be an interactive process the user does need to understand the use of the window and how the coordinates interact. The process of relocating or resizing the panels does not lend itself to an EIS style application. What is needed is the ability to display a standard set of panels such as those shown above which can be adjusted using drag and drop techniques. The SAS/GRAPH Output Object used in FRAME Entries gives the user this capability.

FRAME ENTRIES AND GRAPHICS OBJECTS

Within the display area of a FRAME panel regions are defined and then filled with the attributes of one of several types of objects. The two objects that are of particular interest to the display of graphs with banner titles are the Graphics Text Object and the SAS/GRAPH Output Object. The Graphics Text Object is used to display text that can be manipulated and redisplayed using a variety of sizes, colors, and fonts, while the SAS/GRAPH Output Object redisplay previously created graphs and charts.

The Graphics Text Object displays a text string, and its attributes can be controlled either directly when the object is first filled or through a number of methods in the Graphic Text Class. While it is possible to specify the text string through a SCL variable, the object will not import a display of text generated by a SAS/GRAPH procedure such as GSLIDE. This can be limiting if the banner title for the display is generated by SAS/GRAPH and stored in a GRSEG entry or if the graphs were not generated as part of the current job stream.

When the banner title can be generated as part of the current job stream, you can often save the appropriate text in a macro variable which can then be used when you populate the Graphics Text Object.

The Graphics Text Object has an important advantage over the SAS/GRAPH Output Object. The attributes (fonts, color, and size) of a banner title can easily be manipulated in a Graphics Text Object, but an execution of SAS/GRAPH is needed to change the attributes e.g. font, color, or size of a title loaded into a SAS/GRAPH Output Object.
The SAS/GRAPH Output Object is used to display GRSEG catalog entries. GRSEG entries are generally created using SAS/GRAPH procedures, and are the same GRSEG entries that can be replayed using GREPLAY. The object can be populated by specifying the GRSEG entry in the SAS/GRAPH Output Attributes Window or through the use of the _SET_GRAPH_method. The latter provides you with a lot more flexibility since it can be included as part of the SCL program.

The advantage of a SAS/GRAPH Output Object over GREPLAY template panels is that SAS/GRAPH Output Objects can be easily resized and moved by using drag and drop techniques. The disadvantage is that combinations of SAS/GRAPH Output Objects such as are shown below can not be saved as a single entity or plotted on a hard-copy device.

**USING THE _SET_GRAPH_METHOD**

The graph (GRSEG entry) to be displayed in the SAS/GRAPH Output Object is specified by using the CALL NOTIFY routine with the _SET_GRAPH_method in the SCL program associated with the FRAME entry. A typical call might look something like:

```
CALL NOTIFY(graph1, _SET_GRAPH_,'sasuser.mygraph.scatter.GRSEG');
```

In this example the GRSEG entry SCATTER from the catalog MYGRAPH in the libref SASUSER will be displayed in the SAS/GRAPH Output Object labeled GRPH1. Of course usually the name of the entry is not known by the developer so it will be stored in a SCL variable, such as GRENTRY, during execution. The call becomes:

```
CALL NOTIFY(graph1, _SET_GRAPH_, gentry);
```

Notice that the SCL variable containing the name of the entry is not surrounded by quotes.

**DEFINING AND FILLING REGIONS**

The following figure shows a FRAME entry from the developer's point of view. The user is prompted for the names of the two graphs to be displayed. These graphs are automatically displayed in the small preview area in the lower left corner. The user can then display the graphs using the “Display Graphics” push button.

![Figure 3](image_url)

The display of graphs is very similar to what one would see if GREPLAY had been used with a template containing two side-by-side panels. Figure 4 contains an example of two graphs displayed by the user.

![Figure 4](image_url)

The two graphs are previewed in the lower left corner. Pressing the “Display Graphics” push button causes the selected graphs to be
The push button object is named GRAPHNOW and the following code is executed when this button is pushed.

```sas
*** Display the graphics ****
graphnow:
call display('show2.frame' , gname1, gname2);
return;
```

The names of the two selected graphs (GNAME1 and GNAME2) are passed to the entry used to display the graphs (SHOW2.FRAME). Here the _SET_GRAPH_ method is used once again to load the appropriate graph into each of the two SAS/GRAPH Output Objects.

**Figure 5**

Prompting the user for the name of the GRSEG entry or any catalog entry for that matter is a common problem. The following discussion relates more to the SCL coding used to prompt the user than it does to the topic of filling the SAS/GRAPH Output Objects. If you are familiar with the steps necessary to prompt the user for this information feel free to skip to the next section of this paper. Since I won't know my feelings will not be hurt.

There are any number of ways to prompt the user for a catalog entry. The approach used in these programs includes prompts for the library and catalog as well as the GRSEG entry. A quick look at Figure 4 shows that the prompt for the library or LIBREF consists of two objects. A Text Entry Object which is used if the user knows the name of the LIBREF and a Control Object (arrow) which will display a list of available LIBREFs when selected.

The Text Entry Object for the LIBREF is named LIBNAME1 and the SCL contains a section of code that is executed when the contents of this object is modified. The key is the call to the LIBLIST function that creates a selection list of available LIBREFs if the user enters a blank or ?.

```sas
libnamel:
  * Determine the libref;
  libnamel = upcase(libnamel);
  if libnamel in ( blank , '?')
  | libref(libnamel) ne 0 then
    libnamel = liblist( '*' , '*' ,
                      , 'Select one LIBREF' ,
                      , 'y' , 1);
  return;
```

The control object which is named SELLIB automatically calls the LIBLIST function. Both LIBLIST functions assign the selected value to the SCL variable LIBNAME1 which is automatically displayed in the LIBNAME1 Text Entry Object. The following section of SCL is executed when the arrow control object is selected.

```sas
sellib:
  * select a libref from those already established;
  libnamel = liblist( '*' , '*' ,
                    , 'Select one LIBREF' ,
                    , 'y' , 1);
  return;
```

Similar sections of code are executed to determine the catalog and GRSEG entry of interest. The selection list for the catalog is created using the DIRLIST function which uses the selected LIBREF as an argument and then the LIBREF and CATALOG names are used with the CATLIST function to create a list of GRSEG entries. The code used for these calls is shown below.
Advanced Tutorials

catalogi:
* Determine the name of the catalog of
  * interest;
catalogi = upcase(catalogi);
cnamel = libnamel || '.' ||
catalogi;
if catalogi in (_blank_, '?')
  | cexist(cnamel) eq 0 then
    link selcat;
return;

selcat:
  catalogi = dirlist(libnamel
    , 'catalog'
    , 1,'n',''
    ,'Select a Catalog');
cnamel = libnamel||'.'||catalogi;
return;

graphl:
* Select the graphics entry in this
  * catalog;
* gnamel contains the full name of
  * the graph;
gnamel = gnamel||'.'||graphl;
if graphl in (_blank_, '?')
  | cexist(gnamel) eq 0 then
    graphl = catlist(cnamel,'grseg'
      ,1,'n'
      ,'Select an Entry');
gnamel = gnamel||'.'||graphl;
call notify('graphl','_clear_');
call notify('graphl','_set_graph_'
    , gnamel);
return;

selgra:
  graphl = catlist(cnamel,'grseg'
      ,1,'n'
      ,'Select an Entry');
gnamel = gnamel||'.'||graphl;
call notify('graphl','_clear_');
call notify('graphl','_set_graph_'
    , gnamel);
return;

USING GRAPHICS TEXT OBJECTS AS
BANNER TITLES

The following figure shows an screen that allows
the user to optionally provide text which is
placed in a Graphics Text Object for display as a
banner title. In the example below the banner
title “Air Quality Data” comes in through a SCL
variable. In an actual application the text would
probably be determined dynamically through an
earlier process. A Text Entry Object has been
included so that the user can change the text of
the banner title.

MOVING AND RESIZING GRAPHICS
OBJECTS

The real advantage of graphical FRAME objects
is the ease with which they can be moved and
resized. This can be of particular use when the
developer would like to allow the user the ability
to control the layout of the presentation. When
using PROC GREPLAY templates are not easily
reworked by end users unfamiliar with the
Template Design Windows, therefore user
controlled layout is not viable option.

In Figures 4 and 6 above, the developer has
provided a push button (‘Resize Graphics”)
which opens the appropriate FRAME Entry in
Build mode where the various Graphical Objects
can be moved and resized. The push button
object is named RESIZE and the SCL associated
with the panel has a labeled section which is
executed when the button is pressed.

resize:
call build(
    'frame.frameit.show2_gt.frame');
return;

The following figure displays
SHOW2_GT.FRAME as it appears prior to
resizing. The narrow upper object is a Graphical
Text Object and the other two are SAS/GRAPH
Output Objects.
Since the SCL needed to load the graphs has already be supplied by the developer and has already been associated with the appropriate graphics object, the user can move or resize the individual objects as they would do for any other FRAME entry. The objects are filled using the _SET_GRAPH_ method described above and the name of the GRSEG entry selected in previous screen (Figure 6).

**USING GRSEG ENTRIES AS BANNER TITLES**

In Figures 6 and 7 the banner title is displayed through the use of a Graphics Text Object. It is more common to develop banner titles using SAS/GRAPH and then store them as GRSEG entries in a catalog. When using PROC GREPLAY to redisplay this title along with other GRSEG entries (Figure 1), a template such as the one shown in Figure 2 might be used.

SAS/GRAPH Output Objects can be used to display a banner title along with one or more graphs.

Generally the banner title will be created as a one line title at the top of an otherwise blank panel (Figure 8). By placing the title in the appropriate location on an otherwise blank panel problems associated with aspect ratios can be avoided.

Since the panel used to define the banner title is full sized with no changes to the aspect ratio, the SAS/GRAPH Output Object used to display the title will also need to be full sized. If the object is narrow as was the Graphics Text Object in Figures 6 and 7 the text in the title will appear squashed (the aspect ratio would be different). The following figure shows the three objects used to display a banner title along with two other SAS/GRAPH Output Objects. The border of the object used to display the banner title has been highlighted.

The object used for the title (highlighted border in Figure 9) overlaps the other two objects. When created the object used for the title must be laid down first or the 'Make transparent' region attribute must be used. This prevents this object from obscuring the other two objects.
The arrangement of objects in Figure 10 mimics the template shown in Figure 2. Notice that the object used for the banner title is not quite as wide as the other panels. Although this changes the aspect ratio to a small degree, each of the five panels has some area that is uniquely that panel's. This unique area allows the user to easily select a particular object for moving or resizing.

**Figure 10**

**SUMMARY**

The use of FRAME entry graphics objects can facilitate the design and layout process of displays of graphics that require multiple panels. Traditional techniques include the use of PROC GREPLAY and templates to create these types of layouts. Unlike FRAME objects, however, template panels are not easily resized and moved. The use of the FRAME Graphics Text Objects and SAS/GRAPH Output Objects can be especially useful in EIS type applications where the user is less sophisticated or is unfamiliar with the Template Facility in PROC GREPLAY.

**TRADEMARK INFORMATION**

SAS/GRAPH is a registered trademark of SAS Institute, Inc. in the USA and other countries.

* indicates USA registration.

**REFERENCES**


**ABOUT THE AUTHOR**

Arthur L. Carpenter has over eighteen years of experience as a statistician and data analyst and has served as a senior consultant with California Occidental Consultants, CALOXY, since 1983. His publications list includes a book on SAS/GRAPH, a number of papers and posters presented at SUGI, and he has developed and presented several courses and seminars on statistics and SAS programming. Art has served as a steering committee member and president of the Southern California SAS User's Group, a Section Chair and Conference Co-chair of the Western Users of SAS Software regional conference, WUSS, and in various positions at SUGI. He has developed and presented several courses and seminars on statistics and SAS programming and has taught for Colorado School of Mines, University of Redlands, and University of California at San Diego.

CALOXY offers SAS contract programming and in-house SAS training nationwide.

**AUTHOR**

Arthur L. Carpenter
California Occidental Consultants
4239 Serena Avenue
Oceanside, CA 92056-5018

(619) 945-0613
email: 72212.211@compuserve.com