Paper 049-30

Creating Flexible Graphics with Data step Graphics Interface (DSGI)

Hui – Ping Chen, Eli Lilly and Company, Indianapolis, IN

ABSTRACT

SAS PROC (procedure) is not the only vehicle for creating graphics. DATA step Graphics Interface (DSGI) combines the functions of GSLIDE, GREPLAY, and Annotate facility and enables you to generate graphic output within DATA step; not only can DSGI generate graphics faster than Annotate facility, but it also provide more flexibility than GSLIDE and GREPLAY, while working on customized graphics.

INTRODUCTION

Summary and analysis reports or descriptive statistical reports usually provide information in detail while graphics present the outline of information. Therefore, graphics allow readers to catch the key points immediately. However, a literal explanation coupled with graphics provides readers more sufficient information, especially on a presentation.

As for graphics, we can create them through SAS/GRAPH Software, such as GCHART, GPLOT, or GMAP procedure, etc. We also can combine several graphics in one output with GREPLAY procedure. If we would like to add some Text or Symbols to the graphics, perhaps the first choice for most SAS users should be Annotate facility. Therefore, we need to annotate the data within a DATA step before executing graphics procedures such as Gplot or GSLIDE. Since DSGI combines the functions of GSLIDE, GREPLAY, and Annotate facility, it enables you to create graphics output within one DATA step.

DATA STEP GRAPHICS INTERFACE (DSGI)

The functions of DSGI enable you to access graphics functions within the DATA step or through SCL (SAS Component Language), which is a programming language designed to facilitate the application interface using SAS system. Through DSGI, you can call or add features to an existing graph generated by SAS procedure (PROC) and create a customized or a drill-down graph. It is easier to add common titles or footnotes within DSGI than within GREPLAY, since title and footnote do not work in GREPLAY procedure without GSLIDE.

The example of creating a drill-down graph with DSGI will be a topic for a future paper.

FUNCTIONS IN DSGI

There are six functions to execute DSGI, which are

1) GINIT: initializing DSGI in order to activate it;
2) GRAPH: opening a graphics segment;
3) GSET: setting up attributes for an entire graph;
4) GDRAW: generating customized graphics;
5) GPRINT: printing the message of a DSGI error code and the default setting is to

print error messages automatically;

6) GTERM: ending DSGI.

To display the output you have to submit a RUN statement at the end of DSGI.

ADVANTAGES OF USING DSGI

DSGI is not only able to create a customized graph, but can also call existing graphs generated by other SAS procedures (PROC). DSGI can also display multiple graphics in one output with common titles or footnotes or, add Texts or symbols to the graphics.

When writing notes, titles, or footnotes with GSLIDE procedure, you have to pay attention to the length of each statement, since it displays entire statements in the same row, and changes the size of characters automatically if they are too long; even if you define the height of text. Under the same situation, DSGI will truncate the excessive part to match the explicit requirements. When displaying multiple graphics created by previous SAS procedures such as Gchart or Gplot with GREPLAY, you cannot remove titles/footnotes that have already been created or simply use Title/Footnote statements to add them without GSLIDE. However, title or footnote statements work in DSGI in the same way as with any other DATA step. It is convenient and flexible to display multiple graphics in one output with common titles or footnotes with DSGI.

When using DSGI to display a listing report and a graph in one output, first you need to export it as an external text file, then convert it in a graph file, since the function of DSGI, Graph ('insert'), only allows you insert existing graphs. To do this, simply export the listing report as an external text file with Printto procedure then convert it in a graph file through GPRINT procedure. This requirement is the same as using GREPLAY, but the benefit of using DSGI is easier control of the output layout and the size of characters presented in previous text files.

CREATE A SIMPLE GRAPH WITH DSGI

When creating a graph with DSGI, you need to include the functions in DSGI introduced above. The basic steps are

1) Initialize DSGI
   DSGI = Ginit ();
   Where (DSGI =) is an any return-code-variable name.
2) Open a graphic segment, so that graphics can be submitted
   DSGI = Graph ('Clear');
3) Set up attributes
   DSGI = Gset ('Texheight', 5);
   Where ('Texheight', 5) means the height of text is 5.
   DSGI = Gset ('Texfont', 'Swiss');
   Where ('Texfont', 'Swiss') means the font of text is Swiss style.
4) Generate a customized graph
   DSGI = Gdraw ('Text', 48, 70, 'Create a Simple Graph with DSGI');
   Where (48, 70) is the coordinates of X and Y and the 'Create a Simple Graph with DSGI' is the Text written in the output at the coordinates,

(48, 70).

DSGI = Gdraw (‘Pie’, 50, 0, 60, 0, 180);

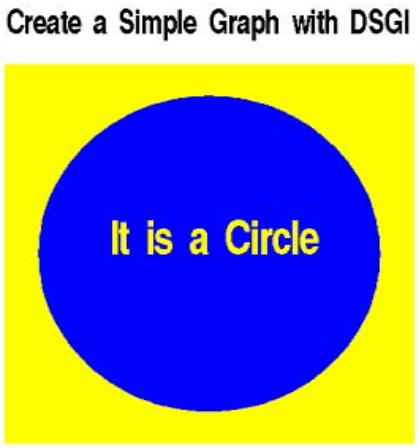
Gdraw (‘Pie’) means to draw a ‘Pie’ shape. The first two numbers in (50, 0, 60, 0, 180) are the coordinates of center of the pie; the third number is the semi-diameter; and the last two numbers mean to draw a pie from 0 degree to 180 degree (half circle). In other word, with Gdraw () function you can draw a half circle at (50, 0) with semi-diameter equal to 60.

5) Close the graphics segment
   DSGI = Graph (‘Update’);
6) Terminates DSGI
   DSGI = Gterm ();

Here is a simple example to create a circle with text annotated inside and a rectangular bar outside.

```
Data _null_;

 /*Initialize DSGI*/
 DSGI = Ginit ();

 /*Open a graphic segment*/
 DSGI = Graph (‘Clear');

 /*Set up the height of Text*/
 DSGI = Gset ('Texheight', 5);

 /*Set up the text font*/
 DSGI = Gset ('Texfont', 'Swissb');

 /*Write the title*/
 DSGI = Gdraw ('Text', 45, 70, ‘Create a Simple Graph with DSGI');

 /*Define colors*/
 DSGI = Gset (‘Colrep', 1, 'Blue');
 DSGI = Gset (‘Colrep', 2, 'Yellow');

 /*Fill solid colors in certain areas*/
 DSGI = Gset ('Filtype', 'Solid');

 /*Fill in a rectangular bar with second color, Yellow*/
 DSGI = Gset ('Filcolor', 2);

 /*Draw a rectangular bar around the circle from the low left corner of the bar,
   (45, 5), to the high right corner of the bar, (105, 65)*/
 DSGI = Gdraw('bar', 45, 5, 105, 65);
```

```
 /*Fill in circle with first color, Blue*/
 DSGI = Gset ('Filcolor', 1);

 /* Draw a circle at (75, 35), which is the coordinates of center of circle, with
    semi-diameter = 25*/
 DSGI = Gdraw ('Pie', 75, 35, 25, 0, 360);

 DSGI = Gset ('Texheight', 7);
 DSGI = Gset ('Texfont', 'Swissb');

 /*Write the text with color 2, Yellow */
 DSGI = Gset ('Texcolor', 2);

 /*Write the text inside the circle starting at (60, 35)*/
 DSGI = Gdraw('Text', 60, 35, 'It is a Circle');

 /*Close the graphics segment*/
 DSGI = Graph ('Update');

 /*Terminate DSGI*/
 DSGI = Gterm ();

/*Display output*/
Run;
```



Using FILENAME and GSFNAME (GOPTION statement) to output graph as an external graphic file while working with DSGI.

DISPLAY EXISTING GRAHPICS IN A SPECIFIED AREA

With Viewport and Window functions, DSGI enables you to display an existing graphic in a specified area. The default Viewport is defined as (0, 0) to (1, 1) while '1' represents 100 percent of the graph output area. Once the Viewport is defined, you can decide which part of an existing graph is displayed by defining Window from coordinates (0, 0)

to (100, 100). For example, if you do not want to display the titles generated by the previous procedure (PROC), you can define smaller Window coordinates to cut off the titles.

Before inserting an existing graph with Gset function, which defines Viewport and Window, you have to make a transformation to activate both Viewport and associated Window. DSGI has 21 transformations from 0 to 20. Transformation 0, by default, always uses the entire output area for Viewport and fills in the Viewport with the graph, which is further defined by the Window Coordinates. For example, if you like to separate the entire output area in 4 specified areas, you need to make transformations 4 times to activate each Viewport and associated Window, respectively.

Here is an example to display two existing graphics in one output area with DSGI.

```
Data _null_;

  /*Initialize DSGI*/
  DSGI = Ginit ();

  /*Open a graphic segment*/
  DSGI = Graph (‘Clear');

  /*Define first Viewport as (0, 0) to (1, .49)*/
  DSGI = Gset ('Viewport', 1, 0, 0, 1, .49);

  /*Define associated Window coordinates as (0, 30) to (100, 100)*/
  DSGI = Gset ('Window', 1, 0, 30, 100, 100);

  /*Transformation 1 activates Viewport and associated Window*/
  DSGI = Gset ('Transno', 1);

  /*Insert an existing graph generated by previous PROC procedure with Name = ‘Entry-
Name‘
    statement in Viewport defined above*/
  DSGI = Graph ('Insert', 'Existing Entry-Name');

  DSGI = Gset ('Viewport', 2, 0, .51, 1, 1);
  DSGI = Gset ('Window', 2, 0, 30, 100, 100);
  DSGI = Gset ('transno', 2);
  DSGI = Graph ('Insert', 'Existing Entry-Name');

  /*Close the graphics segment*/
  DSGI = Graph ('Update');

  /*Terminate DSGI*/
  DSGI = Gterm ();
```

```
/*Display output*/
Run;
```

DISPLAY AN EXTERNAL TEXT FILE WITH DSGI

You cannot insert text output with DSGI directly, only graphics output. GPRINT allows you to convert text output in graphics output with some restriction. In other word, you have to export text output as an external text file before converting it in a graphics file with GPRINT. For example, using Printto with GPRINT is one way. The Name = 'Entry-Name' statement in GPRINT procedure enables you to call this graph later with DSGI. The next step is to define Viewport and associated the Window for displaying this graph in a specified output area.

Here is an example to display an external text file with DSGI.

```
Filename External 'Destination of Output';

/*Print the output result in a file*/
Proc Printto File = External New;
Run;

Proc Print Data = Dataset Name;
Run;

/*Execute Printto procedure*/
Proc Printto;
Run;

/*VPOS: Sets up the number of rows in the graphics output area.
  HPOS: Sets up the number of columns in the graphics output area.
  Using VPOS and HPOS to adjust the size of characters*/
Goption Reset = All Device = Device Name VPOS = 43 HPOS = 104;

/*Convert an external file stored in 'External' defined above.
  (FILEREF = ) specifies the external file (for example, External), which will be input to
  the GPRINT procedure. In addition, 'External' must have been previously defined in a
  Filename statement as shown above. (Name = ) specifies a name as a catalog entry
  name for this graph, which will be used with DSGI function, Graph ('Insert', 'Entry-
  Name'), later.
Proc GPRINT FILEREF = External Name = 'Entry-Name';
Run;

Data _null_;

 /*Initialize DSGI*/
 DSGI = Ginit();
```

```
 /*Open a graphics segment*/
 DSGI = Graph('Clear');

 /*Define first Viewport as (.05, .05) to (.85, .85)*/
 DSGI = Gset('Viewport', 1, .05, .05, .85, .85);

 /*Define associated Window coordinates as (0, 0) to (100, 100)*/
 DSGI = Gset('Window', 1, 0, 0, 100, 100);

 /*Transformation 1 activates Viewport and associated Window*/
 DSGI = Gset('Transno', 1);

 /*Inset an existing graph generated by Printto procedure with Name = 'Entry-Name'
   statement in Viewport defined above*/
 DSGI = Graph('Insert', 'Existing Entry-Name');

 /*Close the graphics segment*/
 DSGI = Graph('Update');'

 /*Terminate DSGI*/
 DSGI = Gterm();

/*Display output*/
Run;
```

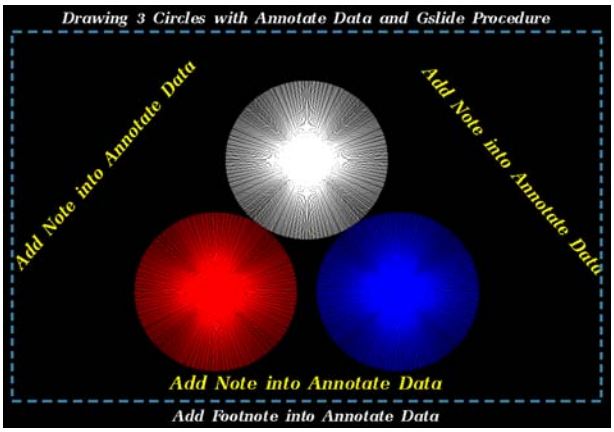
Through DSGI, it is easy to display an external text file with a related existing graph in one output. GREPLAY also enables you to do the same thing; however, not like DSGI, it is not so easy to control the size of characters for both external text files and existing graphs.
On the other hand, DSGI can add titles or footnotes easily with Title or Footnote statements while you have to use GSLIDE to create titles or footnotes for GREPLAY.

ANNOTATE TEXT OR SPECIAL ELEMENTS IN GRAPHS WITH DSGI

DSGI has the same function as Annotate facility to enable you to generate customized graphics by adding text or special elements to the graphics. Here are two examples of graphics creation with annotated text through Annotate DATA step and DSGI, respectively. Both graphics are very similar. The main difference is that you need to use GSLIDE to display Annotate data but with DSGI directly.

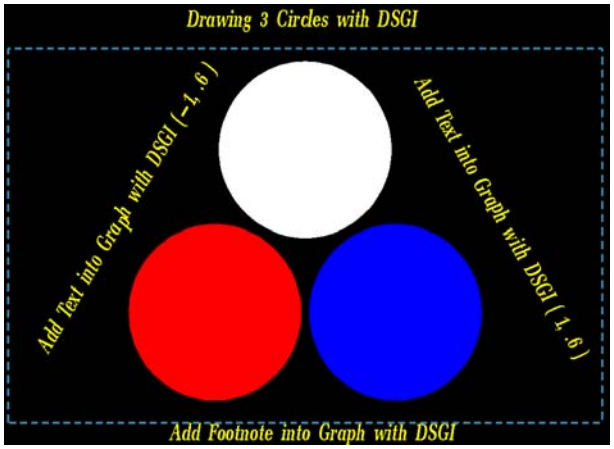
```
/*Create an annotate data, which draws 3 circles*/
Data Art;
  length function color $ 8 angle 3.0;
  retain xsys ysys hsys '5';
  do circle = 0 to 360;
     function = 'Pie'; x = 50; y = 65; size = 21; angle = circle; color = 'White'; output;
```

```
      function = 'Pie'; x = 35; y = 30; size = 21; angle = circle; color = 'Red'; output;
      function = 'Pie'; x = 65; y = 30; size = 21; angle = circle; color = 'Blue'; output;
  end;
Run;

/*Display annotated data which contains 3 circles.
  Add title, note, and footnote into Annotate data*/
Proc GSLIDE annotate = Art border wframe = 4 lframe = 3 cframe = steel;
 title f = zapfbi c = white 'Drawing 3 Circles with Annotate Data and GSLIDE
Procedure';
 note f = zapfbi c = yellow h = 2.3
            j = l a = 50 'Add Note in Annotate Data'
            j = r a = -50 'Add Note in Annotate Data';
 note h = 12;
 note f = zapfbi c = yellow h = 2.3
            j = c a = 0 'Add Note in Annotate Data';
 footnote f = zapfbi c = white h = 2 'Add Footnote into Annotate Data';
Run;
```




```
/*Draw 3 circles same as previous procedures did*/
Data _null_;
 DSGI = Ginit();
 DSGI = Graph('Clear');

 /*Assign 5 colors to each number*/
 DSGI = Gset('Colrep', 1, 'White');
 DSGI = Gset('Colrep', 2, 'Red');
 DSGI = Gset('Colrep', 3, 'Blue');
 DSGI = Gset('Colrep', 4, 'Yellow');
 DSGI = Gset('Colrep', 5, 'Steel');

 /*Fill type = Solid*/
 DSGI = Gset('Filtype', 'Solid');
```

```
/*The first circle is White*/
DSGI = Gset('Filcolor', 1);
DSGI = Gdraw('Pie', 70, 67, 20, 0, 360);

DSGI = Gset('Filcolor', 2);
DSGI = Gdraw('Pie', 49, 30, 20, 0, 360);

DSGI = Gset('Filcolor', 3);
DSGI = Gdraw('Pie', 91, 30, 20, 0, 360);

DSGI = Gset('Texheight', 5);
DSGI = Gset('Texfont', 'zapfbi');
DSGI = Gset('Texcolor', 4);
DSGI = Gdraw('Text', 38, 95, 'Add Title into Graph with DSGI');
DSGI = Gdraw('Text', 38, 1, 'Add Footnote into Graph with DSGI');

/*Rotate the text string.
  The natural angle of text string is toward 6 o'clock when the value for X and Y is
  (1.0, 0.0) or is toward 3 o'clock when the value for X and Y is (0.0, 1.0).
  If you change the values of X and Y, the coordinates which has the highest value will
  be taken as Angle and the Lowest value will be taken as Offset*/
DSGI = Gset('Texup', -1, .6);
DSGI = Gdraw('Text', 10, 20, 'Add Text into Graph with DSGI (-1, .6 )');
DSGI = Gset('Texup', 1, .6);
DSGI = Gdraw('Text', 95, 83, 'Add Text into Graph with DSGI ( 1, .6 )');

/*Assign color 5, Steel, as line color*/
DSGI = Gset('Lincolor', 5);

/*Set up the width of line*/
DSGI = Gset('Linwidth', 4);

/*The values for the type of line are from 1 to 46.
  The type 1 represents solid line and type 3 is dash line*/
DSGI = Gset('Lintype', 3);

/* Draw a frame with 'Line' function.
   Return Code Variable = Gdraw('Line', number of points, X-values, Y-values).
   That is, drawing a line to connect 5 points starting from (1, 5) -> (1, 90) -> (139, 90)
   -> (139, 5) and ending at (1, 5).
DSGI = Gdraw('Line', 5, 1, 1, 139, 139, 1, 5, 90, 90, 5, 5);

DSGI = Graph('Update');
DSGI = Gterm();
Run;
```



When using the Annotate DATA step to draw circles, as shown above, it draws 360 lines to display a circle; therefore, the output does not look like thuse created by DSGI, which are solid circles. You can also display circles with different Filtype options such as 'Hatch' (polygon-type patterns), 'Hollow' (empty), or 'Pattern' (bar-type patterns). When using 'Hatch' or 'Pattern' as the Filtype, you have to specify the style of the filling area by setting up Filstyle from 1 to 60 or from 1 to 15, respectively. For example,

```
DSGI = Gset(('Filstyle', 10);
DSGI = Gset('Filtype', 'Pattern');
```

CONCLUSION

DSGI provides us with another option for generating customized graphics. DSGI is not only able to create graphics but it is also able to insert more than one existing graphic created by another PROC procedure, especially external text files. DSGI is more efficient for creating presentation slides than other software because you can use the output generated by SAS directly. In addition, DSGI combines the functions of GREPLAY, GSLIDE, and Annotate facility within a single DATA step. That is, DSGI is very flexible and efficient. You can also save the graphics created with DSGI and work with other software later. Therefore, SAS PROC (procedure) is not the only choice for creating graphics.

REFERENCES

SAS® Institute (1999), SAS/GRAPH® Software: Reference, Version 8,
Copyright 1999 by SAS® Institute Inc., Cary, NC, USA

CONTACT INFORMATION

Your comments and questions are valuable and appreciated. Author can be reached at

Hui-Ping Chen
Eli Lilly and Company
Lilly Corporate Center
Indianapolis, IN 46285 U.S.A.
Email: huiping@lilly.com