New features in Version 5 of SAS/GRAPH software have made the product much more flexible. New procedure options, AXIS and LEGEND statements, and the ANNOTATE facility give you control over almost every aspect of your graphics output. Everything anyone could ever ask for in a business graph, right? Well, maybe you want to have control over a few more items.

This is a presentation of a graphics tool that gives you additional control over SAS/GRAPH procedure output. The tool is a SAS® DATA step program called METATATE. It reads a Metagraphics facility metafile and produces an ANNOTATE data set. This ANNOTATE data set can be used with PROC GANNO to exactly replicate the original graphics procedure output. By modifying the ANNOTATE data set you can alter the output from any graphics procedure to suit your needs.

To understand the METATATE program, we need background information on the Metagraphics and ANNOTATE facilities.

The Metagraphics Facility

The Metagraphics facility was introduced with Version 5 of SAS/GRAPH software to give hardware vendors and software developers the necessary tools to write drivers for graphics hardware currently not supported by the SAS System. It is based on the simple concept of an internal SAS/GRAPH device driver generating a file of generic graphics commands (metacode) that describe the graph you are creating. These commands correspond to common graphics primitives like MOVE to (x,y), DRAW to (x,y), RECTANGLE from lower left (x,y) to upper right (x, y), use COLOR (i), and so forth.

The idea behind the Metagraphics facility is that an internal driver will generate the metacode that is appropriate for the capabilities of your device. For example, if your device has the built-in intelligence to draw a circle, the metacode describing a pie chart might contain a CIRCLE command giving the coordinates for the center of the circle as well as the diameter. The job of the SAS user trying to support a new device is to write a computer program, in any language, that will translate the metacode into the actual graphics commands their device understands. This program is called an external device driver. (For more information on the technical details of the Metagraphics facility, consult SAS Technical Report P-155, "SAS/GRAPH® Metagraphics Driver Facility.")

Developing testing strategies for the Metagraphics facility led to the conclusion that it would be nice to be able to display the picture described by a metafile on any supported graphics device without actually writing the external driver for that device. The solution to the problem was to write a fairly simple SAS DATA step program (which I call METATATE) that could read a metafile and produce as output an ANNOTATE data set that describes the same picture.

The ANNOTATE Facility

An ANNOTATE data set is a special SAS data set that enables you to customize SAS/GRAPH procedure output. Its most common use is to add special labels or markers to charts and plots. However, it can also be used to describe a whole picture. For example, all of the word slides in this presentation are purely the creation of an ANNOTATE data set used with PROC GANNO.

Each observation in an ANNOTATE data set describes all or part of a graphics function that you want performed. Three of the required variables are FUNCTION, X, and Y. Basically, the value of FUNCTION tells the ANNOTATE facility what you want done and X and Y tells where you want it done. For example, an ANNOTATE data set with these two observations would produce a horizontal line from coordinates (10,10) to (20,10).

```
OB  FUNCTION  X  Y
  1   MOVE    10  10
  2   DRAW    20  10
```

Other variables can be added to the data set that control the width, color, and line style used, as well as the type of coordinate system used.

Another common FUNCTION value is LABEL. An ANNOTATE data set with this observation would write the text string, "Hi", at coordinate (10,10).

```
OB  FUNCTION  X  Y  TEXT
  1   LABEL   10  10  Hi
```

Again, other variables can be added that would change the color, font, and size of the text from the default. For complete documentation of the options and functions available in the ANNOTATE facility, refer to the SAS/GRAPH® User's Guide, Version 5 Edition.
The METATATE Program

As it turns out, all of the graphics primitives that can be produced in a Metagraphics metafile have a counterpart in the ANNOTATE data set. That means that the METATATE program can be used to translate any metafile into an ANNOTATE data set. Or more specifically, the output from any SAS/GRAPH procedure can be translated into an ANNOTATE data set. To facilitate the remaining discussion, this ANNOTATE data set will be referred to as a METATATE data set. There are several practical benefits that can be derived from having procedure output in a METATATE data set.

Once you have SAS/GRAPH procedure output in a METATATE data set you are free to make low-level modifications of the picture. While Version 5 of SAS/GRAPH software gives you some control over most aspects of your graphics output, the METATATE program gives you complete control.

How might we put this program to use?

Use of the METATATE Program

The METATATE program requires that you have an ATTRIBUTE file available to the Metagraphics facility. For the applications discussed in this paper, the QCR2X driver, a metagraphics driver supplied with SAS/GRAPH software, can be used. (Since there is some CPU overhead here due to the execution of the QCR external driver you may want to create your own ATTRIBUTE file that does not require an external driver. For information on how to do this, refer to the previously mentioned Technical Report P-155. The sample GXTALINK attribute file provided with SAS/GRAPH software would be a good starting place.) Here is a sample of how it might be used under TSO:

GOPTIONS DEV=QCR2X;
TSO ALLOCATE DD(QVPIN) NEW;
*The metacode is written to QVPIN;
TSO ALLOCATE DD(QVPOUT) DUMMY;
*We don't need the real driver output;
Graphics PROC you want processed by METATATE;
%INCLUDE DDNAME(METATATE)/NOSOURCE;
*Bring in the METATATE program;
*It reads the QVPIN file and produces;
*WORK.METATATE, an ANNOTATE data set;
GOPTIONS DEVICE=your_device HPOS=40 VPOS=25;
*see note below about HPOS and VPOS;
PROC GANNO ANNO=METATATE;
RUN;

Note: Because the METATATE data set created is based on the SCREEN CELL coordinate system, it is necessary to specify the HPOS= and VPOS= options that reflect the defaults for the Metagraphics driver you use to create the metafile.

Practical Applications of the METATATE Program

Elimination of part of your graph. While SAS/GRAPH procedures give you the option of eliminating almost any default labeling, there are still a few situations where this is not the case. Removing selected portions of a graph from its METATATE representation is usually a simple matter of having a DATA step delete some observations on the basis of the value of one or more of the ANNOTATE variables.

For example, using the ACROSS= option on the PIE statement in PROC GCHART will produce multiple pie charts on a page. Each pie has a default header indicating the value of the GROUP variable. The procedure offers no way to eliminate this header. On page 216 in the SAS/GRAPH User's Guide, Version 5 Edition, there is a two-pie graph with the headers "MERIDIAN = AM" and "MERIDIAN = PM". These could easily be removed from a METATATE data set with the following DATA step statements:

IF SUBSTR(TEXT,1,8)="MERIDIAN" THEN DELETE;

Changing details on the graph. PROC GCHART does not allow control over the font of the pie headers. If, in the above example, you wanted to have the headers printed in the triplex font and with a larger size, you could do so by processing the METATATE data set with the following DATA step statements:

IF SUBSTR(TEXT,1,8)="MERIDIAN" THEN DO;
  STYLE="TRIPLEX";
  SIZE=2;
END;

Of course, it is possible to change the text in such a way that it is no longer centered or the label overlays a portion of the graph. You are, however, in complete control of the graph and can make any necessary adjustments.

Adding a font to PROC GPRINT. A similar application is to get PROC GPRINT to use a font. The primary problem encountered when trying to print procedure output with a font is keeping the text properly aligned. Since fonts produce proportional spacing, text that is lined up on a line printer will not be lined up when plotted with a font. A METATATE data set of PROC GPRINT output can compensate for this difficulty with the following code:

/*Make a separate observation for each character in the TEXT variable.
  Center each character in a space 1.2 cells wide. */
DATA METATATE;
  / * Set METATATE data set created is based on the SCREEN CELL coordinate system, it is necessary to specify the HPOS= and VPOS= options that reflect the defaults for the Metagraphics driver you use to create the metafile.
  */
    #Make a separate observation for each character in the TEXT variable.
    Center each character in a space
    1.2 cells wide. */
    DATA METATATE;
    SET METATATE;
    style="xswissb"
    length=length(text);
    temp(text)=text;
    temp(x)=(x+1)*1.2;
    position='b'
DO I=1 TO LENGTH;
   TEXT=SUBSTR(TEMPTEXT,I,1);
   X=(I-.2)+TEMPX;
   OUTPUT;
END;

Each observation in the METATATE data set is a 
LABEL="FUNCTION" with TEXT= to an entire 
line from the file to be printed. The code above 
changes this into a single observation per 
character in the TEXT= string, centers each 
character in a position 1.2 character cells wide, 
and changes the font to XSWISSB. (The same 
result could have, of course, been accomplished 
in a single DATA step that reads an external file 
and generates an ANNOTATE data set, but 
starting with the METATATE data set simplifies 
the process.)

The same sort of process could also be applied to 
adding a font to the VBAR statistics printed by 
PROC GCHART. It is easy to isolate the 
statistics from the other observations in the 
METATATE data set if you have produced the 
plot in such a way that this text is the only text 
with no font or is the only portion of the graph 
ploitted with a particular color:

IF FUNCTION='LABEL' AND STYLE='NONE'
   THEN STYLE='TRIPLEX';

or

IF FUNCTION='LABEL' AND COLOR='SPECIAL'
   THEN STYLE='TRIPLEX';

Creating fonts from procedure output. If you 
create a METATATE data set from a metafile 
created with the following statement in effect,

GOPTIONS NOCHARACTERS NOFILL NOPOLYGONFILL
   NOPOLYFILL NOFILL NOSYMBOLS;

the only functions that will appear will be MOVE 
and DRAW. That is because you have told 
SAS/GRAPH that you do not want to use any of 
the special hardware capabilities of the 
Metagraphics driver. This means that all text, 
polygons, symbols, and so forth, will be 
produced with simple MOVE and DRAW commands.

The METATATE data set can then be used as 
input to the GFONT procedure to create a font 
from the output of another graphics procedure. 
For example, first produce a METATATE data set 
of the famous G3D cowboy hat (see the SAS/GRAPH 
the NOAXIS option on the PLOT statement so all 
you have are the MOVE and DRAW commands to 
create the hat itself. The METATATE data 
can be made ready for PROC GFONT with the 
following code:

DATA FONT; SET METATATE;
   CHARS='A';
   IF FUNCTION='MOVE' THEN SEGMENT+1;
   IF SEGMENT>256 THEN DO;
      PUT 'TOO MANY SEGMENTS FOR GFONT!';
      STOP;
   END;
   PROC GFONT NAME=COWBOY DATA=FONT RESOL=4;

Now you have a font called COWBOY that is a 
recreation of the PROC G3D output. Surely 
someone can find a use for this application.

Combining graphics from two procedures. Most 
people are satisfied using the template features 
of PROC GREPLAY when they want to combine 
the output from two graphics procedures. If this 
is not what you want, use the ANNOTATE facility 
to add the desired information to the output from 
a single procedure. But how can you put 
GCONTOUR output into a G3D graph of the same 
data?

First, create a METATATE data set of the 
GCONTOUR output using the NOAXIS option. 
Now all you need to do is rescale the X and Y 
coordinates (these are based in a characters cell 
coordinate system) into the limits of the data 
used to create the graph in the first place. It is 
not as hard as it might seem. The following 
example is used to put a CONTOUR plot of the 
cowboy hat data on the floor of the G3D of the 
same data.

/*Create the cowboy hat data, X and Y 
will range from -5 to +5. */
DATA COWBOY;
   DO X=-5 TO 5 BY .25;
      DO Y=-5 TO 5 BY .25;
         Z=SIN(SQRT(X*X+Y*Y));
         OUTPUT;
      END;
   END;
/*Create the metacode of the contour plot 
that we will put on the floor of the G3D */
GOPTIONS DEV=QGR2X;
PROG GCONTOUR DATA=COWBOY;
   PLOT Y=X ZI;
   NOAXIS;
/*Bring in METATATE to process metafile. */
%ING IN(METATATE);
/*Get range of X and Y for rescaling. */
PROC MEANS DATA=METATATE NOPRINT;
   VAR X Y;
   OUTPUT OUT=STATS
      MAX=XMAX YMAX MIN=XMIN YMIN;
/*Rescale X and Y to -5 to 5 range. 
Add Z=2 and change to DATA SYSTEM. */
DATA NEW;
   RETAIN XRANGE YRANGE XMIN YMIN;
   XRANGE=2*XMAX;
   YRANGE=2*YMAX;
   XMIN=-5;
   YMIN=-5;
   Z=Z+2;
   INPUT X Y Z;
   DATA SYSTEM;
   OUTPUT;
*/
Adding colors to a PROC G3D surface. One of the prettiest applications of METATATE is to add color to your graph in ways the PROC you are using will not allow. The following example takes a METATATE data set representing a G3D plot and adds several colors to various portions of the surface plot.

/*divide observations into 7 colors. */
DATA NEW;
SET METATATE POINT=POINT NOBS=NOBS;
NUM=INT(~_/(NOBS/7»;
IF NUM=0 THEN COLOR='BLUE'j ELSE IF NUM=1 THEN COLOR='RED'j ELSE IF NUM=2 THEN COLOR='PINK'; ELSE IF NUM=3 THEN COLOR='GREEN'; ELSE IF NUM=4 THEN COLOR='CYAN'; ELSE IF NUM=5 THEN COLOR='YELLOW'; ELSE COLOR='WHITE';
RUN;

This will produce a G3D surface with bands of different colors. Using the procedure alone you are limited to a single color for the top surface and a single color for the bottom surface.

Adding color shading to polygon fills. If you have a graphics device that is capable of producing 256 colors at a time, it is possible to create some beautifully shaded area fills. For example, the bars in a horizontal bar chart could be gradually shaded from left to right from a cyan to a dark blue to magenta, crowning the tips of the longest bars with a bright red. Here is how to do it.

This example takes advantage of the fact that software rectangle fill is always stroked vertically. By specifying NOFILL in a GOPTIONS statement when the metafile is created, the solid area in the bars are stroked. When the METATATE data set is processed, we assign colors to the lines that make up the solid bar on the basis of their location in the X dimension. This is done with an algorithm that constructs an HLS-type color specification (see p. 20 in the SAS/GRAPH User’s Guide, Version 5 Edition for more information on user-defined HLS colors). The following program will shade the bars with 180 different hues starting with a hue of 300 (cyan) up to 360 (blue) and then from 0 (blue) to 120 (red). The lightness factor of the color is held constant at a medium level of ‘7F’. A constant full saturation level is used (‘FF’).

/*STEP determines the number of X values per color using 180 colors. For each X in the bar we determine which STEP it is in and create the proper HUE. Saturation and lightness components stay constant. */
DATA NEW;
RETAIN STEP XMIN;
IF ~X=1 THEN DO;
SET STATS;
STEP=(XMAX-XMIN)/180;
END;
SET METATATE;
IF COLOR='PATTERN' THEN DO;
HUE=320+«X-XMIN)/STEP);
END;
IF HUE>360 THEN HUE=HUE-360j
COLOR='H'II
PUT(HUE,HEX3.)II
'7FFF' ;
SIZE=2;
END;
DROP XMIN XMAX STEP HUE;
RUN;

Color separation for graphics printing. While adding colors to a graph may be a fun and interesting application, perhaps a more useful application of the METATATE program is to pull parts of the graph out on the basis of color. The ability to create individual black on white images that represent each of the unique colors in a graph can greatly reduce the cost of graphics printing.

The following example is rather sophisticated, but can be simplified if necessary. It takes the METATATE data set and produces a new ANNOTATE data set for each color in the original graph. Each of these pictures has registration marks at each corner to help the alignment of the pictures during printing. Also, at the bottom of each picture is the name of the color that should be used to print that portion of the picture.
MACRO COLORS;
  %DO COLOR=1 %TO &PENMOUNT;
    COLOR &COLOR
  %END;
%MEND COLORS;
%MEND OUTPUT;
%MEND ANNOTATE;
%DO 1=1 %TO &PENMOUNT;
  PROC GANNO ANNO=COLOR &I;
%END;
%MEND ANNOTATE;

PROC SORT DATA=METATATE; BY COLOR;

DATA TICK;
  XSYS='4'; YSYS='4'; COLOR='.';
  *lower left registration mark;
  FUNCTION='MOVE'; X=4; Y=3; OUTPUT;
  FUNCTION='DRAW'; X=3; Y=3; OUTPUT;
  FUNCTION='DRAW'; X=3; Y=4; OUTPUT;
  *upper left registration mark;
  FUNCTION='MOVE'; X=4; Y=54; OUTPUT;
  FUNCTION='DRAW'; X=3; Y=54; OUTPUT;
  FUNCTION='DRAW'; X=3; Y=53; OUTPUT;
  *upper right registration mark;
  FUNCTION='MOVE'; X=83; Y=54; OUTPUT;
  FUNCTION='DRAW'; X=84; Y=54; OUTPUT;
  FUNCTION='DRAW'; X=84; Y=53; OUTPUT;
  *lower right registration mark;
  FUNCTION='MOVE'; X=83; Y=3; OUTPUT;
  FUNCTION='DRAW'; X=84; Y=3; OUTPUT;
  FUNCTION='DRAW'; X=84; Y=4; OUTPUT;
  *Maximum number of colors in the graph;
  LET PENMOUNT=5;
  /*Move everything up and over so there is room for the registration marks. Change all colors to black. Add notation on the graph to indicate the color to be produced.*/
  DATA %COLORS;
    SET METATATE (IN=INANNO) TICK (IN=INTICK);
    BY COLOR;
    IF INANNO THEN DO;
      X=X+4; Y=Y+4;
    END;
    OUTPUT COLOR &PENMOUNT;
    SCOLOR=COLOR; COLOR='BLACK';
    IF INANNO AND FIRST.COLOR THEN OUT=1;
    IF INANNO THEN DO; %OUTPUT END;
    IF (INANNO AND FIRST.COLOR) THEN DO;
      FUNCTION='LABEL';
      X=3; Y=0; POSITION='3'; TEXT=SCOLOR;
      STYLE='NONE'; SIZE=1; %OUTPUT;
    END;
    IF INTICK THEN DO;
      COLOR='BLACK';
    OUTPUT %COLORS;
    END;
  */Add 8 to HPOS= and VPOS= of Metagraphics driver to create extra room for the registration marks. */
  OPTIONS DEV=your_device HPOS=88 VPOS=58;
  %ANNOTATE

How to Get a Copy of the METATATE Program

Unfortunately, the METATATE program is too lengthy (about 600 lines, 400 without comments) to list here. I will send you a hardcopy on request. If you would like a copy on PC diskette, please send a formatted diskette with your request. This program will be included in the SAS/GRAPH software sample library in future releases.

Conclusion

While the METATATE program may not be suited for the casual SAS/GRAPH user, it offers a new degree of control over procedure output, never before available to the power-user of SAS/GRAPH software. The applications offered here are merely a beginning. The possibilities are limited only by your imagination.

Acknowledgments

I would like to thank my former manager, Mike Kalt, for providing me the time and support to develop this project. Mike has greatly facilitated my education in the application of SAS/GRAPH software.

SAS and SAS/GRAPH are registered trademarks of SAS Institute, Inc., Cary, North Carolina, USA.