PharmaSUG 2016 Beijing – Paper 52

Scalable Vector Graphics (SVG) using SAS

Yang Wang, Seattle Genetics, Inc., Bothell, WA
Vinodita Bongarala, Seattle Genetics, Inc., Bothell, WA

ABSTRACT

Scalable Vector Graphics (SVG) is an XML-based vector graphic format, compatible with most modern browsers. Unlike pixel-based graphics which lose resolution when enlarged, Scalable Vector Graphics can be magnified infinitely without loss of quality for any screen resolution and size. Starting with SAS 9.2 version, Scalable Vector graphics is supported through Vector Graphics device. SVG can be produced by Graph Template Language (GTL) and traditional SAS/GRAPH procedures like PROC GPLOT/GCHART.

This paper explains the difference between the two kinds of graphs and discusses the pros and cons of using them for clinical outputs. SAS features such as GTL and SGRENDER can be efficiently used to produce high quality SVG. Examples that use traditional SAS/GRAPH such as PROC GPLOT/GCHART procedures to produce SVG graphs are also included. This paper will serve the following purposes:

- Help decide what kind of graphics format to produce to meet specific needs
- Quick Start guide to learn to use GTL to create vector graphics
- Quick conversion of current pixel-based graphics to vector graphics.

INTRODUCTION

SVG was developed by Adobe Systems as an XML-based vector graphic format. It is the recommended standard to describe 2-D vector graphics on the Internet by the World Wide Web Consortium (W3C). It is editable in any text editor and compatible with most modern browsers.

With the 9.2 version, SAS now has the capability to create graphics in the SVG standard. Being a XML-based file, it can be searched, scripted, edited with any text editor. The SVG images are created mathematically using a set of shapes and paths. Because the shape elements can be scaled up and down the resolution is not lost during resizing of the image. The SVG includes formats like EMF, CGM, PDF.

Some of the advantages of the SVG images are listed below:

Advantages:

- Less consumption of memory space.
- The resolution is preserved during scaling. It is not impacted by zooming in or out.
- It can interact with Java scripts to animate images.

Disadvantages:

- They cannot make complex photo images
- Harder or impossible to add some effects
- Not supported by older browsers.

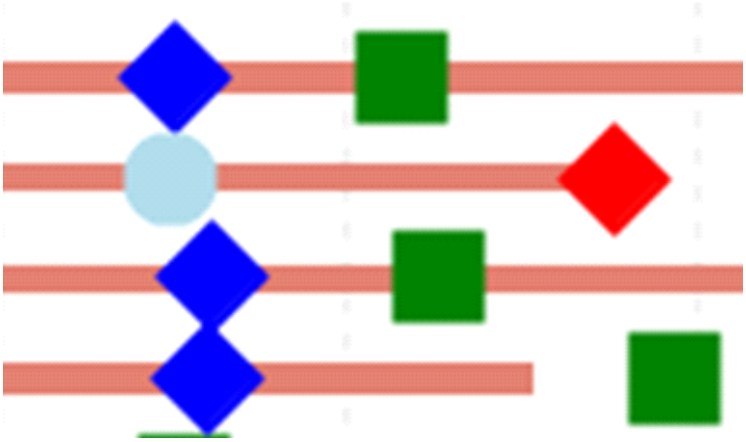
Pixel-based images like JPEG, GIF, PNG, BMP etc. are composed from small points called pixels. They support more colors and are great for clear photographic images. However, the color information of each pixel needs to be stored separately for any kind of image. For larger images this causes significant memory consumption. Pixel-based images are not very scalable. Enlarging causes loss in the quality of the image, giving it a distorted appearance. On the contrary, shrinking an image causes deletion of pixels, merging of pixels and results in the image being blurry.

SAS is capable of producing both SVG and pixel-based images. It is the user's prerogative to choose which format is best for the deliverable depending on the requirement. Due to the inherent advantages and disadvantages, both types of graphic formats are suitable for different purposes. SVG would serve as a viable choice for web publications,

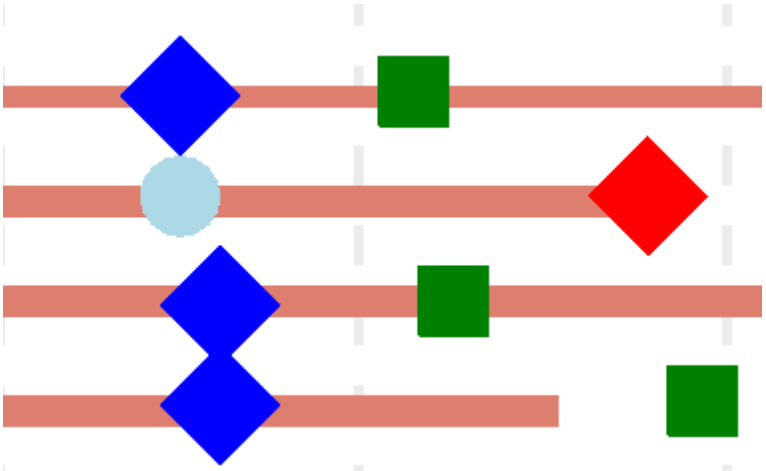
animation, conference poster, banners and logo design. Pixel-based images would be an ideal choice for scientific journal, text books and color rich photo images.

VISUAL COMPARISON OF THE TWO TYPES OF IMAGES

Here is a visual representation of the two kinds of images for comparison



PIXEL-BASED IMAGES



SVG IMAGES

Figure 1. A pixel-based PNG image (top) and an SVG EMF image (bottom) are enlarged 500x. The resolution is well maintained in the SVG image but not the pixel-based image.

USE GTL TO CREATE VECTOR GRAPHICS WITH ODS IN SAS

The structure of GTL contains two PROC TEMPLATE blocks and one PROC SGRENDER block. All the GTL graphs share this structure. ODS is used to generate SVG and is called before the SGRENDER procedure.

Example code: Sample codes of the two PROC TEMPLATE blocks. The first block defines the general display style, the 2nd block adds more customized display features.

```
/* This block defines basic graph style */
proc template;
    define style sty_custom;
    parent=rtftnr10;
    style graphfonts from graphfonts;
    end;
run;

/* This block defines styles of all the elements of the graph, including
bars, plots, legend */
proc template;
define statgraph Graph;

dynamic _USUBJID _TRTDURW _DXDIAGLU _USUBJID2 _OSW1 _DXDIAGLU2 _DEATHDUR
_USUBJID3 _DXDIAGLU3 _CRDUR _USUBJID4 _DXDIAGLU4 _PRDUR _USUBJID5
_DXDIAGLU5 _PDDUR _USUBJID6 _DXDIAGLU6 _CONTINUE _USUBJID7 _DXDIAGLU7
_SDDUR _USUBJID8 _USUBJID9 _TRTDURW2 _TRT1P _TRT1PA;

begingraph /    designwidth=750 designheight=570 border=false;
   layout lattice /   rowdatarange=data columndatarange=data
                                  rowgutter=10  columngutter=10;
      layout overlay / axisopts=(display=(TICKS LINE LABEL
                                  TICKVALUES)
               Linearopts = (     viewmin=0.0 viewmax=150.0
                                  tickvaluesequence=( start=0.0 end=110.0
                                  increment=10.0))
               griddisplay = on   gridattrs=(pattern=2)
                                  tickvalueattrs=(size=8 family='Arial'))
               yaxisopts = (tickvalueattrs=(size=8 family='Arial'));

               barchart     x=_USUBJID y=_TRTDURW /
                            group=_TRT1P name='bar_h'
                            display=(FILL)  orient=horizontal barwidth=0.3
                            groupdisplay=Cluster clusterwidth=0.85;
               scatterplot x=_DEATHDUR y=_USUBJID3 /
                            name='scatter'
                            markerattrs=(symbol=CIRCLEFILLED size=10 )
                            legendlabel='Death';
               scatterplot x=_CRDUR y=_USUBJID4 /
                            name='scatter2'
                            markerattrs=(symbol=SQUAREFILLED size=9
                            color=green ) legendlabel='CR';
               scatterplot x=_PRDUR y=_USUBJID5 /
                            name='scatter3'
                            markerattrs=(symbol=DIAMONDFILLED size=9
                            color=blue ) legendlabel='PR';
               scatterplot x=_PDDUR y=_USUBJID6 /
                            name='scatter4'
                            markerattrs=(symbol=DIAMONDFILLED size=9
                            color=red ) legendlabel='PD';
               scatterplot x=_CONTINUE y=_USUBJID7 /
                            group=_TRT1PA name='scatter5'
                            markerattrs=(symbol=DIAMONDFILLED size=7)
                            legendlabel='Treatment Ongoing';
               scatterplot x=_SDDUR y=_USUBJID8 / name='scatter6'
                            markerattrs=(symbol=CIRCLEFILLED size=9
                     color=lightblue ) legendlabel='SD';
```

```
                discretelegend    'bar_h' 'bar_h2' 'scatter' 'scatter2'
                                  'scatter3' 'scatter4' 'scatter6
                                  'scatter5' / border=true halign=right
                                  valign=bottom title="EE Patients (N=&pts)
                                  displayclipped=true across=1
                                  order=rowmajor location=inside
                                  titleattrs=(family='Arial' size=9)
                                  valueattrs=(family="Arial" size=8pt);
      endlayout;
   endlayout;
endgraph;

end;
run;
```

Example code: SGRENDER procedure renders the data from a dataset and displays the data using the template defined above.

```
/*This block associate dataset variables and their values to the template*/
proc sgrender data=WORK.ADSL template=Graph;
    dynamic      _USUBJID="USUBJID"
                 _TRTDURW="TRTDURW"
                 _DXDIAGLU="DXDIAGLU"
                 _USUBJID2="USUBJID"
                 _OSW1="OSW1"
                 _DXDIAGLU2="DXDIAGLU"
                 _DEATHDUR="DEATHDUR"
                 _USUBJID3="USUBJID"
                 _DXDIAGLU3="DXDIAGLU"
                 _CRDUR="CRDUR"
                 _USUBJID4="USUBJID"
                 _DXDIAGLU4="DXDIAGLU"
                 _PRDUR="PRDUR"
                 _USUBJID5="USUBJID"
                 _DXDIAGLU5="DXDIAGLU"
                 _PDDUR="PDDUR"
                 _USUBJID6="USUBJID"
                 _DXDIAGLU6="DXDIAGLU"
                 _CONTINUE="CONTINUE"
                 _USUBJID7="USUBJID"
                 _DXDIAGLU7="DXDIAGLU"
                 _USUBJID8="USUBJID"
                 _SDDUR="SDDUR"
                 _TRT1P="TRT1P"
                 _TRT1PA="TRT1PA" ;
run;
```

The output format is specified in the statement: ODS graphics / imagename ='testplot' imagefmt = emf. ODS statement is called before the SGRENDER procedure above, the SVG output format EMF and style defined above is specified as this example code below. Note the final output format is still .rtf, only the embedded graph is in SVG format.

```
ods graphics on/reset imagefmt=emf;
```

```
ods rtf file="f-trt-dur-ee-pharmaSUG2016.rtf"  path="&outdir."
style=sty_custom;
```

Different ODS destinations are more compatible with certain specific output formats.

- ODS HTML is more compatible with SVG
- ODS LISTING is more compatible with EMF, PDF and SVG
- ODS PFD default to vector format
- ODS PRINTER is more compatible with EMF, PCL, PDF, PS and SVG
- ODS RTF is more compatible with EMF.

There are cases where vector graphics image cannot be generated. In those cases, a PNG image is generated but the output file format and extension are not changed. An example case is a surface plot. Further details can be found in SAS user guide.

CONVERT CURRENT PIXEL-BASED GRAPHICS TO VECTOR GRAPHICS IN SAS

If traditional SAS/GRAPH procedures like PROC GPLOT/GCHART are used to generate pixel graphs, device value needs to be reset to SGMOFML and output file name format needs to change accordingly.

The following examples aim to illustrate ways to use GPLOT to create different graphic formats. The code has been reduced, in many cases, to the basic commands and is not necessarily executable in this form.

Example code for PROC GPLOT pixel-based output:

```
ods rtf style=rtftnr10 file=outputfile.rtf;
          goptions reset=all
          cback=white
          gunit=pct
          device=png target=png
          xmax=9in xpixels=3000
          ymax=4.5in ypixels=1350
          ftext=”Arial”
          htext=9pt
          ctext=black
          colors=(black red orange green blue purple);
```

Example code for PROC GPLOT vector graph output:

```
filename outfile outputfile.cgm;
          goptions reset=all
          cback=white
          gunit=pct
          device=cgmofml
          xmax=9in
          ymax=4.5in
          ftext=”Arial”
          htext=9pt
          ctext=black
          colors=(black red orange green blue purple)
          gsfname=outfile
          gsfmode=replace;
```

CONCLUSION AND DISCUSSION

For presentation purpose, SVG and Pixel-based graphics each have certain advantages. Depending on the user's requirement both formats can serve the purpose. Compared to other formats, SAS GTL can be used to generate SVG output effortlessly. The pixel-based graphs can be converted to SVG in SAS even though the original graph was generated us SAS GRAPH.

REFERENCES

- https://en.wikipedia.org/wiki/Scalable_Vector_Graphics
- SAS® Documentation
- SAS Scalable Vector Graphics in SAS® 9.2, PhUSE 2011, Nicola Tambascia, Accovion GmbH, Eschborn, Germany and Sven Greiner, Accovion GmbH, Eschborn, Germany

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Yang Wang
Enterprise: Seattle Genetics, Inc.
Address: 21823 30th Drive Southeast
City, State ZIP: Bothell, WA 98021
E-mail: yawang@seagen.com

Name: Vinodita Bongarala
Enterprise: Seattle Genetics, Inc.
Address: 21823 30th Drive Southeast
City, State ZIP: Bothell, WA 98021
E-mail: vbongarala@seagen.coml