SAS® Software and the WWW - What Next?

Larry Hoyle, Institute for Public Policy and Business Research
University of Kansas, Lawrence, Kansas

ABSTRACT

A brief overview of World Wide Web and the use of SAS as an information server over World Wide Web will be followed by a discussion of recent developments which allow for interactive information visualization over the Web. This will include a discussion of VRML and the JAVA language. The discussion will be illustrated by live examples of Web services implemented in SAS**.**

INTRODUCTION

The World Wide Web (WWW) is distributed hypermedia. WWW documents can be text, graphics, sounds, or executable programs. A WWW client (or browser) communicates with a WWW server over the internet. Most WWW documents are written in Hypertext Markup Language (HTML). WWW servers have facilities for launching and passing parameters to applications like SAS.

SAS can be used to generate static documents or graphics to be distributed by standard WWW servers. SAS can also generate and pass back WWW documents or graphics on the fly. For further details on using SAS dynamically with the WWW see: Hoyle - *Connecting SAS to the World Wide Web - Forms Across the Internet.*

The first section of this paper will show the Institute's Kansas Economic Outlook WWW pages. These are an example of a text document with links to HTML tables which in turn have links to graphic image files containing plots of the data in the tables. These pages are a static collection of files. The tables and graphics files are all generated by a SAS program which is run once a quarter.

The remainder of the paper will show recent additions to the capabilities of the WWW which allow for 3D visualization of data. Viewing the data requires the appropriate browser software which is available for a number of platforms.

The second section will show a Virtual Reality Modeling Language (VRML) plot of the Fisher Iris data set. A VRML file describes a 3D scene. This paper will use VRML as a tool for visualizing 4D data. This file was written by a SAS program included in the paper.

The final section will show the same data as displayed by a Java applet. Java is a C++ based language designed by SUN for developing WWW applications. The latest versions of Netscape are capable of running Java applets. Microsoft has also announced that it will support Java.

Where on the WWW

Kansas Economic Outlook:
http://www.ukans.edu/cwis/units/IPPBR/keo/text.htm

Fisher Iris data in VRML:
http://www.ukans.edu/cwis/units/IPPBR/vrml/irisvrml.html

Fisher Iris Data in Java Applet:
http://www.ukans.edu/cwis/units/IPPBR/java/iris/irisglyph.html

An applet is a Java program pre-complied for use by a Java aware WWW client. When the client requests it, the applet is transferred across the internet and then executed on the client machine. For this example a SAS program wrote the data input for the Java applet.

Both VRML and Java allow a user to manipulate 3 dimensional data on a 2 dimensional display. Both achieve the illusion of depth through movement and cues such as shading and size. These capabilities are built into VRML but must be programmed into a Java applet.

TABLES and GRAPHS

Kansas Economic Outlook is published quarterly by the Institute.for Public Policy and Business Research. It presents a forecast generated by the Kansas Econometric Model which is implemented in SAS/ETS®. Once the forecast is generated, a SAS program writes tables of historical and forecasted data out into Excel spreadsheets and HTML files. The program also generates a graphics file in GIF format for each line of each table.

The tables are written with HTML codes which link the row headings to the graphics files. Figure 1 shows one of the tables as it appears in Netscape. The SAS macro and call which wrote the table are listed in Appendix A.

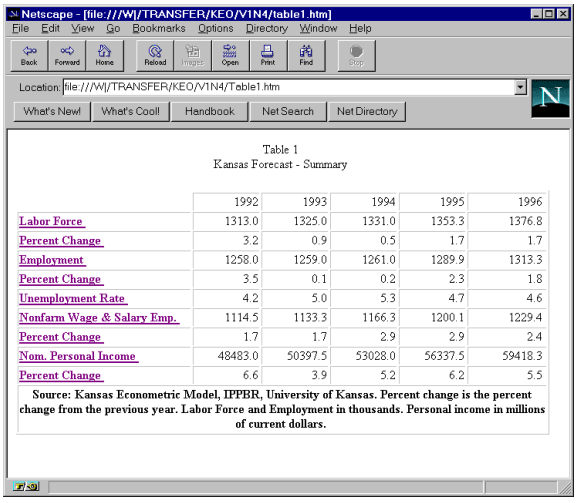

Figure 1 - An HTML3 table produced with SAS

Notice that the labels for the rows of the table are underlined. They also appear on screen in color. These attributes identify them as hyperlinks. When you click on Labor Force , your WWW browser will fetch the file pointed to by the link. Figure 2 shows that graph.

The graph in figure 2 was produced by the SAS macro and call listed in Appendix B.

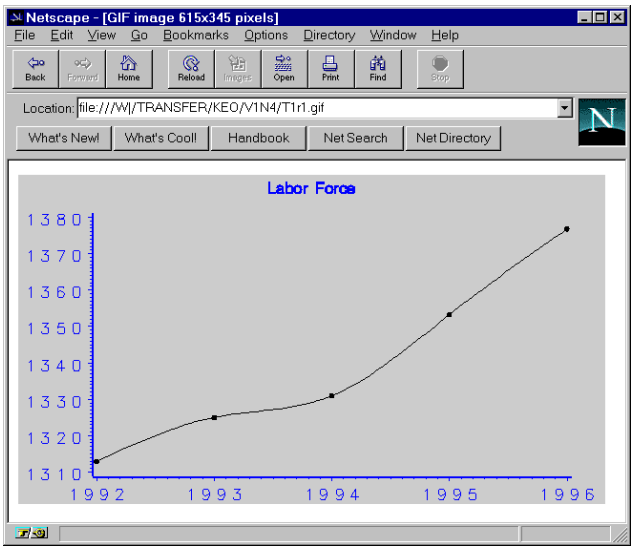

Figure 2 - A GIF graphic from device IMGGIF

```
# cone 1
   Separator {
   Material {
      diffuseColor 1 0 0
   }
   Transform {
      translation 14 50 33
      rotation 0 0 1 0.0981747704
   }
   Cone {
      bottomRadius 0.6
      height 3
   }
   } # separator for cone
# end cone 1
```

Table 1a One cone in VRML

VIRTUAL REALITY MODELING LANGUAGE (VRML)

Virtual Reality Modeling Language is a language for describing objects in a 3 dimensional scene. Objects can be translated and rotated. Lighting and camera positions can also be described. Both Microsoft's Internet Explorer and Netscape have the capability of viewing VRML within the browser.

Table 1a shows a snippet of VRML code which places a cone at the (x,y,z) coordinates (14,50,33), It also rotates that cone about the z axis by 0.098 radians. The circle at the base of the cone has a radius of 0.6 and the height of the cone is 3. The description of the cone is surrounded by a Separator command which serves to restrict the rotation and translation to just the one cone. The VRML file which produced Figure 3 has a similar section of code for each cone.

Table 1b shows the VRML code which sets the lighting and camera position for figure 3. The camera position describes the view into the scene when the file is opened.

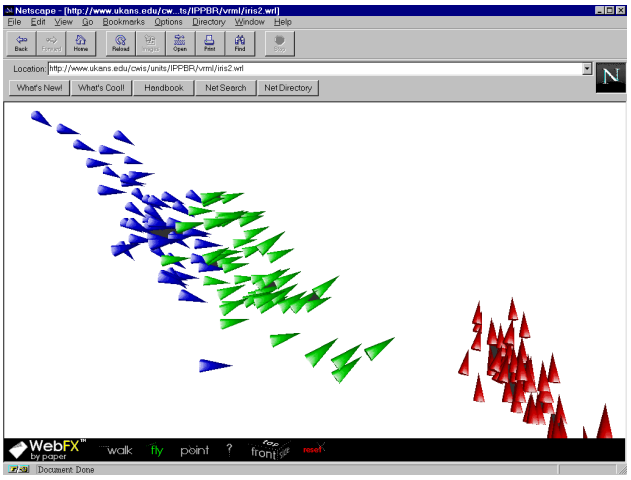

Figure 3 - A VRML glyph plot

You can change that view with browser controls when viewing the file.

The browser automatically interprets the document as VRML and displays it graphically when the server precedes it with the MIME type for VRML which is: *x-world/x-vrml.*

This code was written by the SAS program in Appendix C. The whole VRML file produces the image in figure 3 when viewed by a VRML capable browser. When viewing the scene you can fly through it, or rotate it. VRML can also describe objects as links to other WWW files.

```
#VRML V1.0 ascii
Separator {
   DirectionalLight {
      direction 0 0 1 # Light from viewer into scene
   }
   PerspectiveCamera {
      position 29.5 18 -60
      orientation 0 0.95 -0.15 3.14
      focalDistance 24
      heightAngle 0.755
   }
```

Table 1b Setting up the lights and camera in VRML

In figure 3 the Fisher iris data are displayed in a form of glyph plot[1] with each observation displayed as a cone.

The angular orientation of each cone is used to display a variable (petal width). The other three variables (petal length, sepal length, and sepal width) are displayed spatially. Each species of iris is displayed with a different color. The species Setosa displays in red and is the cluster to the right in figure 3. The species Versicolor displays in green and Virginica in blue (the blue shows darker in this paper). These species form the large cluster on the left in figure 3. The Versicolor irises, which fall toward the center of the figure, mostly show an upward tilt and the Virginica a level or downward tilt.

[1] For a discussion of glyph plots see Friendly, page 395.

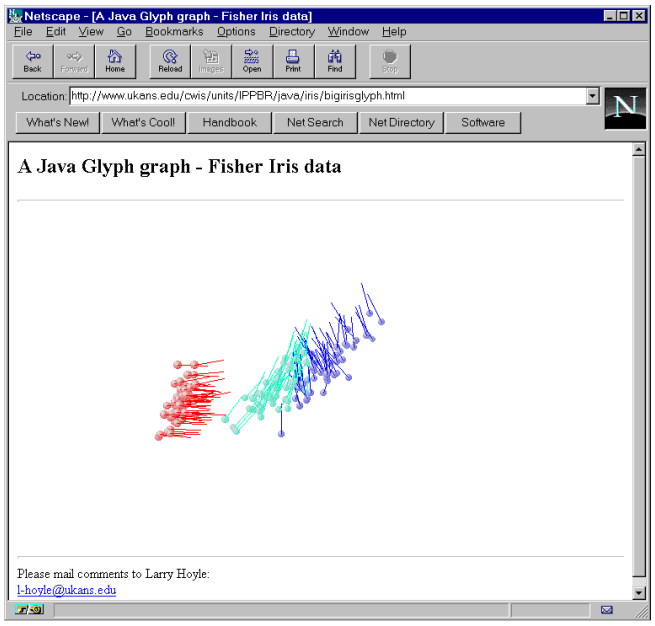

Figure 4 - A glyph plot from a Java applet

JAVA

While Java currently has no native 3D capability it can be used for 3D visualization**.** Java is a language, developed by Sun, based on C++ with some of the more dangerous features removed. It is an object oriented language designed for writing cross platform, internet aware applications. A particular kind of Java program, called an applet, is designed to be run by a WWW browser. Java applets can retrieve data and other applets from across the internet. This allows Java applets to dynamically extend the capabilities of your browser.

Even if your browser doesn't know how to display an interactive glyph plot, it can access a WWW page which includes a reference to a Java applet that does so.

Figure 4 shows the Fisher Iris data in a 3D glyph plot displayed by a Java applet. As in the preceding example the angle of the glyph represents one dimension and the other three are represented spatially. In this case the glyph is a whisker radiating from a small sphere.

As with the VRML browser this Java applet allows you to rotate the data as if they are embedded in a transparent trackball. The applet enhances the perception of depth by shrinking and graying objects as they move to the back.

This Java applet is custom designed to display a glyph plot. A SAS program writes a flat file for the applet. Table 2 shows a portion of the flat file for this example. Appendix D shows the SAS code which wrote the file for the Java applet. In this case the file is static, but a SAS program could write a file "on-the-fly" and generate the HTML to point the Java applet at it.

The Java code for the glyph applet is too lengthy for this paper but can be found at:
http://www.ukans.edu/cwis/units/IPPBR/java/iris/irisglyph.html

```
r 14 50 33 0.0981747704
```

Table 2 One record from the model file iris.gly

It was adapted from the XYZApp.java sample applet included with the Java Developer's Kit.

A Java applet is invoked by a special HTML tag: <applet>. The applet tag has a required parameter which points to the location of the applet. It may have additional parameters defined by the applet itself.

```
<html>
<head>
<title>A Java Glyph graph - Fisher Iris data</title>
</head>
<body>
<H2>A Java Glyph graph - Fisher Iris data</H2>
<hr>
<applet code=XYZAppGlyph.class width=600 height=400>
<param name=model value=models/iris.gly>
</applet>
<hr>
Please mail comments to Larry Hoyle:
<br>
<a href="mailto:l-hoyle@ukans.edu">l-hoyle@ukans.edu</a>
```

TABLE 3 - HTML for Java Glyph Graph

Table 3 shows the HTML file for Figure 4.. The applet tag also contains optional parameters which set the width and height of the display window for the applet. The tag in table 3 sets the window to 600 pixels wide and 400 pixels high.

The XYZAppGlyph applet has one additional parameter named "model" The model parameter points to the data file which contains the parameters for each iris

Writing Java - a Simple Graph Applet

The applet in Table 4a, b, & c was constructed to show complete code for an interactive graphical applet. The data it displays are wired in via a function, but the applet could easily be modified to input data and other parameters from a SAS program.

Table 4a contains introductory comments and statements linking in code from the Java Abstract Window Toolkit (AWT) which is already available in the browser .
Java is an object oriented language. Individual objects are described in terms of the class to which they belong. The SimpleGraph applet defines two classes, SimpleGraph and OvalPoint. An instance of the SimpleGraph class (a

```
/* SimpleGraph.java                                          */
/*    Larry Hoyle, University of Kansas, December 1995
             */
/*        SimpleGraph displays an xy plot of a function      */
/*        computed over each x value in the applet window.   */
/*        Clicking the mouse window will highlight the point */
/*         corresponding to the x value of the click.        */

import java.awt.Graphics;
import java.awt.Event;
import java.awt.Color;
```

Table 4a SimpleGraph.java part 1

SimpleGraph object) is created when an HTML file references it with the <applet> tag.

The SimpleGraph applet in turn creates one instance of an OvalPoint for each point it will display in the applet window. It creates one point for each column of pixels in the window.

The first section of table 4b defines variables, an array, and a function. The init method (subroutine) sets up an array of OvalPoints, one for each x value in the graph. The function f is used to compute the y value for each point. Each point also has a width, a height, and an attribute of picked (true or false) as defined in the first part of 4c.

Each point also knows how to paint itself on the screen as defined in the paint method in table 4c. When picked, a point displays as a red filled oval with its coordinates listed to the right. When not picked the oval is black and not filled. The point's width and height attributes determine the size of the oval.

Picking and unpicking a point is accomplished by the MouseDown method shown in Table 4b. The MouseDown method is called by the browser whenever the mouse button goes down. SimpleGraph's MouseDown method looks at the x coordinate of the cursor where the click occurred and flips the picked attribute of the OvalPoint with that coordinate. It also invokes the applet's repaint method.

The SimpleGraph applet also has a paint method which is invoked each time the browser repaints the applet. This paint method loops through the array of OvalPoints and calls each one's paint method.

```
public class SimpleGraph extends java.applet.Applet {
   OvalPoint OvalPoints[];
   int owide=6;
   int ohigh=6;
   int startwidth, startheight;
   double f(double x) {
            return (Math.sin(x/20) + 1) * size().height / 2;
   }
                        // the array of points has startwidth elements
                        // even if the applet is resized
   public void init() {
      startwidth = (int) size().width;
      startheight = (int) size().height;
      OvalPoints = new OvalPoint[startwidth];
      for (int x = 0 ; x < startwidth ; x++) {
         OvalPoints[x] = new OvalPoint(x, (int) f(x), owide, ohigh);
      }

   }                    // mouseDown toggles a point's picked attribute

   public boolean mouseDown(Event e, int mx, int my){
      OvalPoints[mx].picked = ! OvalPoints[mx].picked;
      repaint();
      return true;
   }
                        // the applet's paint invokes each point's paint
   public void paint(Graphics g) {
     for (int x = 0 ; x < startwidth ; x++) {
         OvalPoints[x].paint(g) ;
      }
   }
}                       // end SimpleGraph
```

Table 4b SimpleGraph.java part 2 The SimpleGraph class

```
                        // each point has a location (x,y)
                        // a width, a height, and a picked attribute
class OvalPoint{
   public int x, y, width, height;
   public boolean picked=false;

   OvalPoint(int xvar, int yvar, int w, int h){
      x=xvar;
      y=yvar;
      width=w;
      height=h;
   }
                        // when picked, the point is labeled and filled
   public void paint(Graphics g) {
     if (picked){
        g.setColor(Color.red);
        g.drawString("x=" + x + ", y=" + y,x+width,y+width);
        g.fillOval(x, y, width, height);
     }
     else{
        g.setColor(Color.black);
        g.drawOval(x, y, width, height);
     }
   }
}                       // end OvalPoint
```

Table 4c SimpleGraph.java part 3 - The OvalPoint class

SimpleGraph's MouseDown method can access the picked attribute of each OvalPoint directly since picked is declared as public. The variables x, y, width, height, and picked of class OvalPoint are *instance* variables, that is there is a separate copy of them for each instance of an OvalPoint. These instances are created with the *new* operator in table 4b. When a new instance is created OvalPoint's *constructor* is invoked. This is the section of table 4c which begins: "OvalPoint(int xvar, int yvar, int w, int h){". This constructor initializes x, y, width, and height.

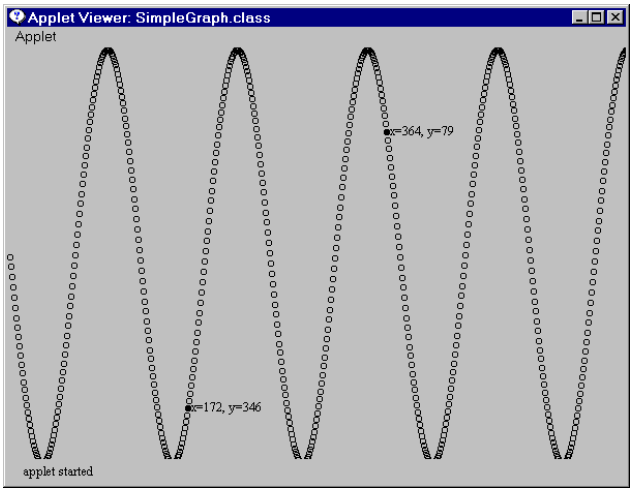

Figure 5 - The display from applet SimpleGraph

While trivial itself, the SimpleGraph applet shows the worth of Java for publishing interactive statistical graphics. Information about the graphic can be available with a mouse click without cluttering up the view. The viewer applet could also zoom and pan the graphic.

CONCLUSION

This paper has been concerned with two main themes. First - the World Wide Web has enabled wonderful new tools for distribution of information visualization. Three dimensional and interactive viewers which are standard across multiple hardware platforms and operating systems will bring publishing to a new level.

The second issue has to do with what the new technology means to the SASsystem. Java, or whatever evolves like it, brings client server computing and a common graphical user interface to everyone with an inexpensive browser.

The Pure Speculation Part

SAS Institute has staff with the expertise to create a suite of Java applets tightly integrated with the internals of the SAS system. The possibilities are tantalizing. It would be possible, for example, to have output from SAS/INSIGHT® to a 3D rotation applet to show an interactive display of a particular 3D view of some multidimensional data set. SAS/SPECTRAVIEW® could output to an applet which would allow moving a cutting plane through a data subset. SAS/GIS® could output to a map applet which would allow popup information about a geographic unit.

While we wait hopefully for such tools, the current SAS system has the flexibility to do a great deal with user designed applets and with VRML files.

REFERENCES

Bell, G., Parisi A., Pesce, M.,. *The Virtual Reality Modeling Language,Version 1.0 Specification*, available at: http://vrml.wired.com/vrml.tech/vrml10-3.html

Clifford, Norman, Kansas Economic Outlook, Volume 1 Number 4, Lawrence, KS: Institute for Public Policy and Business Research, University of Kansas. and available at: http://www.ukans.edu/cwis/units/IPPBR/keo/text.htm

Hoyle, Larry, *Connecting SAS to the World Wide Web - Forms Across the Internet* presented at MWSUG94 and available at:
ww.ukans.edu/cwis/units/IPPBR/sashttp/mwsug94/mwsug94.html

SAS Institute Inc., *SAS Institute*, available at: http://www.sas.com/

SAS Institute Inc. (Friendly, Michael), *SAS System for Statistical Graphics, First Edition,* Cary, NC: SAS Institute Inc. 1991. 697pp.

Sun Microsystems, Inc., *Java: Programming for the Internet*, available at: http://java.sun.com/

APPENDIX A - HTML Table Macro

Figure 1 was produced by this call

```
%HTMLTAB(Y,step17, htable1, t1labs, ncols=5,
 tnum=1,
 fmt=7.1,
 t1=%str(Table 1),
 t2=%str(Kansas Forecast - Summary),
 f1=%str(Source: Kansas Econometric Model, IPPBR, University of
Kansas.),
 f2=%str(Percent change is the percent change from the previous year.),
 f3=%str(Labor Force and Employment in thousands.),
 f4=%str(Personal income in millions of current dollars.));
```

The macro definition:

```
%macro  HTMLTAB(datfmt,
          dfile,
           tfile,
           lfile,
           tnum=,
           ncols=1,
           fmt=,
           t1=,
           t2=,
           t3=,
           f1=,
           f2=,
           f3=,
           f4=);

           /*  write a table to an HTML file                */
           /*  datfmt - The format of the date
              Y=year Q=quarter                               */
           /*  dfile - the data file from proc transpose    */
           /*  tfile - the output table filename            */
           /*  tnum  - table number                         */
           /*  ncols - number of columns                    */
           /*  lfile - the file with row labels             */
           /*  fmt   - format for numeric cells             */
           /*  t1    - first title line                     */
           /*  t2    - second title line                    */
           /*  t3    - third title line                     */
           /*  f_    - footer lines                         */
           /* NOTE: dfile and lfile must have the same length */

           /* HTML 3 Table codes:
           <table border> ....</table>
                    wraps the whole table
           <caption> ... </caption>
                    wraps the caption
           <tr>
                    starts a table row
           <th>
                    starts a table header
           <td>
                    starts a table cell
           <a href="URL>link text</a>
                    when link text is clicked retrieve URL
           <br>
                    line break
           */
```

```
Data _NULL_;
     merge  &lfile &dfile end=theend;
     file &tfile lrecl = 1000;
     rownum = _n_-1;
     m=-1;
     if _n_=1 then
      do;


          /* first record - define table, print headers */


        put '<table border>';
        put '<caption>';
        put "&t1 <br>";
        put "&t2 <br>";
        put "&t3 <br>";
        put '</caption>';
        put '<tr>';
        put '<th>' rlabel $40.
          %DO ixcol=1 %TO &ncols;
            %IF &datfmt=Y %THEN '<td align=right>'
                  col&ixcol YEAR4. ;
            %ELSE
             %DO;
               %IF &datfmt=Q %THEN '<td align=right>'
                     col&ixcol YEAR2. '/'  col&ixcol QTR1. ;
               %ELSE '<td align=right>' col&ixcol ;
             %END;
          %END;    /* ixcol=1 */
       ;
      end; /* if _n_ */

                     /* row labels are anchors */

     else Put '<tr> <th align=left>'
            "<a HREF=""T&tnum.R"  rownum +m '.gif">'
                        rlabel $40. '</a>'
                     /* data cells */

        %DO ixcol=1 %TO &ncols;
          '<td align=right>' col&ixcol &fmt
        %END;
       ;
                     /* footnotes */

     if theend then
      do;
        totcols=&ncols+1;
        %IF &f1 ne %THEN put "<tr> <th colspan="
                              totcols "> &f1";;
        %IF &f2 ne %THEN put "&f2";;
        %IF &f3 ne %THEN put "&f3";;
        %IF &f4 ne %THEN put "&f4";;
        put '</table>';

      end; /* if theend */
%mend HTMLTAB;
```

APPENDIX B - Drawing a GIF Graphic

Figure 2 was produced by this call

```
%rowplot(f=T1R1,
 y=KCLFa,
 i=spline,
 ylab=,
 t=Labor Force,
 x=year, d=step16 );
```

The macro definition:

```
%macro rowplot(t=,
               i=,
               x=,
               xlab=,
               y=,
               ylab=,
               d=,
               f=);

/*             t=  title
*              i=  interpolation (spline or needle)
*              x=   x variable name (date or year)
*              xlab= x axis label
*              y=   y variable name
*              ylab= y axis label
*              d=  data set name
*              f=  output file name
*/
filename grout "w:\transfer\keo\&issue\&f..gif";


/* IMGGIF is the driver for GIF files */

goptions reset=(axis, legend, pattern, symbol, title,
footnote) norotate
        hpos=0 vpos=0 htext=2 cback=white ;
goptions device=IMGGIF gsfmode=replace
gsfname=grout  ctext=blue
        graphrc interpol=join;

 title1
  "&t";
symbol1 c=DEFAULT
   i=&i
   l=1
   v=DOT
   c=black
   ;
axis1
   label=(h=3 "&xlab")
   color=blue
   width=2.0
   ;
axis2
   label=(h=3 "&ylab")
   color=blue
   width=2.0
   ;
axis3
   color=blue
```

```
  width=2.0
  ;

proc gplot data=&d ;
  plot &y * &x /
    haxis=axis1
    vaxis=axis2
    ;
run;
%mend rowplot;
```

APPENDIX C - Writing VRML

```
/*irisvrml.sas  - make a vrml file from SAS x y z data  */

/* Larry Hoyle, IPPBR, University of Kansas            */
/* l-hoyle@ukans.edu                                   */

/* use the Fischer Iris data from candiex.sas
          */
/* use the angle computation from Friendly, M.         */
/* SAS System for Statistical Graphics                 */

filename irvrml 'c:\ddrive\sugi21\vrml\iris.wrl';
  file irvrml;

proc format;
  value specname
    1='SETOSA    '
    2='VERSICOLOR'
    3='VIRGINICA ';
  value specchar
    1='S'
    2='O'
    3='V';
run;

data iris;
  title 'Fisher (1936) Iris Data';
  input sepallen sepalwid petallen petalwid species @@;
  format species specname.;
  label sepallen='Sepal Length in mm.'
        sepalwid='Sepal Width  in mm.'
        petallen='Petal Length in mm.'
        petalwid='Petal Width  in mm.';

/*  The data for this example can be found in */
/* the program candiex.sas from the SAS sample */
/* library */

                    /* find the bounding box for the data */'

proc means data=iris min max;
  var petalwid petallen sepalwid sepallen ;
  output out=range min=pwmin plmin swmin slmin
                max=pwmax plmax swmax slmax;

                    /* compute the glyph angle */

data xyz;
  set iris;
  if _n_=1 then set range;

  p1 = (petalwid-pwmin) / (pwmax - pwmin);


  x=petallen;
  y=sepallen;
  z=sepalwid;
  angle = 135 * p1 * arcos(-1)/180;
  xg  = x + 6 * cos(angle);
  yg = y + 6 * sin(angle);

                    /* write the VRML */

data _null_;
set xyz nobs=n end=last;

  if _n_=1 then do;
    put '#VRML V1.0 ascii';
    put 'Separator {';

    put '    DirectionalLight {';
    put '        direction 0 0 1 # Light from viewer into
scene';
    put '    }';

    put '    PerspectiveCamera {';
    camx = (plmax - plmin) / 2;
    camy = (slmax - slmin) / 2;
    camz = ((swmax - swmin) / 2)  - (3 * (swmax - swmin))  ;
    fd = swmax - swmin;
    put '        position 'camx camy camz;
    put '        orientation 0 0.95 -0.15 3.14';
    put '        focalDistance ' fd;
    put '        heightAngle 0.755';
    put '    }';
   end; /* _n_=1 */


                    /* output the point as a cone */

    put '# cone '_n_;
    put '    Separator {';
    put '    Material {';
    put '        diffuseColor '@;
    select (species);
      when (1) put '1 0 0';
      when (2) put '0 1 0';
      when (3) put '0 0 1';
      otherwise put '0.5 0.5 0.5';
    end; /* select */

    put '    }';

   put '    Transform {';
    put '        translation ' x y z;
    put '        rotation 0 0 1 ' angle;
    put '    }';
```

```
    put '    Cone {';
    put '        bottomRadius 0.6';
    put '        height 3';
    put '    }';
    put '    } # separator for cone';
    put '# end cone '_n_;
    put '# ';

  if last then do;
    put '}';

  end; /* do; */
run;
```

APPENDIX D - Writing a File for XYZAppGlyph

```
/*irisjava.sas  - make a data file from SAS x y z data  */
/* the file will be used by XYZAppGlyph.java             */

/* Larry Hoyle, IPPBR, University of Kansas, Nov. 1995 */

/* Uses  the Fischer Iris data from the SAS Institute
     sample program candiex.sas                         */
/* angle computation from Friendly, M.                  */
/* SAS System for Statistical Graphics                  */

filename irjava
'c:\apps\java\demo\XYZGlyph\models\iris.gly';
proc format;
   value specname
      1='SETOSA    '
      2='VERSICOLOR'
      3='VIRGINICA ';
   value specchar
      1='S'
      2='O'
      3='V';
run;

proc means data=iris min max;
  var petalwid petallen sepalwid sepallen ;
  output out=range min=pwmin plmin swmin slmin
                   max=pwmax plmax swmax slmax;

data xyz;
  set iris;
  if _n_=1 then set range;

  p1 = (petalwid-pwmin) / (pwmax - pwmin);


  x=petallen;
  y=sepallen;
  z=sepalwid;
  angle = 135 * p1 * arcos(-1)/180;
  xg  = x + 6 * cos(angle);
  yg = y + 6 * sin(angle);

data _null_;
file irjava;
  set xyz nobs=n end=last;

  select (species);
     when (1) put 'r ' x y z angle;
     when (2) put 'g ' x y z angle;
     when (3) put 'b ' x y z angle;

     otherwise ;
  end; /* select */
run;
```


ACKNOWLEDGEMENTS

SAS, SAS/ETS, SAS/GIS, SAS/GRAPH, SAS/INSIGHT, SAS/SPECTRAVIEW are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates a USA registration.

Netscape Communications, Netscape, Netscape Navigator and the Netscape Communications logo are trademarks of Netscape Communications Corporation.

Sun, and Java are trademarks or registered trademarks of Sun Microsystems, Inc.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Larry Hoyle
IPPBR, University of Kansas
607 Blake Hall
Lawrence, KS, 66045-2960
(913) 864-3701
l-hoyle@ukans.edu
http://www.ukans.edu/cwis/units/IPPBR/IPPBR_main.html