Paper 162-29

New Palettes for SAS 9 Color Utility Macros

Perry Watts, Independent Consultant, Elkins Park, PA

Abstract

Version 9 SAS replaces the list of predefined SAS colors with color utility macros that work with a full range of code-to-color mappings in a given color space. Start values for RGB (red | green | blue) codes are three decimals representing percents. They are converted to a hexadecimal value SAS understands. On the other hand, start values for HLS (hue | lightness | saturation) are a combination of degrees and percents. Hue ranges from 0 to 360 degrees whereas lightness and saturation are listed as decimal percents. Again the output from the utility macro is a single hexadecimal value for a given color. Here is how to get CYAN from two of the utility macros:

```
%RGB(0,100,100)  = CX00FFFF
%HLS(300,50,100) = H12C80FF
```

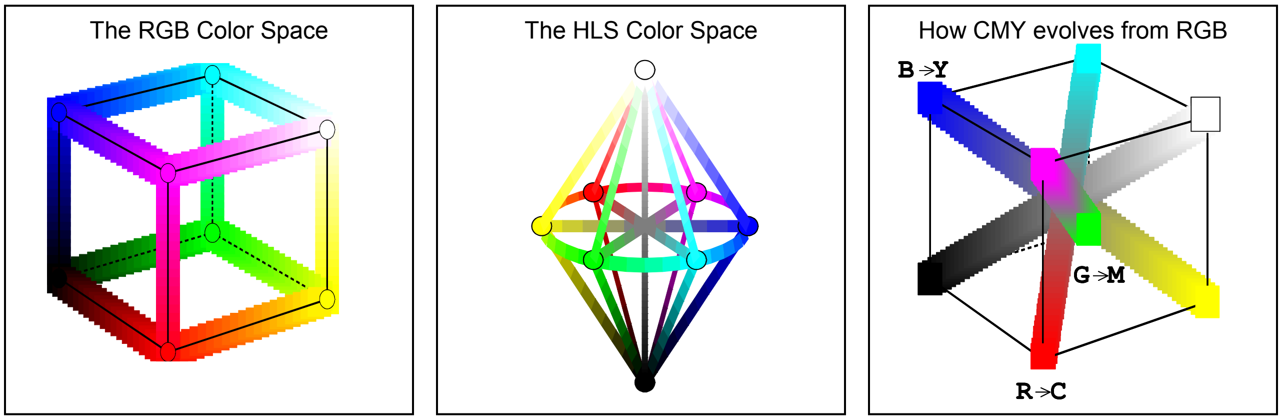
Although decimals are easier to understand than hexadecimals, it is still difficult to know what combination of percents and numbers will yield a particular color. Therefore palettes are needed as color-selection tools. Instead of using ODS with PROC REPORT output to an HTML file, the palettes in this paper are constructed by generating HTML script directly in a data step. They are designed to display uniform distributions of colors at any fixed interval. Percentage-based color codes can be imputed from palette margins and by counting color bands in the cell GIF files. Even though V8.2 SAS is used for programming, the palettes reflect the V9 macro structure in composition. Source code and graphics displays are included in the paper.

Introduction

While the RGB and HLS spaces in Figure 1 show how a gamut of codes is mapped graphically, the percentage-based V9 macros appear from the documentation to permit only an integer sampling of each color space. More specifically, while the three components in RGB space vary from 0 to 255, only coarser integer percents are shown as parameters in the *RGB* color utility macro in SAS *OnlineDoc V9*. Without access to Version 9 software, however, it is impossible to know how the percentage-based macros actually work and how they affect final color renderings.

The V9 color utility macros also operate on additional color systems with spatial dimensions described by Foley and Van Dam (611-623). HSV for Hue|Saturation|Value is a single hex-cone that corresponds to the bottom half of the HLS double-ended cone rotated 120 degrees. Red replaces blue at the origin in this system. Cyan, magenta and yellow shown as complements to red, green and blue in Figure 1 form the CMY system used in printing. CMY space is also a cube, but white, instead of black, is located at the origin. Since CMY**K** adds blac(k) as a fourth dimension to CMY the system cannot be shown in a single graph. Finally, descriptive labels used as parameters in CNS utility macros are derived from the HLS system.

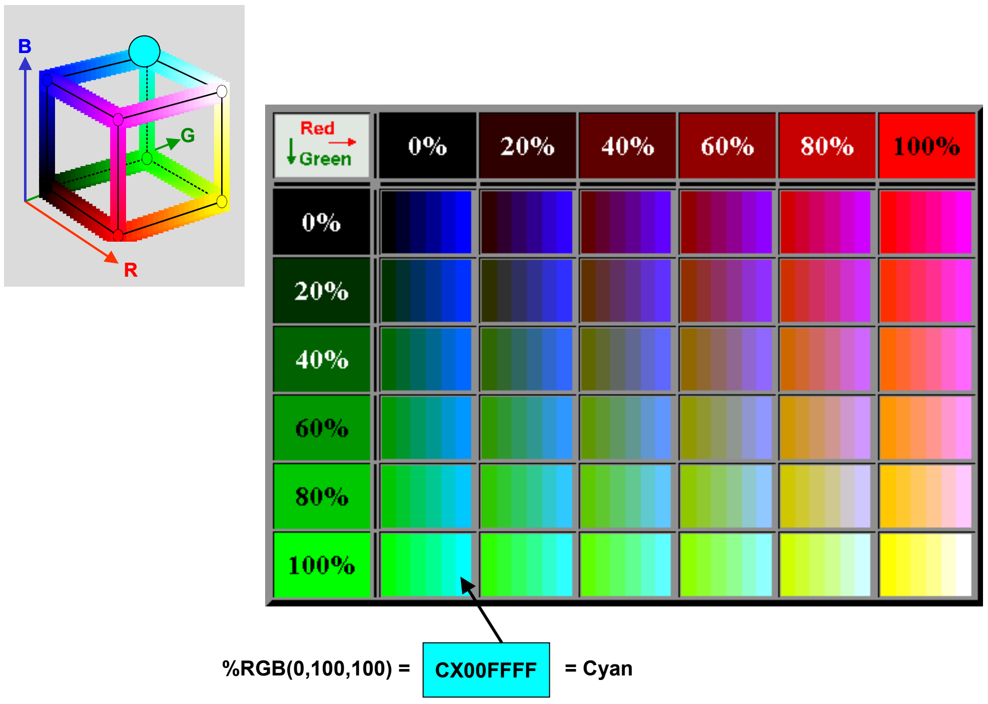
Figure 1. RGB and HLS color spaces are shown in the first two panels. RGB complement colors shown in the third panel define the CMY system. White, also a complement of black, becomes the *origin* in CMY.



RGB: The Color System for the Web

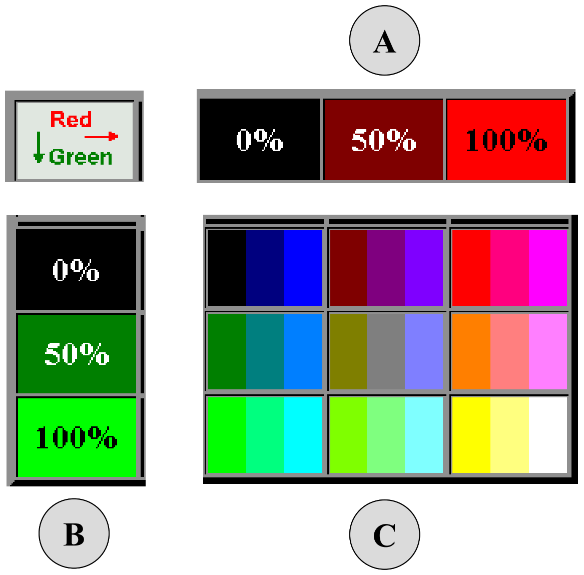
Even though the predefined list of SAS colors from earlier releases of SAS software has been discarded in favor of the color utility macros, a chart or palette is still needed as an aid for selecting colors in V9 SAS. The chart displayed in Figure 2 shows bands of colors uniformly spaced apart in RGB space. *Red* and *green* components are fixed in the margins and *blue* varies within the cell GIF files. With 20% as the fixed interval, the important Browser-safe palette of 216 or 6^3 uniformly distributed colors in RGB space is replicated in this example.

Figure 2. The Browser-Safe "Picker" palette reflects the percentage-based V9 Utility macro structure. Cyan is identified in both the RGB space and in the color chart.



In Figure 3, the three regions in the RGB palette are "exploded" for greater visibility. The chart is a coarser version of the one shown in Figure 2 with intervals increasing from 20 to 50 percent. The regions are labeled with identifying letters that appear in the code summaries below. The letter **A** points to the red margin at the top, **B** is for the green margin in the first column, and **C** covers the remaining cells containing the GIF files. The upper left-hand corner label stands apart from the regions. It is created in Visio and saved as a GIF file.

Figure 3. The three chart regions are identified and labeled for reference in the code summary.



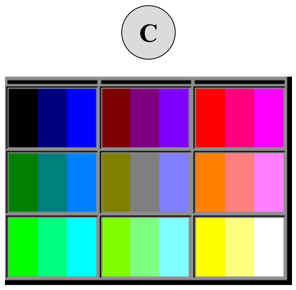
Palette Construction is a Three-Step Process

While many color options are available in the Output Delivery System (ODS), it is not used to make either palette described in this paper. Instead GIF files are created in a macro, and HTML script is generated in a data step with output to an HTML file via DATA _NULL_. To use SAS for direct output to HTML this way, it is best to work with an HTML textbook in hand. Patrick Carey's *Creating Web Pages with HTML 2nd Edition* is an excellent resource.

The same SAS program is used to create the charts in Figures 2 and 3. Only the value of PCTINCR, a macro variable, has to be changed. However, to satisfy HTML requirements, preliminary steps must also be taken to ensure that GIF and HTML files are stored in the same subdirectory.

1) Creating the GIF Files

The RGBGIFS macro is executed in V8.2 SAS. Variables R, G, and B are assigned percent values in do-loops to mimic the color utility macros, and then derivative variables RED, GREEN, and BLUE are created to store the percentages as decimal renditions of the actual codes. A final translation uses the HEX format to convert RED, GREEN and BLUE into a hexadecimal string used for color assignments in SAS.



```
%macro RGBGIFS;
 %do R=0 %to 100 %by &pctIncr;
   %let Red = %sysfunc(round(%sysevalf((&R./100)*255))); ❶
   %do G=0 %to 100 %by &pctIncr;
     %let Green = %sysfunc(round(%sysevalf((&G./100)*255)));
     %let zRed=%sysfunc(putN(&red,z3.)); ❷
     %let zGreen=%sysfunc(putN(&green,z3.));
     %let IncrID=%sysfunc(putN(&pctIncr,z3.));
     %let gifFile=R&zRed._G&zGreen._B&IncrID; ❷
     %DelCatMac(work.gseg);
       data anno;
       %system(3,3,3); *-- coordinates are in percents;
         length color $8;
         retain Red &Red  Green &Green;
         xincr=&xincr;
         x=0;
         do B=0 to 100 by &pctIncr;
           Blue = round((B/100)*255);
           xleft=x; xright=xleft + xincr; ylow=0; yhigh=100;
           substr(color,1,2)='CX';
           substr(color,3,2)=put(Red,hex2.); ❸
           substr(color,5,2)=put(Green,hex2.);
           substr(color,7,2)=put(Blue,hex2.);
           %_bar(xleft,ylow,xright,yhigh,color); ❹
           x= xright;
         end;
       run;
       filename CellGIF "&path.\&gifFile..gif"; ❺
       goptions reset=goptions;
       goptions device=imggif display rotate=landscape
         gunit=pct polygonfill fill hsize=0.75in vsize=0.5in
         gsfname=CellGIF gsfmode=replace cback=CX909090;
       proc gslide annotate=anno;
         title1;
         footnote1;
       run;
    %end; *-- green;
  %end; *-- red;
%mend RGBGIFS;
```

❶ When **R** is 20%, then **RED** is rounded to 20/100*X* 255 or 51. Hex assignments for color codes are correct if decimals are rounded beforehand.
❷ RED here is used in the GIF file name. The Z format is used to ensure that no gaps will appear in the name.
❸ The RED component is assigned to the RGB code with an application of the HEX format that translates decimals to hexadecimals. Note the use of the SUBSTR function on the left-hand side of the assignment statement.
❹ See the paper *Advanced Programming Techniques for Working with Color in SAS® Software* for a description of the user-defined _BAR macro that is related to the SAS-defined ANNOTATE BAR macro. With _BAR, colors can be assigned as variables in a data step.

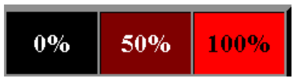
❺ This is the section of code where the GIF files are actually created. The same background color in HTML is reassigned in SAS with the CBACK graphics option (GOPTION). Otherwise thin white lines appear in the display revealing a lack of synchrony between the two software packages. For readers familiar with SAS/GRAPH software, the HTML-direct method also eliminates the need for collecting separate GIF files into a single display with PROC GREPLAY. The advantage of this simpler method for multiple color displays is shown in the Appendix.

2) Generating HTML commands in a Data Step

HTML commands are assigned individually to the STRING variable in the data set HTMLDAT. The variable INDENT is only added to increase the readability of the HTML file when it is viewed in a text editor. The code for the three regions in Figure 2 is presented below.

```
/*ROW HEADERS FOR RED*/
Indent=2;
do R=0 to 100 by &pctIncr; ❶
  Red = round((R/100)*255);
  PctLbl=put(R,3.);
  substr(bgColor,1,2)=put(Red,hex2.);
  substr(bgColor,3,4)='0000'; ❷
  if R lt 95 then fgColor='FFFFFF'; ❸
  else fgColor='000000';
   string='<TH HEIGHT=48 WIDTH=72 BGCOLOR="#'||bgColor||'"> <FONT SIZE='||fontSize||'
          COLOR = "#'||fgColor||'">'||trim(PctLbl)||'% </FONT></TH> '; ❹
  output;
end;
```

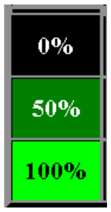
A



❶ Changing the value for PCTINCR results in charts with different percentage intervals.
❷ Only the RED component is varied in the header region. The other two components remain fixed at zero.
❸ Text containing percentage levels is labeled in black or white depending on the background color.
❹ A compound HTML statement is captured in SAS with the concatenation operator (||). Notice that HTML colors start with a pound sign (#). A line break has been added to make the source code more readable.

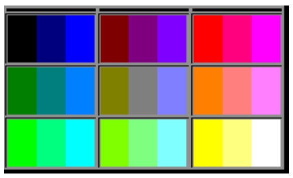
```
/* COLUMN LABELS FOR GREEN*/
do G=0 to 100 by &pctIncr;
  Green = round((G/100)*255);
  PctLbl=put(G,3.);
  substr(bgColor,1,2)='00';
  substr(bgColor,3,2)=put(Green,hex2.);
  substr(bgColor,5,2)='00'; ❶
  if G LE 50 then
    fgColor='FFFFFF'; ❷
  else
    fgColor='000000';
  string='<TD ALIGN=CENTER BGCOLOR="#'||bgcolor||'"> <FONT SIZE='||fontsize||'
 COLOR = "#'||fgColor||'"> <B>'||trim(PctLbl)||'% </B></FONT></TD>';
 output;
… ❸
```

B



❶ RED and GREEN margin processing is almost identical. This time GREEN is varied by PCTINCR intervals while RED and BLUE remain fixed at zero.
❷ Green is lighter than red, so FGCOLOR switches to black earlier. Look at how the red and green labels in Figure 2 differ from each other.
❸ An ellipses is inserted to show that additional row processing (for red below) is performed within the GREEN (G) loop.

C



```
 /*INSERT GIF FILES INTO REMAINING ROW CELLS*/
 do R=0 to 100 by &pctIncr;
  Red = round((R/100)*255);
   zRed=put(red,z3.);
   zGreen=put(green,z3.);
   IncrID =put(&pctIncr,z3.);
   gifFile='R'||zRed||'_G'||zGreen||'_B'||IncrID||'.gif'; ❶
   string='<TD> <IMG SRC="'||gifFile||'" WIDTH=72 HEIGHT=48 ></TD>';
   output;
 end; *-- RED;
end; *-- GREEN;
```

➊Only RED and GREEN margin colors are assigned directly in HTML. BLUE is taken care of in the GIF files. Here is where GIF files created in the RGBGIFS macro are inserted into HTML table cells.

3) Writing Output with Data NULL

DATA _NULL_ is used for generating the HTML file:

```
data _null_;
   filename HTMLOut "&path.\RBGChrt_&pctIncr..htm";
   file HTMLOut;
   set HTMLdat;
   put @Indent +2 string;
run;
```

HLS: A User-Oriented Color System

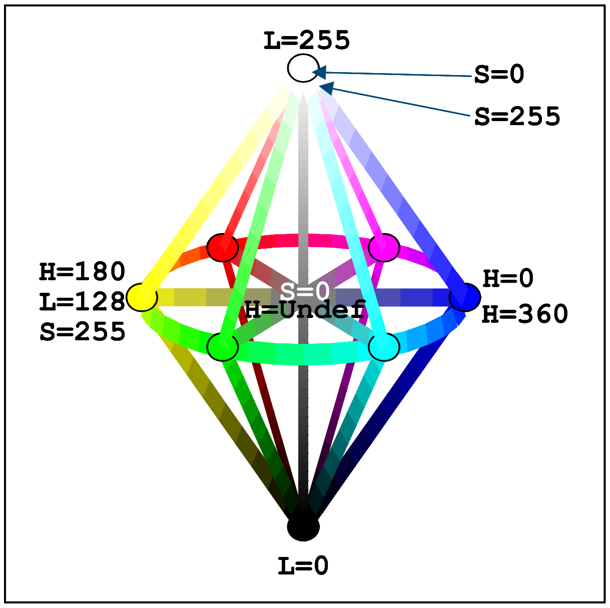
While the HLS coloring system is said to be more intuitive than the older, hardware-oriented RGB system, it presents a couple of challenges that need to be addressed before a percentage-based palette can be constructed. First, the palette must reflect the structure of the HLS space that includes several discontinuities, and secondly colors for the palette margins must be translated into RGB, the only coding system HTML recognizes.

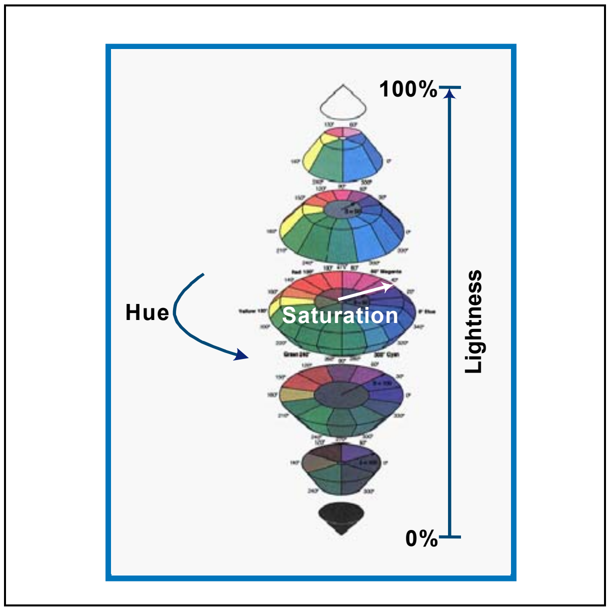
To understand the discontinuities described in Figure 4, HLS components need to be defined precisely. The following definitions come from www.hypersolutions.org/pages/colorDef.html:

HUE: the attribute of color by means of which a color is perceived to be red, yellow, green, blue, purple, etc. Pure white, black, and grays possess no hue (Range 0 to 360°).
LIGHTNESS: (1) the attribute of color perception by which a non-self-luminous body is judged to reflect more or less light. (2) the attribute by which a perceived color is judged to be equivalent to one of a series of grays ranging from black to white (Range 0 to 255).
SATURATION: the attribute of color perception that expresses the degree of departure from the gray of the same lightness. All grays have zero saturation. Commonly used as a synonym for chroma especially in graphic arts (Range 0 to 255).

Lightness and saturation may still be somewhat confusing. An alternative definition for lightness is the amount of white contained within a color whereas the amount of gray defines a color's saturation. Fully saturated colors contain no gray. Black and white are simply "shades" of gray.

Figure 4. The Tektronix HLS Color System used by SAS software maps as a double-ended cone. The diagram to the left highlights the discontinuities of the system, and the one on the right is taken from *SAS Online Doc*.



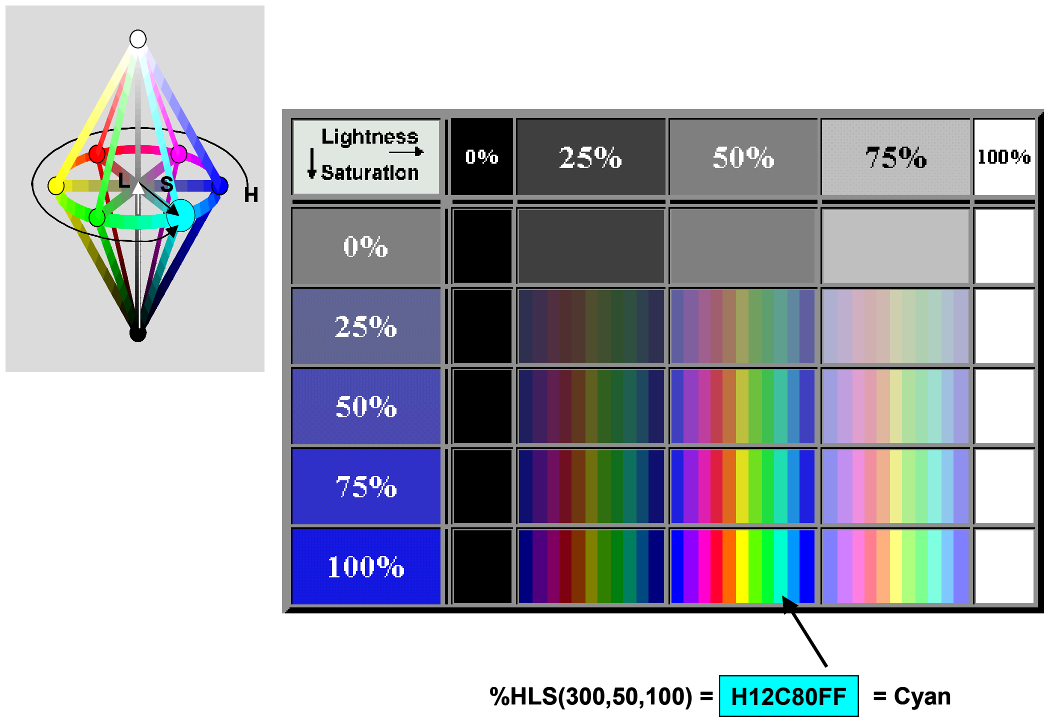


From the component definitions and the left-hand graph in Figure 4, hue is shown to be undefined when saturation drops to zero. Gray in this system only depends on the value for lightness. For example, the same shade of gray is returned when saturation is reset to zero for either blue or yellow midway up the cone. In contrast, no corresponding distinction is made between black, white, gray, and other colors in the RGB system.

Saturation also gives rise to a second anomaly in the system. Fully saturated colors are located on the surface of the cone, but black and white with saturation values of zero are also portrayed as surface colors. Presumably when the value of lightness drops one digit to 254, saturation jumps from 0 to 255.

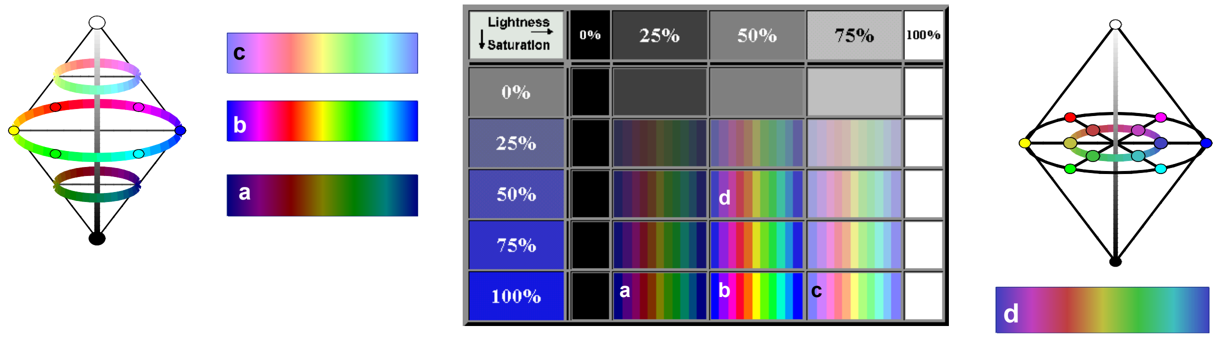
Because of the discontinuities in Figure 4, several decisions have been made about how to present the HLS palette in Figure 5. The shade of blue with hue 0 and lightness 128 (50%) has been arbitrarily selected to represent the saturation dimension, and each row contains black and white cells even though they are really only valid when saturation is zero.

Figure 5. An HLS percentage-based palette contains hue scales plotted at 25% lightness and saturation intervals. Each hue scale contains 10 colors uniformly spaced apart.



The different units of measurement among the component parameters in the HLS V9 utility macros also dictate how the palette is displayed. Since lightness and saturation are the percentage components, they must be defined as the palette margins. Hue, shown in degrees, can assume a more autonomous role as the third dimension in the GIF file inserts. Correspondingly, marginal percentages are allowed to vary between program runs whereas hue intervals remain fixed at 36 degrees. See Figure 6 for alternative displays of hue scales along with related space and palette mappings.

Figure 6. Selected GIF cells in the HLS palette are mapped to space diagrams and more refined color scales.

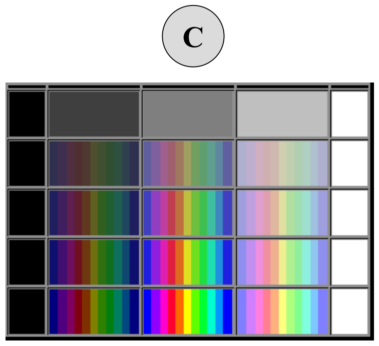


HLS Palette Construction Involves Translations for Marginals to RGB

Fortunately the IMGGIF graphics device supports user-defined colors. This means that HLS codes can be used to generate GIF files from inside SAS. This convenience is far from intuitive, and the SAS code needed for determining the capability of a device is not obvious either:

```
goptions device=IMGGIF;

proc gtestit picture=1;
run;
```



After the program fragment above is executed, the first byte in the OPTS string in the LOG output needs to be converted by hand into binary:

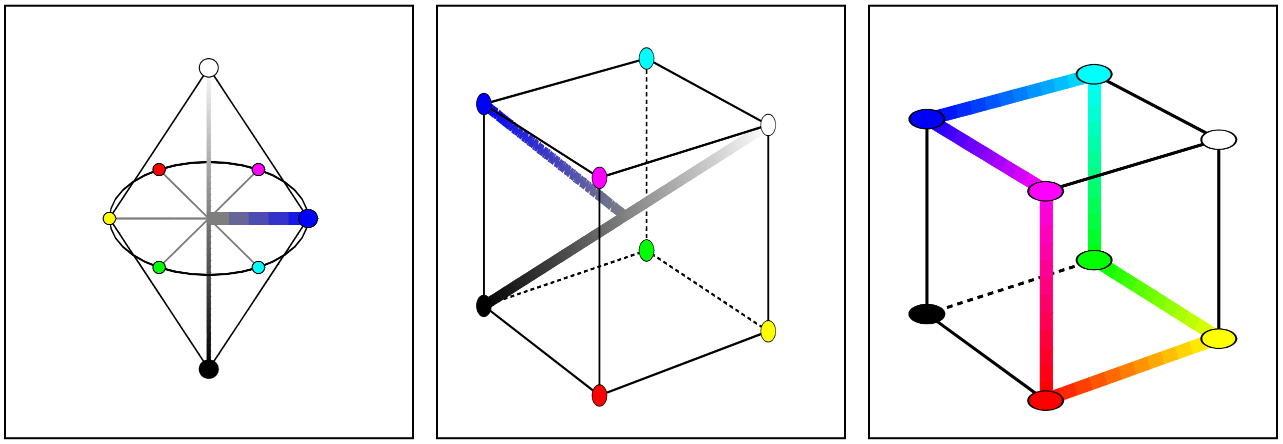
```
D=IMGGIF   B=1200     R= 26 C= 76 P=256
H= 13 W=  8 MAX=  0 D=8000000000000000
RF=8000800000000000 S=0000000000000000
OPTS=3502304009280008 NCOLORS= 10

First byte = 35 [hex] = 00110101 [binary]
```

Since the eighth bit is 1, the device supports user-defined colors (See page 155 in SAS/GRAPH Reference, Version 8 Volume1). What this capability means is that the RGBGIFS macro can easily be modified to generate HLS GIF files. The slightly modified source code therefore is not being reviewed in this section.

Unfortunately this convenience does not extend to the margins. Instead, HLS codes have to be translated into RGB for direct insertion into the HTML file. The task though is not too difficult, because lightness and saturation shown in Figure 7 are configured very similarly in both color spaces.

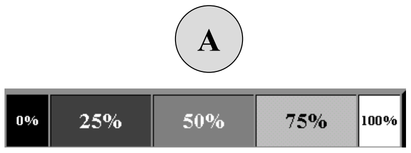
Figure 7. Lightness and saturation are configured similarly in both the HLS and RGB color spaces. The minimum and maximum lightness values are always black and white, and unsaturated grays are embedded in both spaces. Hues are also mapped in a similar fashion. The edge colors in the third panel trace the brightest hue scale in Figure 6.



Generating HTML commands for the Margins

The lightness margin is easy to translate into RGB. Because gray runs along the main diagonal in the RGB cube, all three components are equal in value. Furthermore the values are the same in both color spaces. Black, for example, is (0,0,0) in RGB, and has a lightness value of 0 in HLS. Below is the source code for the Lightness margin in the HLS palette:

```
do L= 0 to 100 by &LS_Incr;
   Lite=round((L/100)*255);
   PctLbl=put(L,3.);
   substr(bgColor,1,2)=put(Lite,hex2.);
   substr(bgColor,3,2)=put(Lite,hex2.);
   substr(bgColor,5,2)=put(Lite,hex2.);
   if L le 50 then fgColor='FFFFFF';
   else fgColor='000000';
   if L eq 0 or L eq 100 then ❶
    string='<TH HEIGHT=48 WIDTH=38 BGCOLOR="#'||bgColor||'"> <FONT SIZE='||fontSmall||'
            COLOR = "#'||fgColor||'"> '||trim(PctLbl)||'% </FONT></TH> ';
```

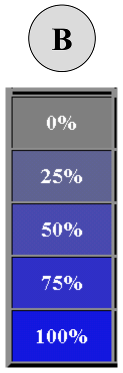


```
      else
        string='<TH HEIGHT=48 WIDTH=96 BGCOLOR="#'||bgColor||'"> <FONT SIZE='||fontBig||'
               COLOR = "#'||fgColor||'"> '||trim(PctLbl)||'% </FONT></TH> ';
      output;
   end;
```

❶ When Lightness is either 0 or 100, the cell width and font size are reduced to cut down on table redundancy.

Programming saturation in RGB is not so straightforward, because the looping structure involves the simultaneous manipulation of the three color-components:

```
   sr=128; sg=128; sb=128; tr=0; tg=0; tb=255; ❶
   n=100/&LS_Incr. + 1; ❷
   i=0;
   do S= 0 to 100 by &LS_Incr;
     fontSize=fontBig;
     r=round(sr+(tr-sr)*i/n); ❸
     g=round(sg+(tg-sg)*i/n);
     b=round(sb+(tb-sb)*i/n);
     i=i+1;
     substr(bgColor,1,2)=put(r,hex2.);
     substr(bgColor,3,2)=put(g,hex2.);
     substr(bgColor,5,2)=put(b,hex2.);
     fgColor='FFFFFF';
     string='<TD ALIGN=CENTER BGCOLOR="#'||bgcolor||'"> <FONT SIZE='||fontSize||'
             COLOR = "#'||fgColor||'"> <B>'||trim(PctLbl)||'% </B></FONT></TD>';
     output;
   end;
```


B
0%
25%
50%
75%
100%

❶ Values are assigned to **s**ource and **t**arget endpoint codes going from the unsaturated gray in the center of the cube to the fully saturated blue at a vertex.
❷ ***n*** for a 25% interval palette equals 100/25=4+1 for 0%, 25%, 50%, 75% and 100%.
❸ Correct color values for intermediate points in the line are calculated with an application of component wise linear interpolation. The formula is:

$$r = sr + (tr - sr) \times {}^{i}\!/_{n}$$

where ***r*** is the interpolated value for the red component, ***sr*** and ***tr*** represent the source and destination component values, ***i*** is the iteration number, and ***n*** represents the number of iterations in the loop.

Summary and Conclusions

Methods for constructing RGB and HLS palettes that reflect the structure of the Version 9 color utility macros are described in the paper. Because palette dimensions can be altered, a method is needed that accommodates large numbers of colors in a single display. Embedding GIF files in HTML is superior to PROC GREPLAY in SAS/GRAPH software because there is no limitation on the number of colors that can be displayed. What happens when the 255-color limitation is exceeded in a SAS/GRAPH application is shown by example in the Appendix. Also listed in the Appendix are more highly refined 10%-interval palettes.

References

Carey, Patrick. *Creating Web Pages with HTML: 2nd Edition*. Cambridge, MA: Course Technology, 2000.

http://www.hypersolutions.org/pages/colorDef.html. Provides *hue, lightness*, and *saturation* definitions used in the paper. Additional terms such as *color, tint* and *dithering* are also defined.

Foley, J.D. and A. Van Dam. *Fundamentals of Interactive Computer Graphics*. Reading, MA: Addison-Wesley Publishing Company, 1983.

SAS Institute Inc. *SAS/GRAPH® Software: Reference, Version 8, Volume 1.* Cary, NC: SAS Institute Inc., 1999.

SAS Institute Inc. *SAS OnLineDoc V9*. Cary, NC: SAS Institute Inc., 2002.

Watts, Perry. *Advanced Programming Techniques for working with Color in SAS® Software*. Proceedings of the Twenty-Ninth SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2004, paper #091.

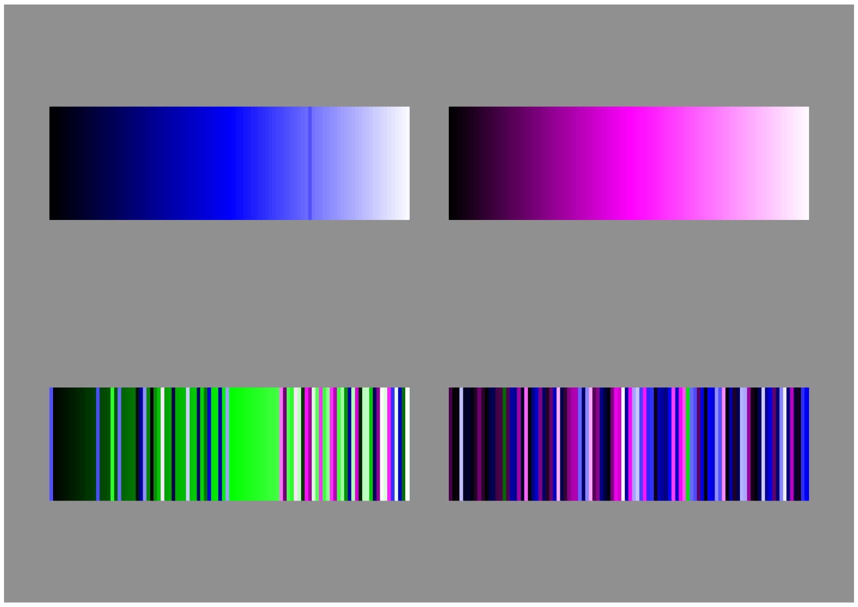
Watts, Perry. *Working with RGB and HLS Color Coding Systems in SAS® Software*. Proceedings of the Twenty-Eighth SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2003, paper #136.

Appendix

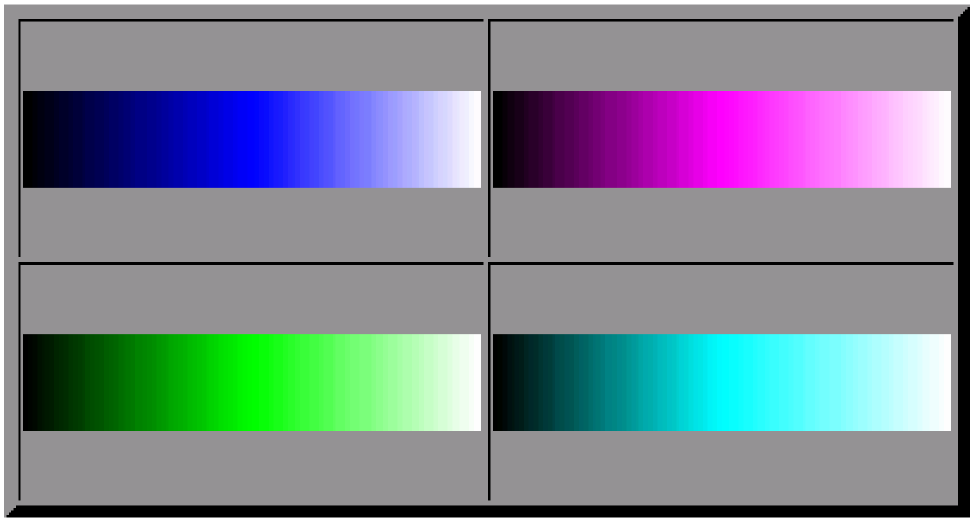
Color Scale Output

Top Panel: PROC GREPLAY Output to a CGM File **Lower Panel:** Embedded GIFs in an HTML File

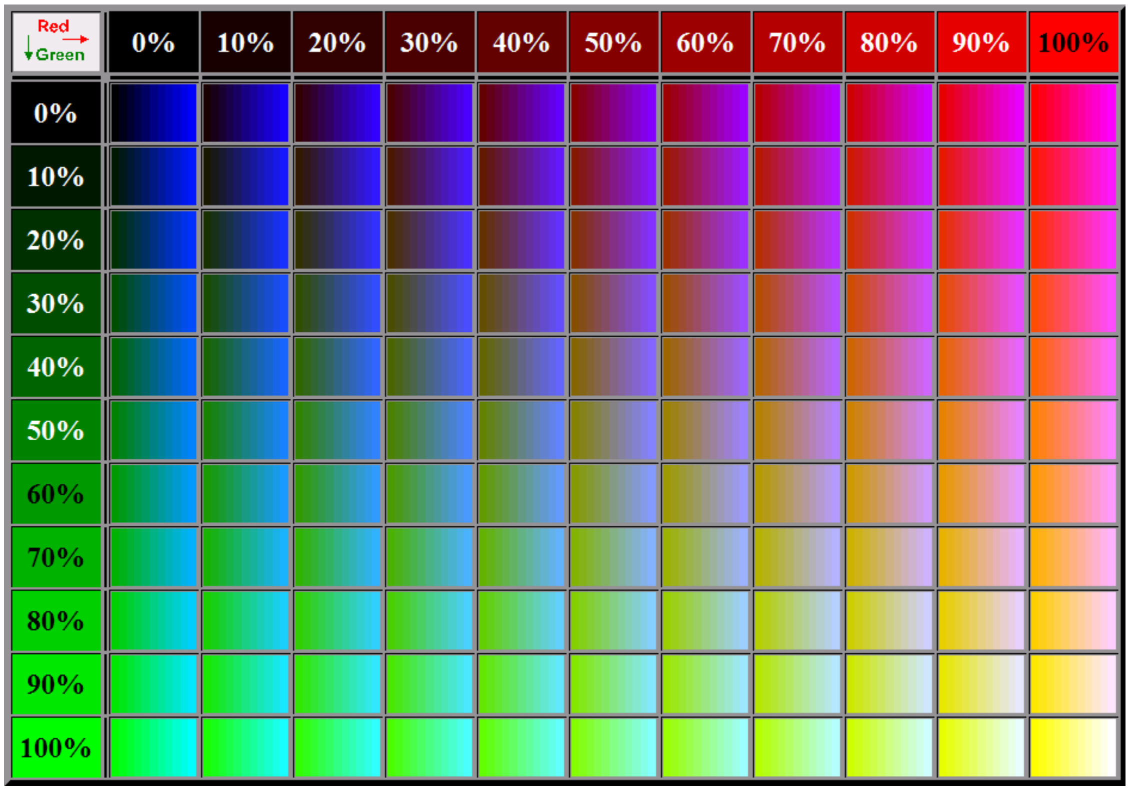
When the 255 color limitation is reached, SAS selects colors at random from existing color names, and issues messages such as the following to the LOG: `ERROR: Color table full, H0F057FF was not added.`

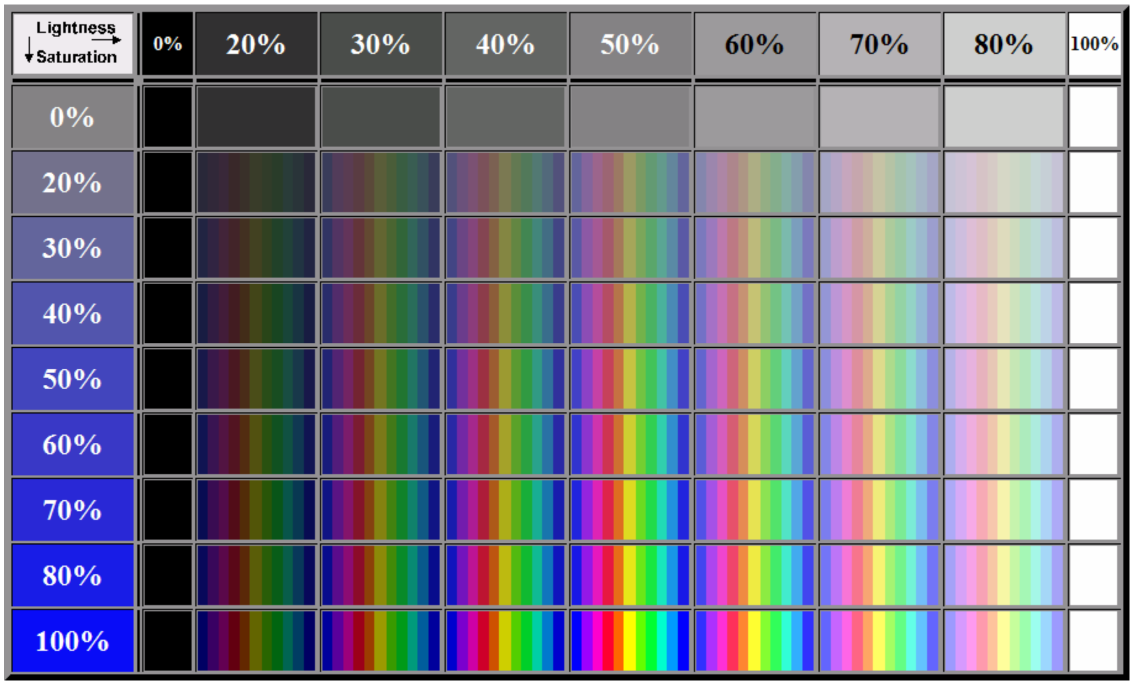


A screen snapshot of Embedded GIFs in an HTML file shows that 510 colors can be managed without difficulty.



Full-Sized RGB and HLS V9 Compatible Palettes





Trademark Citation

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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

Contact Information

The author welcomes feedback via email at: **perryWatts@comcast.net**