Gernot Hoffmann
Color Models
RGB / HLS / HSB

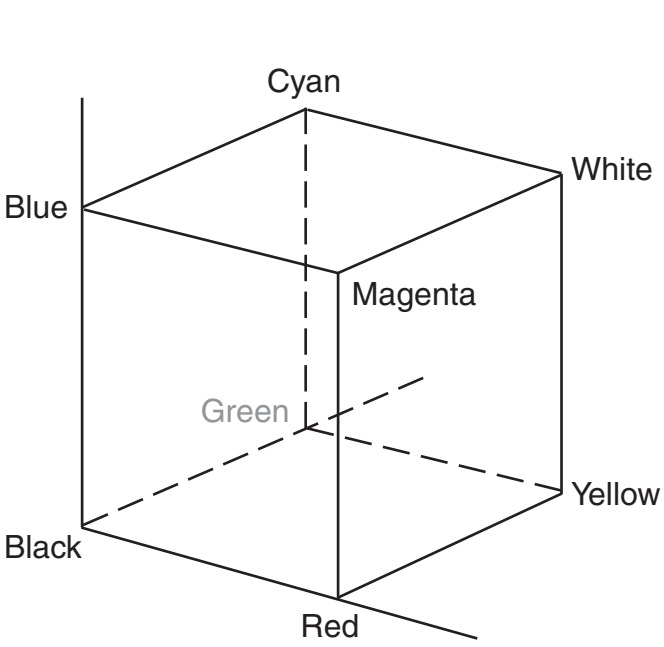
Contents

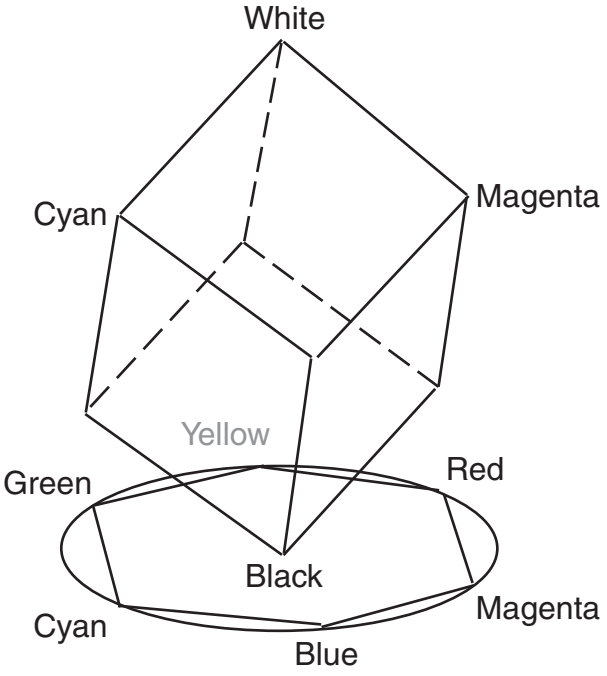
Best view 72 dpi / zoom 100%

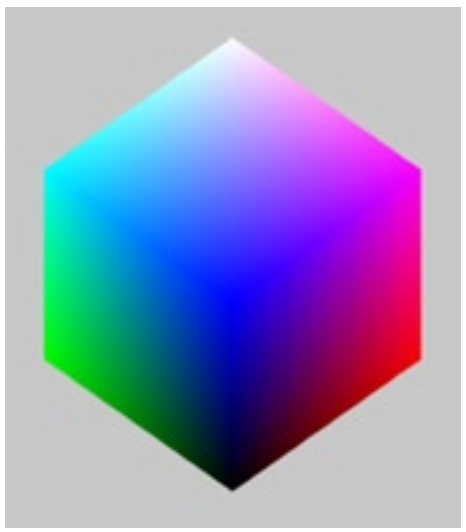
1. Hue-Lightness-Saturation Cone HLS

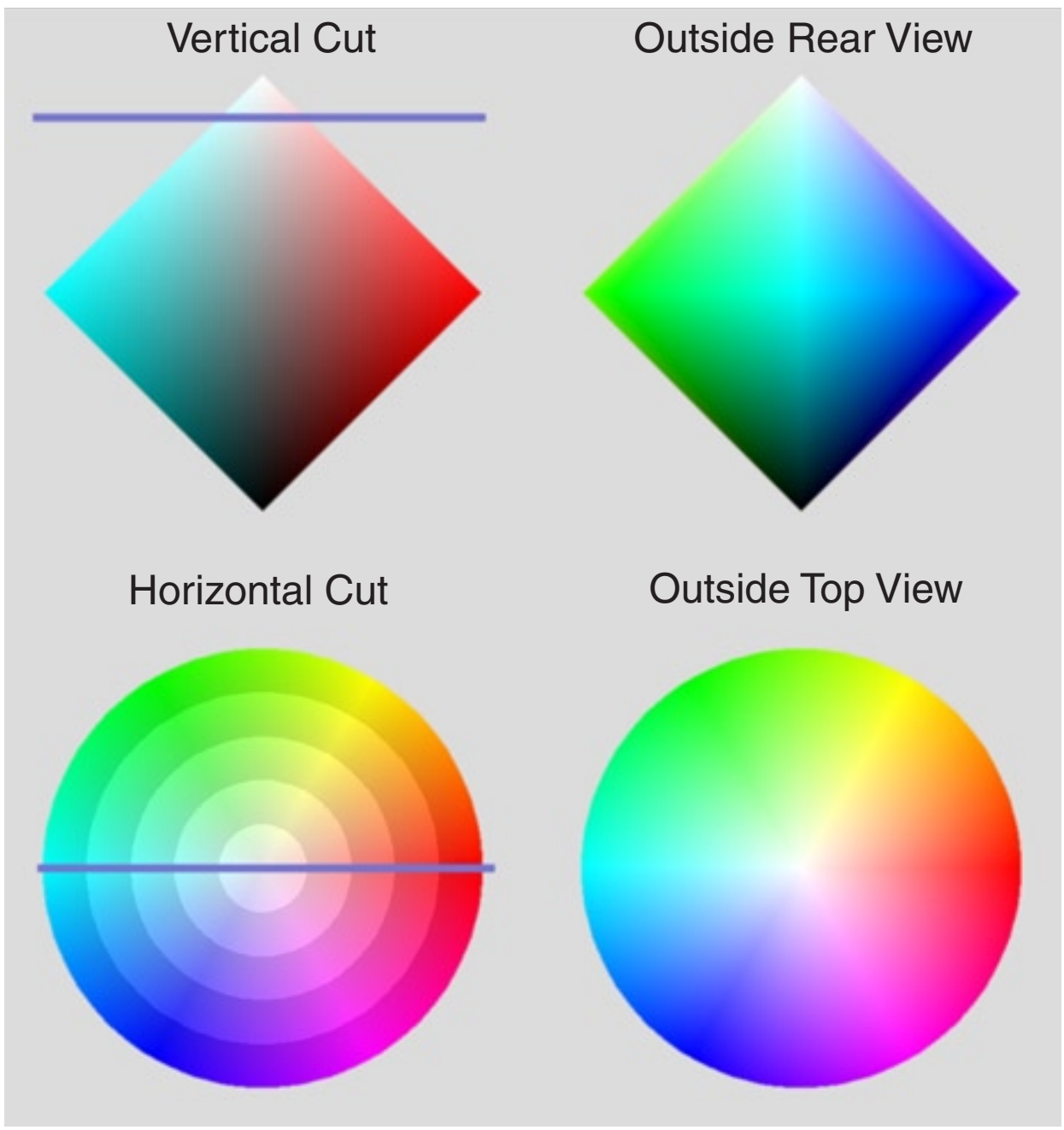
Align an RGB Cube along the Black-White diagonal vertically. This is the color model by H.Küppers (1971).
Then assign a double cone to the Küppers Model. The HLS double cone was mentioned by Ridgway (1912).

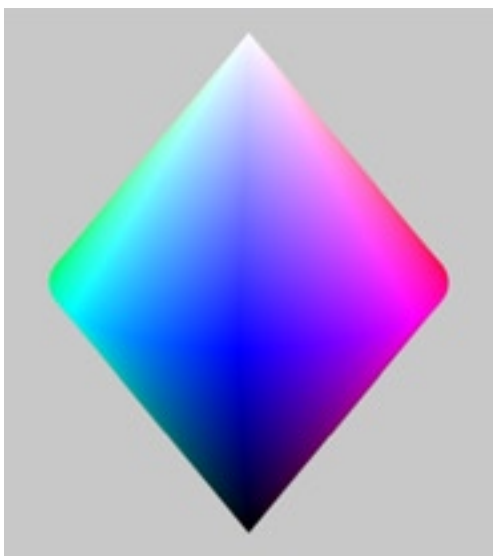
The vertical axis is the Lightness in the range 0...1.
The angle is the Hue in the range 0...360 (degrees).
The relative radius is the Saturation in the range 0...1.







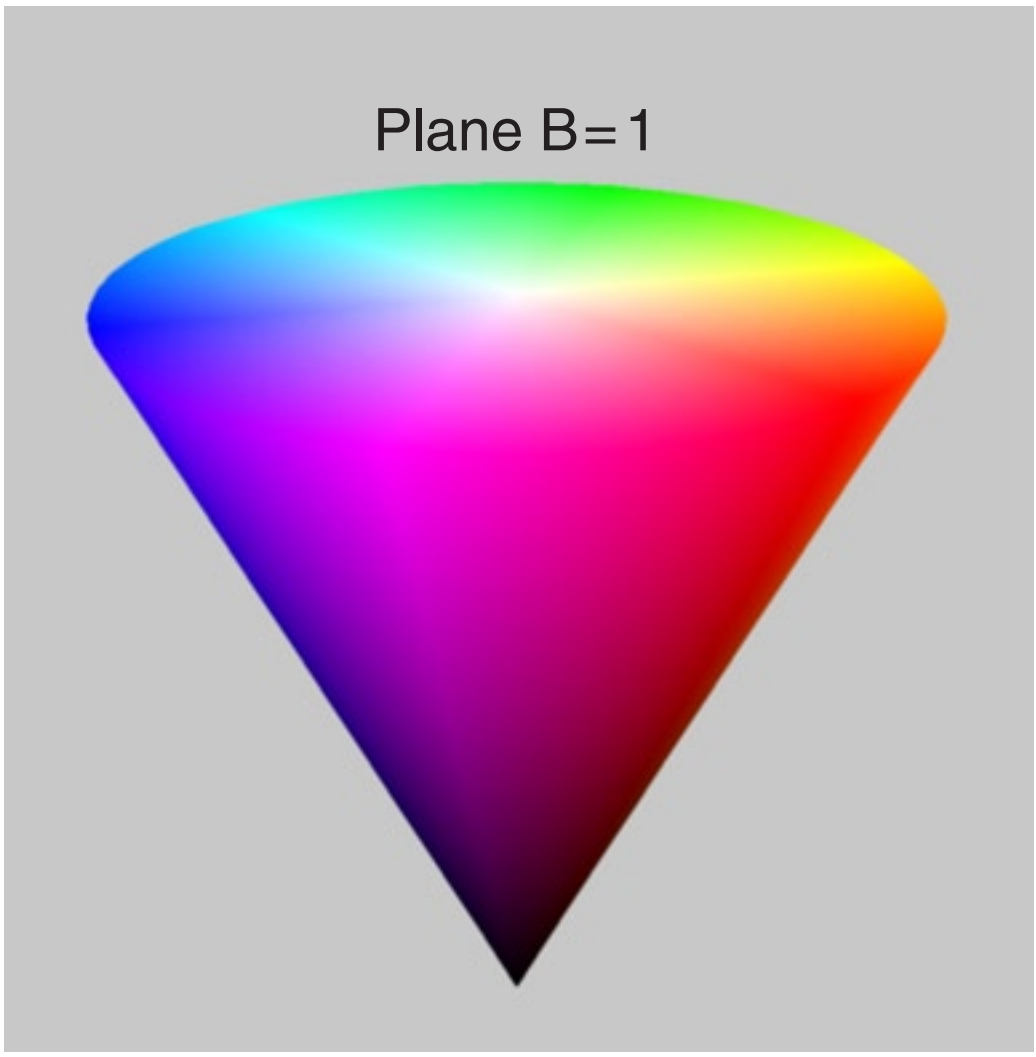




2. Hue-Saturation-Brightness Cone HSB-HSV

This is also called HSV, Hue-Saturation-Value model. Opposed to the HLS cone the upper part is flat. HSB is used in Photoshop.

The vertical axis is the Brightness in the range 0...1.
The angle is the Hue in the range 0...360.
The relative radius is the Saturation in the range 0...1.



The HSB-HSV cone is originally a hexagonal cone. The computer graphic is less complex for a smooth circular cone. The impression is practically the same.
HSB-HSV is not recommended because the interpolation is very inconvenient. The Brightness is the maximum of R,G,B.

Examples

Color	R	G	B	H	L	S	H	S	B
Black	0	0	0	--	0	0	--	0	0
White	255	255	255	--	1	0	--	0	1
Gray	g	g	g	--	0...1	0	--	0	0...1
Red	255	0	0	0	0.5	1	0	1	1
Yellow	255	255	0	60	0.5	1	60	1	1

3. Hoffmann HLS and Sphere

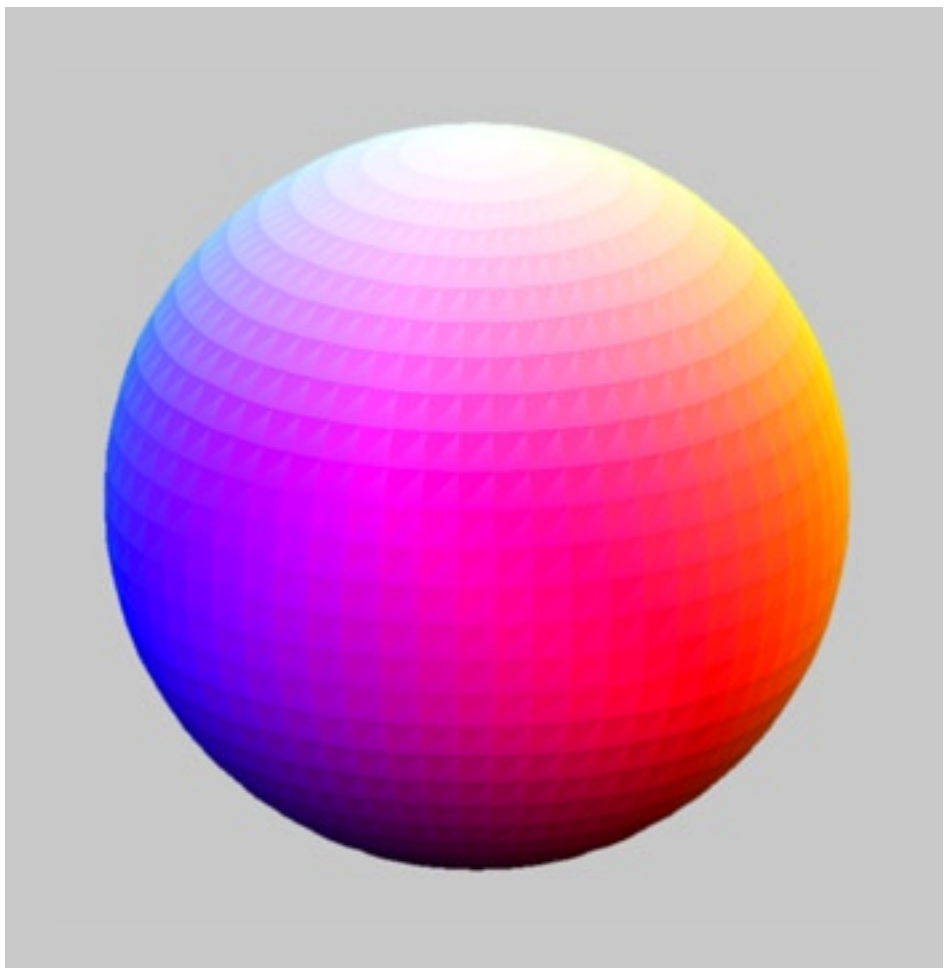
Many other models are possible. The Hoffmann Cone shows more similarity to the vertical RGB Cube.
This cone was made for computer graphics and image processing.
The standard HLS model (Foley et al.) cannot be used for image processing because of terrible discontinuities near to the poles.

In both graphics artificial patterns are drawn to improve the 3D impression.

Sphere

H = Longitude 0...360
L = Latitude -90...+90
S = Radius 0...1

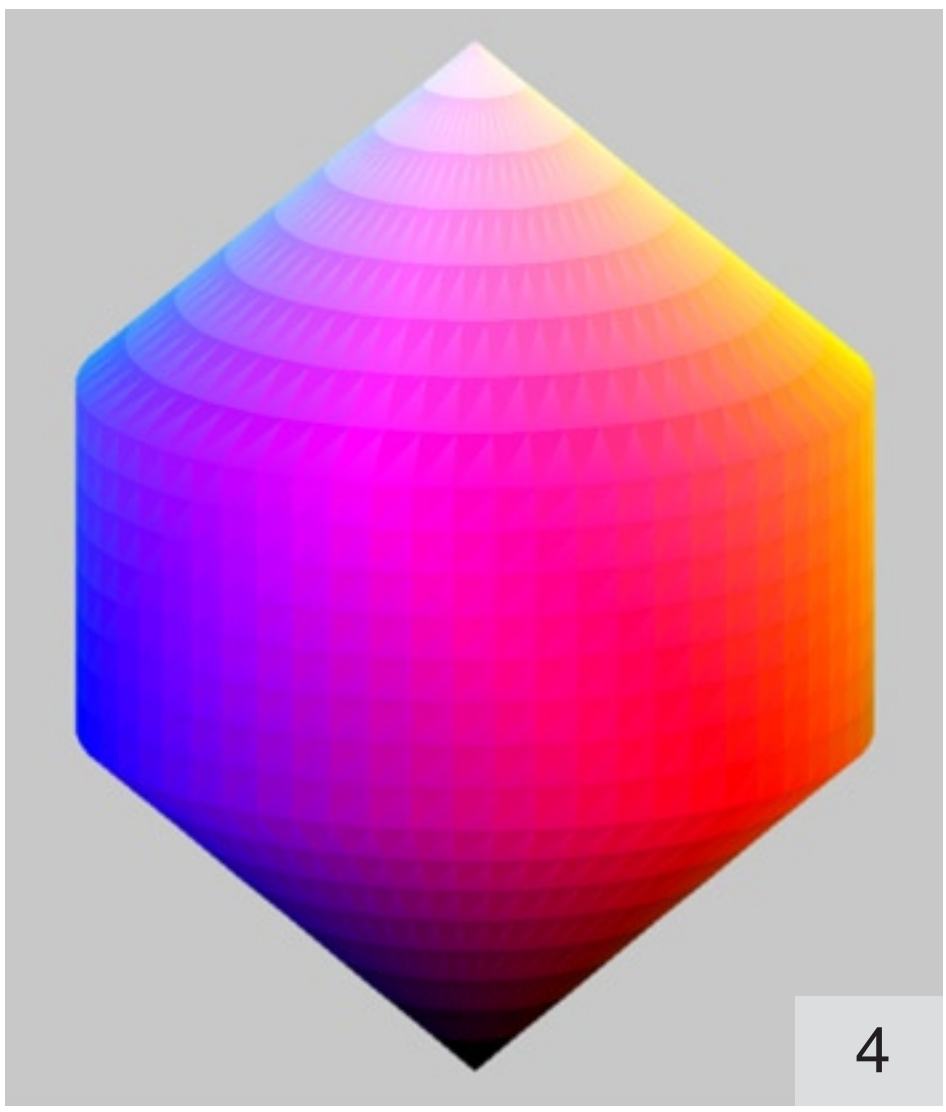
Philip Otto Runge 1810
Wilhelm Wundt 1874



Hoffmann HLS (1996)

H = Hue 0...360
L = Lightness 0...1
S = Radius 0...1

Absolute Radius
Primaries RGB at L=1/3
Secondaries CMY at L= 2/3



4.1 Mathematics for Hoffmann HLS

The color models in this document are *not* color order systems, opposed to CIE concepts or Munsell. A color order system is a conceptual system of organized color perceptions [5]. Coordinates RGB and HLS mean the same colors in different coordinate systems. Physical colors are achieved by a transformation to CIE, which requires primaries and the white point for RGB.
Hoffmann HLS is used for modifications dH,dL,dS and the Contrast dC in images. One may use the 1/3 weights or the NTSC weights for Luminance or Lightness. The difference is not remarkable.
Contrast dC is applied by an S-like shape of the L-function, e.g. by adding dL by a negative sine or a 3-degree polynomial or a similar table function.
The algorithm is programmed straightforward, not by matrices. A Gamma correction is helpful, but G=2.2 (working space sRGB) did not deliver the best results. Examples on p.8 were made by G=1.6 and on p.9 by G=1.4.

Coordinate vectors

$$\mathbf{r}_1 = (r_1,\ g_1,\ b_1)^T$$
$$\mathbf{u}_1 = (u_1,\ v_1,\ w_1)^T$$
$$\mathbf{u}_2 = (u_2, v_2, w_2)^T$$
$$\mathbf{r}_2 = (r_2,\ g_2, b_2)^T$$

Algorithm

$$\mathbf{r}_1 = (1/255)\mathbf{r}_1$$
$$\mathbf{r}_1 = \mathbf{r}_1^G$$
$$\mathbf{u}_1 = \mathbf{T}_{ur}\,\mathbf{r}_1$$
$$S_1 = \sqrt{u_1^2 + v_1^2}$$
$$S_2 = 1$$
If $S_1 > \varepsilon$ Then $S_2 = 1 + dS$
$$w_1 = w_1 + dL$$
$$\mathbf{u}_2 = \mathbf{H}_{21}\,\mathbf{u}_1$$
$$\mathbf{r}_2 = \mathbf{T}_{ru}\,\mathbf{u}_2$$
$$\mathbf{r}_2 = \mathbf{r}_2^{1/G}$$
$$\mathbf{r}_2 = 255\,\mathbf{r}_2$$
Clip r_2, g_2, b_2 by [0,255]

Matrices with 1/3 weight / Hoffmann HLS

$$\mathbf{T}_{ur} = \begin{bmatrix} +1 & -1/2 & -1/2 \\ 0 & +\sqrt{3}/2 & -\sqrt{3}/2 \\ +1/3 & +1/3 & +1/3 \end{bmatrix}$$

$$\mathbf{T}_{ru} = \begin{bmatrix} +2/3 & 0 & +1 \\ -1/3 & +1/\sqrt{3} & +1 \\ -1/3 & -1/\sqrt{3} & +1 \end{bmatrix}$$

Matrices with NTSC weight

$$\mathbf{T}_{ur} = \begin{bmatrix} +1 & -1/2 & -1/2 \\ 0 & +\sqrt{3}/2 & -\sqrt{3}/2 \\ +0.2990 & +0.5870 & +0.1140 \end{bmatrix}$$

$$\mathbf{T}_{ru} = \begin{bmatrix} +0.7010 & -0.2731 & +1.0 \\ -0.2990 & +0.3043 & +1.0 \\ -0.2990 & -0.8540 & +1.0 \end{bmatrix}$$

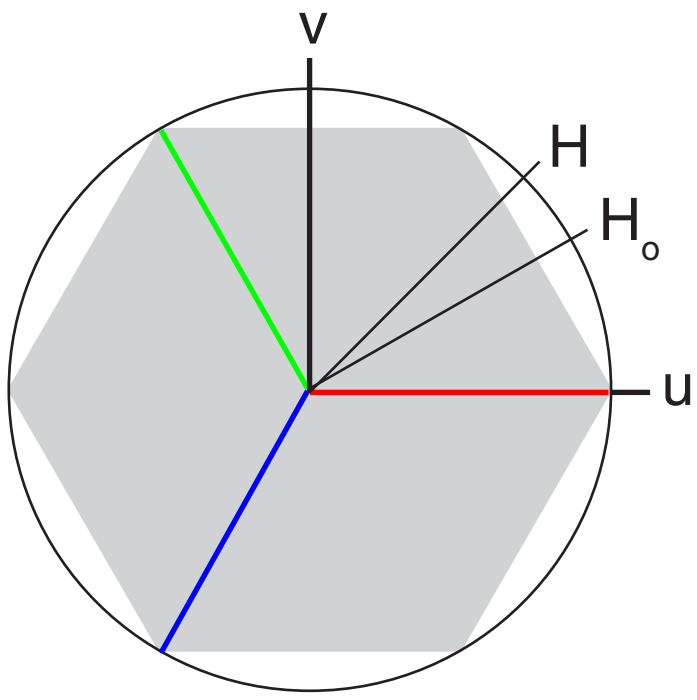
Hue and Saturation correction

$$\mathbf{H}_{21} = \begin{bmatrix} +S_2\cos(dH) & -S_2\sin(dH) & 0 \\ +S_2\sin(dH) & +S_2\cos(dH) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.2 Mathematics for Hoffmann HLS

The HLS system on the previous page is very reasonable for image processing, but there is one disturbing feature: the Saturation can reach 1.0 only for primaries R,G,B and secondaries C,M,Y but not between. In fact, all colors are inside the hexagon or on the contour.

Therefore the model has to be improved. S=1 should be valid on the contour of the hexagon. For better readability the angles are written in degrees.



H_0 is the nearest angle 30, 90,150, 210, 270, 3

$$\text{Trunc}(x) = \text{Round}(x - 0.499999)$$

$$H_0 = 30 + 60\,\text{Trunc}(H/60)$$

$$S = \frac{\sqrt{u^2+v^2}\cos(H-H_0)}{\cos(30)}$$

$$\cos(H-H_0) = \cos(H)\cos(H_0) + \sin(H)\sin(H_0)$$

$$\tan(H) = v/u = \sin(H)/\cos(H)$$

$$\sin(H) = v/\sqrt{u^2+v^2}$$

$$\cos(H) = u/\sqrt{u^2+v^2}$$

$$S = \frac{u\cos(H_0) + v\sin(H_0)}{\cos(30)}$$

The Saturation can be calculated using the formula above or – much simpler – without without Hue. In the actual program we define the constants in advance and avoid divisions.

```
t=u·Sqrt(3)
If v>=0 Then
Begin
   If +t>+v Then   S=+u  +(1/t)·v Else     {  30 }
   If -t>+v Then   S=-u  +(1/t)·v Else     { 150 }
                   S=    +(2/t)·v          {  90 }
End Else
Begin
   If +t>-v Then   S=+u  -(1/t)·v Else     { 330 }
   If -t>-v Then   S=-u  -(1/t)·v Else     { 210 }
                   S=    -(2/t)·v          { 270 }
End
```

4.3 Mathematics for Hoffmann HLS

Finally we need the accurate inverse transformation. H, L,S are given and we look for u,v. Of course, L is not relevant here.

$$H_0 = 30 + 60\,\mathrm{Trunc}(H/60)$$

$$\tan(H) = v/u = \sin(H)/\cos(H)$$

$$u\sin(H) - v\cos(H) = 0$$

$$u\cos(H_0) + v\sin(H_0) = S\cos(30)$$

The two linear equations are solved by Cramer's rule:

$$D = \sin(H)\sin(H_0) + \cos(H)\cos(H_0)$$

$$A = S\cos(30)\cos(H)$$

$$B = S\cos(30)\sin(H)$$

$$u = A/D = \left(\frac{S\cos(30)}{D}\right)\cos(H)$$

$$v = B/D = \left(\frac{S\cos(30)}{D}\right)\sin(H)$$

Using 1/3 weight, the matrix multiplication by $\mathbf{T}_{ru}$ can be simplified by scaled variables $(u/3)$ and $(v/\sqrt{3})$:

$$r = 2(u/3) + L$$

$$g = -(u/3) + (v/\sqrt{3}) + L$$

$$b = -(u/3) - (v/\sqrt{3}) + L$$

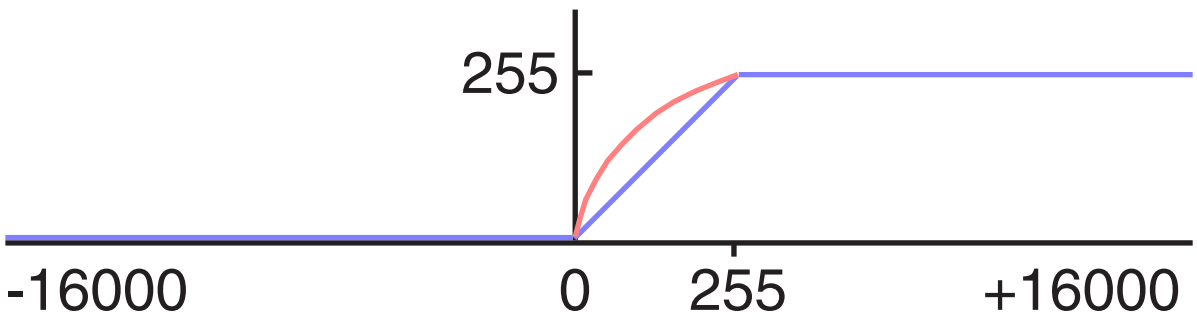
For users who are familiar with the Foley HLS model it might be surprising that the primaries have L=1/3 and the secondaries L=2/3.
It´s also quite unusual that the Saturation is the absolute radius instead of the relative.

Altogether, the Foley model is more convenient for choosing colors, but Hoffmann HLS is better for image processing and for any kind of interpolation.

Proportional clipping improves the quality

```
m=max(r,g,b)
If m>255 Then
Begin
s=255/m
r=Round(r*s)
g=Round(g*s)
b=Round(b*s)
End
```

Table clipping is very fast because the instruction pipeline is not disturbed by jumps.
Either linear (blue curve) or together with a gamma correction (red curve).



5. Examples for Hoffmann HLS

Original (top)

Modified (right, in test window)

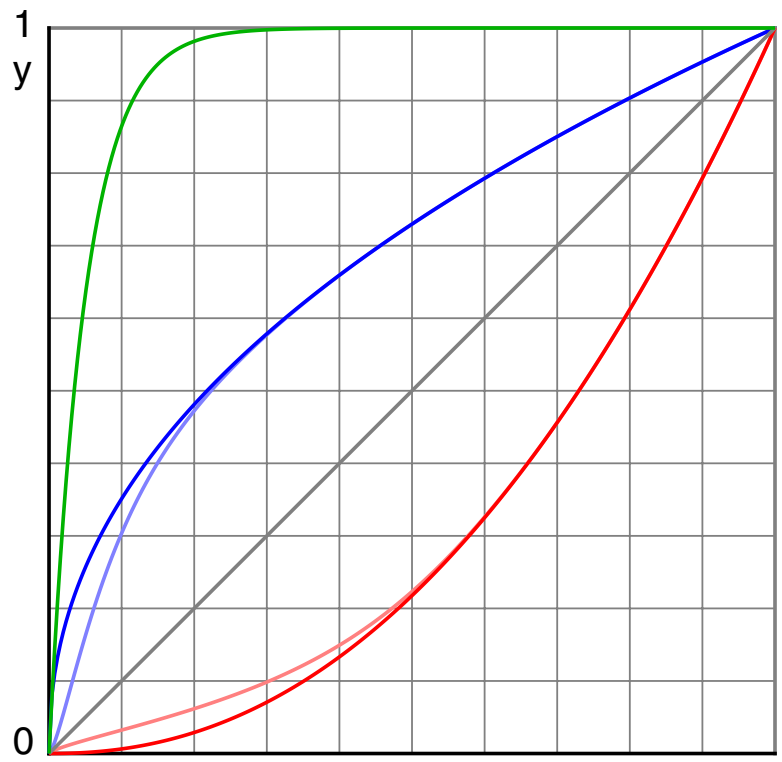
Numbers for changes in Lightness, Saturation and Contrast are relative values. Hue in degrees.

Best view 72 dpi / zoom 100%

6.1 Improved Tone Reproduction Curves

The conversion by Gamma-Functions (e.g. G=2.2) can be improved.

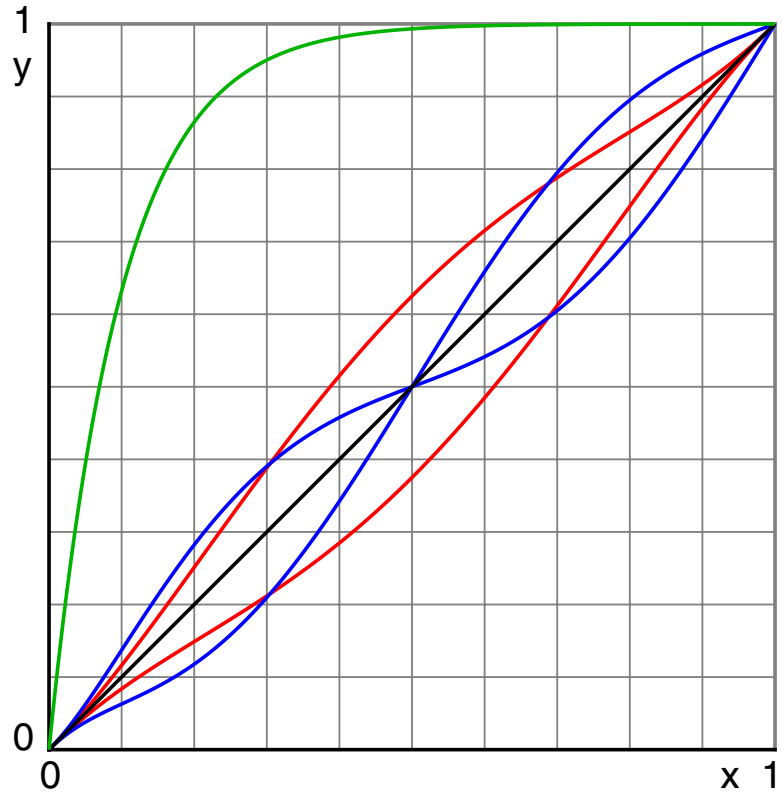
1. Dark blue $y = x^{1/G}$
2. Dark red $y = x^{G}$
3. Light blue $y = x^{1/G}(1-e^{-x/0.05})$
4. Light red true inverse function for 3
5. Green $y = 1-e^{-x/0.05}$



Curve 2 is used for the conversion of nonlinear image data into the linear light space. Curve 1 works in opposite direction. The horizontal slope of curve 2 leads to lost dark levels. Therefore the effects are washed out in 3 and 4 by the exponential function 5. The inverse function is found by Newton iteration (source code in Pascal).

Here we have tone reproduction functions for midtones and contrast. These are applied only to the Lightness channel.

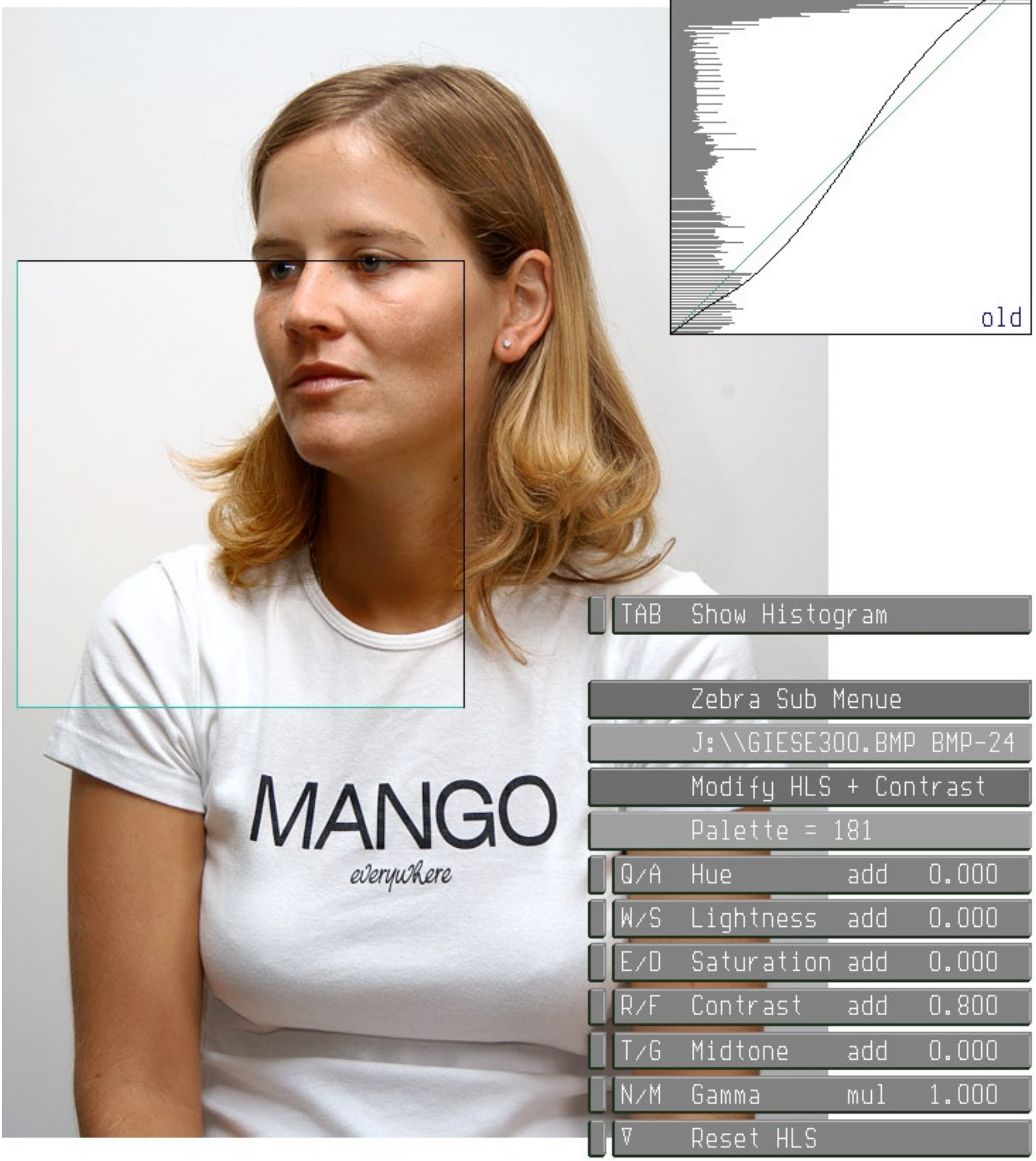
1. Red $y = x+2Mx^2(1-x)^2$
2. Blue $y = x-0.1C\sin(2\pi x)(1-e^{-x/0.1})$
3. Green $y = 1-e^{-x/0.1}$



Curve 1 is used for midtones in the range M=-1..+1. Curve 2 is used for contrast in the range C=-1..+1.
Contrast for dark colors is washed out by an exponential function 3 .
The source code shows additionally a *further* gamma conversion for gamma in the range G=0.5..1.5 .

6.2 Improved Tone Reproduction Curves / Example

Strong contrast. The TRC for the L-channel and the histogram are valid for the small test window. Contrast does not change the hues. Applied by G=1.4. Best view 72 dpi / zoom 100%.



6.3 Improved Tone Reproduction Curves / Code

Gamma functions

```
Procedure Ga1Func(x,gai: Single; Var y: Single);
Var flag: Integer;
Begin
XPowerA(x,gai,y,flag); y:=y*(1-exp(-x/0.05));
End;

Procedure GamFill(GamTab: LTyp; gam: Single);
Var  i,k,flag: Integer;
     u,v1,v2,dv,gai,f1,f2: Single;
Const i255=1/255;
Begin
dv:=0.0001;
gai:=1/gam;
For i:=-lTab To 0    Do GamTab[i]:=0;
For i:= 256  To lTab Do GamTab[i]:=255;
For i:=   1  To 255  Do
 Begin
 u:=i*i255;
 XPowerA(u,gam,v1,flag); { initial, then Newton }
 k:=1;
 Repeat
     Ga1Func(v1,gai,f1); f1:=f1-u;
     v2:=v1+dv;
     Ga1Func(v2,gai,f2); f2:=f2-u;
     v1:=v1-f1*dv/(f2-f1);
     Inc(k);
 Until (Abs(f1)<dv) Or (k=6);
 GamTab[i]:=Round(255*v1);
 End;
End;

Procedure GaiFill(GaiTab: LTyp; gam: Single);
Var i      : Integer;
    x,y,gai: Single;
Const i255=1/255;
Begin
gai:=1/gam;
For i:=-lTab To 0    Do GaiTab[i]:=0;
For i:= 256  To lTab Do GaiTab[i]:=255;
For i:=   1  To 255  Do
 Begin
     x:=i*i255;
     Ga1Func(x,gai,y);
     GaiTab[i]:=Round(255*y);
 End;
End;
```

Pascal procedures are showing structures.
Some of them cannot be simply copied because they are based on a library in the background.

Midtones and contrast

```
{ Inputs conT,midT in the range -1..+1, gamR 0.5..1.5 }
conT:=-0.1*conT;
midT:= 2*midT;
For i:=0 to 255 Do
 Begin
  x1:=i*i255;
{ Contrast control conT }
  y1:=conT*sic(x1*c2pi)*(1-exp(-x1/0.1));  { c2pi=2*pi }
{ Gamma control gamR + Midtone control midT }
  XPowerA(x1,gamR,y2,flag);
  y3:=midT*Sqr(x1)*Sqr(1-x1);
  TabG[i]:=y1+y2+y3;
 End;
```

7. Standard HLS Code / Pascal

```
Procedure HLStoRGBf (H,L,S: Single; Var rb,gb,bb: Byte);
{  Standard Foley + van Dam                          }
{  G.Hoffmann, Tutorial version                      }
{  H: 0..360, L: 0..1, S: 0..1; rb,gb,bb: 0..255;    }
Var   min,max,mmm,hue,col : Single;
Begin
While H>360.0 Do H:=H-360.0;
While H<0.0   Do H:=H+360.0;
If S<0 Then S:=0 Else If S>1 Then S:=1;
If L<0 Then L:=0 Else If L>1 Then L:=1;
If L<=0.5 Then max:=L*(1.0+S) Else max:=L+S-L*S;
min:=2*L-max; mmm:=(max-min)/60;
hue:=H+120.0;
      If hue>360.0 Then hue:=hue-360.0;
      If hue< 60.0 Then col:=min+mmm*hue
      Else If hue<180.0 Then  col:=max
      Else If hue<240.0 Then  col:=min+mmm*(240.0-hue)
      Else                    col:=min;
      rb:=Round(255.0*col);
hue:=H;
      If hue< 60.0 Then col:=min+mmm*hue
      Else If hue<180.0 Then  col:=max
      Else If hue<240.0 Then  col:=min+mmm*(240.0-hue)
  Else                        col:=min;
      gb:=Round(255.0*col);
hue:=H-120.0;
      If hue<0.0 Then hue:=hue+360.0;
      If hue< 60.0 Then col:=min+mmm*hue
      Else If hue<180.0 Then  col:=max
      Else If hue<240.0 Then  col:=min+mmm*(240.0-hue)
  Else                        col:=min;
  bb:=Round(255.0*col);
End;
```

Pascal procedures are showing structures.
Some of them cannot be simply copied because they are based on a library in the background.

```
Procedure RGBtoHLSf (rb,gb,bb: Byte; Var H,L,S: Single);
{  Standard Foley + van Dam                          }
{  G.Hoffmann, Tutorial version                      }
{  rb,gb,bb: 0..255; H: 0..360, L: 0..1, S: 0..1;    }
Var R,G,B,sum,dif,max,min: Integer; dsf: Single;
Const r510:Single=1/510;
Begin
S:=0; H:=0;
R:=rb; G:=gb; B:=bb;
max:=R; If G>max Then max:=G; If B>max Then max:=B;
min:=R; If G<min Then min:=G; If B<min Then min:=B;
sum:=max+min;
dif:=max-min;
L:=sum*r510;
If dif>0 Then
 Begin
  If L<=0.5 Then S:=dif/sum Else S:=dif/(510-sum);
  dsf:=60/dif;
  If R=max Then  H:=    (G-B)*dsf Else
  If G=max Then  H:=120+(B-R)*dsf Else
                 H:=240+(R-G)*dsf;
  If H<0 Then H:=H+360;
 End;
End;
```

8. Hoffmann HLS Code / Pascal

This code does not contain gamma encoding.
The luminance is calculated by 1/3 weights.
The actual implementation uses much assembly code.

```
Const ci25: Single=1/255;
      c333: Single=0.333333;
      c500: Single=0.500000;
      c866: Single=0.866025;
      c577: Single=0.577350;
      c115: Single=1.154701;
      c173: Single=1.732050;
      wdeg: Single=180/pi;
      c255: Single=255;
      c666: Single=0.666667;
      c160: Single=1/60;
      wrad: Single=pi/180;

{ Local version }
```

Pascal procedures are showing structures.
Some of them cannot be simply copied because they are based on a library in the background.

```
Procedure RGBtoHLSho(rb,gb,bb: Byte; Var H,L,S: Single);
{Version 13.02.2002
 rb,gb,bb: 0..255; H: 0..360; L: 0..1; S: 0..1 }
 Var flag,ri,gi,bi        : Integer;
     R,G,B,U,V,T,H0,si,co : Single;
 Begin
 R:=rb*ci25;
 G:=gb*ci25;
 B:=bb*ci25;
 L:=c333*(R+G+B);
 U:=R-c500*(G+B);
 V:=c866*(G-B);
 H:=wdeg*Atac(V,U);              { the same as H=atan2(V,U), a four quadrant arctangent }
 If H<0 Then H:=H+360;
 T:=c173*U;
  If V>=0 Then
  Begin
   If  T>V Then S:= U+c577*V Else
   If -T>V Then S:=-U+c577*V Else
                S:=   c115*V;
  End Else
  Begin
   If  T>-V Then S:= U-c577*V Else
   If -T>-V Then S:=-U-c577*V Else
                 S:=  -c115*V;
  End;
End;
```

```
Procedure HLStoRGBho(H,L,S: Single; Var rb,gb,bb: Byte);
{ Hoffmann
  rb,gb,bb: 0..255; H: any; L: 0..1; S: 0..1; }
{ Requires S—>0 for 0<—L—>1 }
Var U,V,F,f2,H0,D,si,co,s0,c0,r,g,b: Single; ri,gi,bi,f1: Integer;
Begin
  SicCoc(H*wrad,si,co);          { Fast sine+cosine     }
  H0:=30+60*Trunc(H/60);
  SicCoc(H0*wrad,s0,c0);         { Fast sine+cosine     }
  D:=S*c866/(co*c0+si*s0);       { 0.5*Sqrt(3) }
  U:=c333*D*co;                  { scaled U:  1/3       }
  V:=c577*D*si;                  { scaled V:  1/Sqrt(3) }
  R:=L+U+U; G:=L-U+V; B:=L-U-V;
  ri:=Round(c255*R);
  gi:=Round(c255*G);
  bi:=Round(c255*B);
  rb:=LimTab^[ri];               { Limiter by table     }
  gb:=LimTab^[gi];
  bb:=LimTab^[bi];
End;
```

9.1 Standard HLS Code / PostScript

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 624 624
%%Creator:      Gernot Hoffmann
%%Title:        HLStoRGB-01
%%CreationDate: December 07 / 2015

/mm {2.834646 mul} def % points per mm

/Hx0 110 mm def
/Hy0 110 mm def
/Hsx 100 mm def

/Bx  624 def
/By  Bx  def

/Bbox
{ 0.4 mm setlinewidth
  0 setgray
  0 0 moveto Bx 0 rlineto 0 By rlineto Bx neg 0 rlineto closepath stroke
} def

/Axes
{ 0.4 mm Hsx div setlinewidth
  0 setgray
  -1 0 moveto +1 0 lineto stroke 0 -1 moveto 0 +1 lineto stroke
} def

/HLStoRGBf
% HLS Foley
{/sat exch def
 /lig exch def
 /hue exch def
  hue 360 gt {/hue hue 360 sub def} if  Correction
  hue   0 lt {/hue hue 360 add def} if  December 6, 2015
  lig 1.0 gt {/lig 1.0 def    } if
  lig 0.0 lt {/lig 0.0 def    } if
  sat 1.0 gt {/sat 1.0 def    } if
  sat 0.0 lt {/sat 0.0 def    } if
  lig 0.5 le
  {/max 1.0 sat add lig mul def}
  {/max lig sat add lig sat mul sub def} ifelse
 /min lig 2 mul max sub def
 /mmm max min sub 60 div def
 /hux hue 120 add def
  1
  { hux 360 gt {/hux hux 360 sub def } if
    hux  60 lt {/col min mmm hux mul add         def exit} if
    hux 180 lt {/col max                         def exit} if
    hux 240 lt {/col min mmm 240 hux sub mul add def exit} if
   /col min def
 } repeat
/Red col def
/hux hue def
 1
 { hux  60 lt {/col min mmm hux mul add         def exit} if
   hux 180 lt {/col max                         def exit} if
   hux 240 lt {/col min mmm 240 hux sub mul add def exit} if
  /col min def
 } repeat
/Grn col def
/hux hue 120 sub def
 1
 { hux   0 lt {/hux hux 360 add def} if
   hux  60 lt {/col min mmm hux mul add         def exit} if
   hux 180 lt {/col max                         def exit} if
   hux 240 lt {/col min mmm 240 hux sub mul add def exit} if
  /col min def
 } repeat
/Blu col def
```

9. 2 Standard HLS Code / PostScript

```
%/Red Red iga exp def      % no gamma encoding
%/Grn Grn iga exp def
%/Blu Blu iga exp def
  Red Grn Blu setrgbcolor
} def

/WheelPolar
% Polar coordinates
{ gsave
 /dH H2 H1 sub NH div def
 /dL L2 L1 sub NR div def
 /dS S2 S1 sub NR div def
 /dR R2 R1 sub NR div def
 /H H1 dH 0.5 mul add def
  NH
  {/R R1 def
   /L L1 def
   /S S1 def
   NR
   { H L S HLStoRGBf
    /RR R dR add def
     newpath
     0 0 RR 0 dH arc
     0 0 R dH  0 arcn
     closepath
     fill
    /R RR def
    /L L dL add def
    /S S dS add def
   } repeat
   /H H dH add def
    dH rotate
  } repeat
  grestore
} def

/WheelCart
% Cartesian coordinates and clipping
{ gsave
  0 0 1 0 360 arc clip
 /dL L2 L1 sub def
 /dS S2 S1 sub def
 /dX 1 NR div def
 /dY dX def
 /dB dX 0.5 mul def
 /Y -1 def
  NR neg 1 NR
  {pop
  /X -1 def
   NR neg 1 NR
   { pop
    /H Y X atan def
    /R X dup mul Y dup mul add sqrt def
    /L L1 R dL mul add def
    /S S1 R dS mul add def
     H L S HLStoRGBf
     newpath
     X dB sub Y dB sub moveto
     dX 0 rlineto 0 dY rlineto dX neg 0 rlineto
     closepath fill
    /X X dX add def
   } for
  /Y Y dY add def
  } for
  grestore
} def
```

9. 3 Standard HLS Code / PostScript

```
% -- Begin

% Bbox

Hx0 Hy0 translate
Hsx Hsx scale

/gamma 2.2 def
/iga 1 gamma div def

0.4 mm Hsx div setlinewidth
/H1   0 def
/H2 360 def
/L1 1.0 def
/L2 0.5 def
/S1 1.0 def
/S2 1.0 def
/R1 0.0 def
/R2 1.0 def
/NH 180 def % use 18 for test
/NR  50 def % use  5 for test

 WheelPolar

/L1 1.0 def
/L2 0.5 def
/S1 1.0 def
/S2 1.0 def
/NR  50 def

% WheelCart

Axes

showpage
```

10.1 A CMYK Color Wheel

An RGB color wheel is here defined by a view onto the top of an HLS or HSB cone. Viewing along the white–black diagonal of an RGB cube delivers the same appearance if we replace the hexagonal contour by a circle.

If we assign a color space like sRGB or AdobeRGB and convert by a standard CMYK profile like ISOCoated-v2-eci, then the result is disappointing – almost all colors are out of gamut and substituted by crude approximations.

Essentially we would like to have at the circumference normalized CMYK values of this type, all with K=0: 0…1C / 0…1 M / 0…1Y, but one of CMY is always 0. Like Red = 0C / 1M / 1Y or Yellow = 0C / 0M / 1Y.
This is achieved by applying in PostScript this generic conversion:
C=1–R, M=1–G, Y=1–B, together with K=0.

Open the EPS-file (next pages) or this PDF page in any CMYK mode in Photoshop and measure the values by *Info*. Values 1.0 may appear as 99%.

10. 2 A CMYK Color Wheel / PostScript Code

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 624 624
%%Creator:      Gernot Hoffmann
%%Title:        Wheel-CMYK
%%CreationDate: December 6 / 2015

/mm {2.834646 mul} def % points per mm

/Hx0 110 mm def
/Hy0 110 mm def
/Hsx 100 mm def

/HLStoRGBf
% HLS Foley, here modified for CMYK-output
{/sat exch def
 /lig exch def
 /hue exch def
  hue 360 gt {/hue hue 360 sub def} if
 hue   0 lt {/hue hue 360 add def} if
 lig 1.0 gt {/lig 1.0 def    } if
 lig 0.0 lt {/lig 0.0 def    } if
 sat 1.0 gt {/sat 1.0 def    } if
 sat 0.0 lt {/sat 0.0 def    } if
 lig 0.5 le
 {/max 1.0 sat add lig mul def}
 {/max lig sat add lig sat mul sub def} ifelse
/min lig 2 mul max sub def
/mmm max min sub 60 div def
/hux hue 120 add def
 1
 { hux 360 gt {/hux hux 360 sub def } if
   hux  60 lt {/col min mmm hux mul add         def exit} if
   hux 180 lt {/col max                         def exit} if
   hux 240 lt {/col min mmm 240 hux sub mul add def exit} if
  /col min def
 } repeat
/Red col def
/hux hue def
 1
 { hux  60 lt {/col min mmm hux mul add         def exit} if
   hux 180 lt {/col max                         def exit} if
   hux 240 lt {/col min mmm 240 hux sub mul add def exit} if
  /col min def
 } repeat
/Grn col def
/hux hue 120 sub def
  1
  { hux   0 lt {/hux hux 360 add def} if
    hux  60 lt {/col min mmm hux mul add         def exit} if
    hux 180 lt {/col max                         def exit} if
    hux 240 lt {/col min mmm 240 hux sub mul add def exit} if
   /col min def
  } repeat
 /Blu col def
% Red Grn Blu setrgbcolor
% Conversion of RGB into CMYK, with C=1-R, M=1-G, Y=1-B, K=0
 /C 1 Red sub def
 /M 1 Grn sub def
 /Y 1 Blu sub def
 /K 0 def
  C M Y K setcmykcolor
} def
```

10. 3 A CMYK Color Wheel / PostScript Code

```
/WheelPolar
% Polar coordinates
{ gsave
 /dH H2 H1 sub NH div def
 /dL L2 L1 sub NR div def
 /dS S2 S1 sub NR div def
 /dR R2 R1 sub NR div def
 /dH2 dH 0.5 mul def
  dH2 neg rotate % center of red segment at 0°
 /H H1 def
  NH
  {/R R1 def
   /L L1 def
   /S S1 def
   NR
   { H L S HLStoRGBf
    /RR R dR add def
     newpath
     0 0 RR 0 dH arc
     0 0 R dH  0 arcn
     closepath
     fill
    /R RR def
    /L L dL add def
    /S S dS add def
   } repeat
   /H H dH add def
    dH rotate
  } repeat
  grestore
} def

Hx0 Hy0 translate
Hsx Hsx scale

/H1   0 def
/H2 360 def
/L1 1.0 def
/L2 0.5 def
/S1 1.0 def
/S2 1.0 def
/R1 0.0 def
/R2 1.0 def

% Number of radius steps
/NR  17 def % first pass.
% Two-pass strategy avoids interpolation artifacts

% Number of hue steps
% Last one is valid
/NH   6 def %   6 segments
/NH  12 def %  12 segments
/NH 360 def % 360 segments, practically continuous
/NH  12 def

/H1   0 def /H2 360 def
/L1 1.0 def /L2 0.5 def
/S1 1.0 def /S2 1.0 def
/R1 0.0 def /R2 1.0 def

 WheelPolar
/NR 100 def % second pass
 WheelPolar

showpage
```

11. References

[1] J.D.Foley, A.van Dam, S.K.Feiner, J.F.Hughes
Computer Graphics
Addison-Wesley Publishing Company 1993

[2] N.Silvestrini
Idee Farbe
Baumann & Stromer Verlag, Zürich 1994

[3] H.Küppers
Harmonielehre der Farben
DuMont Buchverlag, Köln 1999

[4] Everything about Color and Computers
http://www.efg2.com

[5] Roy S. Berns
Billmeyer and Saltzman‘s
Principles of Color Technology
Third Edition
John Wiley & Sons, Inc.
New York ... 2000

This document:
http://docs-hoffmann.de/hlscone03052001.pdf

Gernot Hoffmann
May 03 / 2001 – December 7 / 2015
Website
Load browser / Click here