Supplemental Color Topics for Programmers

Peter Occil

This version of the document is dated 2024-12-24.

Peter Occil

1 Introduction

This document presents supplemental topics about color. They add to my article on **color topics for programmers**[1]. The **pseudocode conventions**[2] apply to this document.

2 Contents

3 Kinds of Color Spaces

Color spaces are designed to organize colors. They can be categorized as any of the following:

- **Light-mixture** color spaces are based on mixtures of colored light sources (such as RGB, red–green–blue). (The term "additive mixture" is better used to mean ***additive mixture of color stimuli***, rather than light mixture.)
- **Colorant-mixture** color spaces are based on mixtures of inks, dyes, or other colorants (such as CMYK, cyan–magenta–yellow–black). ("Subtractive mixture" is an inferior term.)
- **Hue-based** color spaces have three dimensions, one of which is hue. Examples include HSL, HSV, and CIE (International Commission on Illumination) *L*C*h*.

[1] https://peteroupc.github.io/colorgen.html
[2] https://peteroupc.github.io/pseudocode.html

- **Light/dark** color spaces have three dimensions, one of which is a light/dark dimension. Examples include HSL, CIE *L*C*h*, CIE XYZ, and Y ′ $C_B C_R$. Of these color spaces:
- **Opponent** color spaces are light/dark color spaces arranged in three axes: black/white, red/green, and blue/yellow. Examples include CIELAB and Hunter L,a,b.
- It's discouraged to speak of **"device-dependent"** and **"device-independent"** color spaces, because the difference between the two is not always clear. While color spaces based on how humans perceive color, such as XYZ, may be considered "device independent", some RGB (red–green–blue) color spaces may also be, depending on whether they are convertible, without loss, to XYZ and back (*colorimetric color spaces*).

4 "Primary Colors"

In general, so-called "primary colors" are not relevant to programming except in the context of light-mixture, colorant-mixture, or opponent color spaces.

For light-mixture and colorant-mixture color spaces, the colors of the light sources or colorants those spaces are based on can be called "primary colors".

For opponent color spaces, the four so-called *unique hues* red, green, blue, and yellow, and maybe white and black, can be called "primary colors"; such "primary colors", however, generally serve as axes only.

"Primary colors" can be, and often are, imaginary. For example, the **ACES2065-1 RGB color space**, and certain other RGB color spaces, include imaginary points for one or more "primary colors" in exchange for covering a range of colors not normally possible otherwise.

5 Calculating the Mean Hue Angle

The `MeanAngle` method, as given in the pseudocode below, finds the average of one or more angles expressed in radians (which is important when averaging colors in hue-based color models such as HSL, HSV, and CIE *L*C*h*, which contain hue components that are angles).

```
METHOD MeanAngle(angles)
    if size(angles)==0: return 0
    xm=0
    ym=0
    i=0
    while i < size(angles)
        c = cos(angles[i])
        s = sin(angles[i])
        i = i + 1
        xm = xm + (c - xm) / i
        ym = ym + (s - ym) / i
    end
    return atan2(ym, xm)
END
```

6 Additional Text-based RGB colors

The following color formats express **8-bit-per-color-component *encoded RGB* colors**[3] as text strings:

[3] https://peteroupc.github.io/colorgen.html#RGB_Integer_Formats

- **Delphi** format: Consists of "$00" followed by six base-16 (hexadecimal) digits, two each for the blue, green, and red components, in that order.
- **Visual Basic** format: Consists of "&H" followed by six base-16 digits, two each for the blue, green, and red components, in that order.
- **C++** format: Consists of "0x00" followed by six base-16 digits, two each for the three RGB components. If the format expresses a Windows `COLORREF` color, the three components are blue, green, and red, in that order.
- **PowerBuilder** format: Consists of the integer form of the 8-bit-per-color-component format color, packed red/green/blue, in that order from lowest to highest bits.

7 Additional Color Models

7.1 HSI

A color following the HSI color model consists of three components, in the following order:

- *Hue* has the same general meaning as HSV hue, but is calculated differently.
- A component called "saturation" is 0 or greater and 1 or less.
- A component called "intensity", the average of the red, green, and blue components, is 0 or greater and 1 or less.

The conversions given below are independent of RGB color space, but should be done using ***linear RGB colors***[4].

```
METHOD RgbToHsi(rgb)
    sum=rgb[0]+rgb[1]+rgb[2]
    if sum==0: return [0,0,0]
    r=rgb[0]*1.0/sum
    g=rgb[1]*1.0/sum
    b=rgb[2]*1.0/sum
    coshue=(2*r-g-b)/(2*sqrt((b-g)*(b-r)+(g-r)*(g-r)))
    hue=atan2(sqrt(1-coshue*coshue),coshue)
    if b>g: hue=2*pi-hue
    return [hue, 1-min(r,g,b)*3, sum/3.0]
END METHOD

METHOD HsiToRgb(hsi)
   h=hsi[0]
   if h < 0: h = pi * 2 - rem(-h, pi * 2)
   if h >= pi * 2: h = rem(h, pi * 2)
   deg120=2*pi/3
   hmod=rem(h, deg120)
   a=hsi[2]*(1-hsi[1])
   b=(hsi[1]*cos(hmod)/sin(hmod+pi/6)+1)*hsi[2]
   c=3*hsi[2]-a-b
   if h>=deg120 and h < deg120*2: return [a,b,c]
   if h>=deg120*2: return [c,a,b]
   return [b,c,a]
END METHOD
```

[4] https://peteroupc.github.io/colorgen.html#RGB_Color_Spaces

7.2 Hunter L,a,b

The conversion between XYZ and Hunter L,a,b colors is as given later.

```
METHOD HunterLabFromXYZ(xyz, wpoint)
    x=xyz[0]/wpoint[0]
    y=xyz[1]/wpoint[1]
    z=xyz[2]/wpoint[2]
    l=100*sqrt(y)
    if l==0: return [0,0,0]
    a=(7*sqrt(102)*sqrt(wpoint[0]/y)*(x-wpoint[0]*y))/(4*wpoint[0])
    b=(77*sqrt(70)*sqrt(wpoint[2]/y)*(wpoint[2]*y-z))/(100*wpoint[2])
    return [l,a,b]
END METHOD

METHOD HunterLabToXYZ(lab, wpoint)
    y=lab[0]*lab[0]/10000.0
    if y==0: return [0,0,0]
    x=2*sqrt(102)*lab[1]*wpoint[0]/(357*sqrt(wpoint[0]/y))+wpoint[0]*y
    z=-10*sqrt(70)*lab[1]*wpoint[2]/(539*sqrt(wpoint[2]/y))+wpoint[2]*y
    return [x,y/wpoint[1],z]
END METHOD
```

The `LabToHue`, `LabToChroma`, `LabHueDifference`, `LabChromaHueDifference`, and `LchToLab` methods from the **discussion on CIELAB colors** work with Hunter L, a, b colors analogously to CIELAB colors.

The difference in lightness, a, b, or chroma (ΔL, Δa, Δb, or ΔC, respectively), between two Hunter L, a, b colors is simply the difference between the corresponding value of the second Hunter L, a, b color and that of the first.

8 Additional Color Formulas

CIE94. The following pseudocode implements the color difference formula published in 1994 by the CIE, called CIE94 or ΔE^*_{94}, between two **CIELAB**[5] colors. Note that in this formula, the order of the two colors is important (the first color is the reference, and the second color is the test). In the pseudocode below, `TEXTILES` is `true` for a color difference suitable for textile applications, and `false` otherwise.

```
METHOD COLORDIFF(lab1, lab2)
    c1=LabToChroma(lab1)
    c2=LabToChroma(lab2)
    dl=1
    dc=1+0.045*c1
    dh=1+0.015*c1
    if TEXTILES
            dl=2
            dc=1+0.048*c1
            dh=1+0.014*c1
    end
    da=lab2[1]-lab1[1]
    db=lab2[2]-lab1[2]
    dchr=c2-c1
    dhue=sqrt(max(0,da*da+db*db-dchr*dchr))
```

[5] https://peteroupc.github.io/colorgen.html#CIELAB

```
    dl=((lab2[0]-lab1[0])/dl)
    dc=(dchr/dc)
    dh=(dhue/dh)
    return sqrt(dl*dl+dc*dc+dh*dh)
END METHOD
```

9 Terminal Graphics

Some command-line terminals (or terminal emulators) support coloring the background or foreground of text. In such programs that support **"ANSI" (American National Standards Institute) graphics codes**[6] (generally in the category "select graphic rendition", or SGR), the sequence U+001B (escape character) followed by "[" followed by a semicolon-separated sequence of numbers (given later) followed by "m" is a graphic control sequence (see also Ecma-048, sec. 8.3.117):

- "0": Reset the foreground and background color and other graphic properties to default. (The graphic control sequence U+001B followed by "[m" has the same effect.)
- "1": Set the following text in bold type.
- "2": Use a slightly dimmer foreground color than usual.
- "3": Set the following text in italic type.
- "4": Underline the following text.
- "7": Reverse the meaning of "foreground" and "background" in the following text.
- "8": Hide text while still taking up space.
- "21", "22", "23", "24", "27", "28": Turns off the feature mentioned earlier in "1", "2", "3", "4", "7", or "8", respectively.
- "3" followed by one of the *color numbers* below: Dimmer version of foreground color.
- "4" followed by color number: Dimmer version of background color.
- "9" followed by color number: Brighter version of foreground color.
- "10" followed by color number: Brighter version of background color.

The *color number* is one of the following (only eight colors were defined by ANSI X3.64): "0" (black), "1" (red), "2" (green), "3" (yellow), "4" (blue), "5" (magenta), "6" (cyan), or "7" (white). Note that not all terminals or terminal emulators support all the SGR codes given here, and that the exact colors named by each color number can vary with the implementation.

10 Color Measurement Devices

Measuring color is not like pointing and shooting with a camera, and it's not like measuring height or weight. In general, special devices or technologies are needed to measure color.

There are two general kinds of color measurement devices: *colorimeters* and *spectrophotometers*. In general:

- A *colorimeter* detects light passing through a small number of special filters, and converts the light detected this way to numbers. These numbers are usually three *tristimulus values* that identify a particular color.
- A *spectrophotometer* breaks light like a prism into many wavelength bands, detects these bands, and converts each band into a number. These numbers together form a ***spectral curve***[7].

Color measurements of the same sample can vary depending on many things, including—

- how the sample is prepared,
- how the sample is presented to the measurement device,

[6] https://en.wikipedia.org/wiki/ANSI_escape_code
[7] https://peteroupc.github.io/colorgen.html#Spectral_Color_Functions

- for opaque samples, whether that device includes gloss in the measurement (as in "sphere" or "diffuse/8°" devices) or excludes it (as in "45/0" or "0/45" devices),
- for nonopaque samples, whether the device measures light passing through the sample in all directions (total transmission) or straight-on only (regular transmission),
- how the measurement device illuminates the sample and filters the light in the device's path (especially if it's a colorimeter or the sample is fluorescent),
- the measurement device's aperture (sample view area),
- ambient temperature and relative humidity, and
- for spectrophotometers, the wavelength range, resolution, and bandwidth of measurement.

Several **application notes**[8] by HunterLab (AN 1018, AN 1031, AN 1033) provide more detailed information. Color measurements should also be reproducible, but how to ensure this is outside the scope of this section.

At the time of this writing, most color measurement devices are still expensive and mostly for professional use. However, **several colorimeters**[9] are available in the consumer market, as is a limited selection of spectrophotometers. G. W. Gill describes a **selection** of color measurement devices.

11 Irrelevant Topics

The following topics on color are rarely relevant to programmers:

- The psychology, symbolism, or "meaning" of colors, since they vary from culture to culture and can change over time, even within the same culture.
- Language differences in color lexicons, for the same reason.
- "Color forecasting", or predicting which colors will be in high demand, especially in the fashion and design realms.

12 License

[8] https://www.hunterlab.com/application-notes.html
[9] https://www.coltechcon.com/publication/overview-low-cost-color-capture-devices/
[10] https://creativecommons.org/publicdomain/zero/1.0/