GrapeFruit Documentation

Release 0.1a3

Xavier Basty <xbasty@gmail.com>

March 15, 2015

Contents

Welcome! This is the documentation for GrapeFruit 0.1a3, last updated March 15, 2015.

See the *genindex* for a list of the topics.

class `grapefruit.Color`

The grapefruit module contains only the `Color` class, which exposes all the functionnalities. It can be used to store a color value and manipulate it, or convert it to another color system.

If you are only interested in converting you colors from one system to another, you can store them using regular tuples instead of `Color` instances. You can then use the class static methods to perform the conversions.

`Color` stores both the RGB and HSL representation of the color. This makes possible to keep the hue intact when the color is a pure white due to its lightness. However, certain operations work only with the RGB values, and might then lose the hue.

All the operations assume that you provide values in the specified ranges, no checks are made whatsoever. If you provide a value outside of the specified ranges, you'll get some strange results...

The class instances are immutable, all the methods return a new instance of the `Color` class, and all the properties are read-only.

Note: Some operations may provide results a bit outside the specified ranges, the results are not capped. This is due to certain color systems having a widers gamut than others.

CHAPTER 1

Class content

CHAPTER 2

Example usage

To create an instance of the grapefruit.Color from RGB values:

```
>>> import grapefruit
>>> r, g, b = 1, 0.5, 0
>>> col = grapefruit.Color.NewFromRgb(r, g, b)
```

To get the values of the color in another colorspace:

```
>>> h, s, v = col.hsv
>>> l, a, b = col.lab
```

To get the complementary of a color:

```
>>> compl = col.ComplementaryColor()
>>> print compl.hsl
(210.0, 1.0, 0.5)
```

To directly convert RGB values to their HSL equivalent:

```
>>> h, s, l = Color.RgbToHsl(r, g, b)
```

CHAPTER 3

Class Constants

`Color.`**`WHITE_REFERENCE`**

The reference white points of the CIE standards illuminants, calculated from the chromaticity coordinates found at: http://en.wikipedia.org/wiki/Standard_illuminant

A dictionary mapping the name of the CIE standard illuminants to their reference white points. The white points are required for the XYZ <-> L*a*b conversions.

The key names are build using the following pattern: `<observer>_<illuminant>`

The possible values for `<observer>` are:

Value	Observer
std	CIE 1931 2° Standard Observer
sup	CIE 1964 10° Supplementary Observer

The possible values for `<illuminant>` are the name of the standard illuminants:

Value	CCT	Illuminant
A	2856 K	Incandescent tungsten
B	4874 K	Direct sunlight at noon (obsolete)
C	6774 K	North sky daylight (obsolete)
D50	5003 K	ICC Profile PCS. Horizon light.
D55	5503 K	Compromise between incandescent and daylight
D65	6504 K	Noon daylight (TV & sRGB colorspace)
D75	7504 K	North sky day light
E	~5455 K	Equal energy radiator (not a black body)
F1	6430 K	Daylight Fluorescent
F2	4230 K	Cool White Fluorescent
F3	3450 K	White Fluorescent
F4	2940 K	Warm White Fluorescent
F5	6350 K	Daylight Fluorescent
F6	4150 K	Lite White Fluorescent
F7	6500 K	Broadband fluorescent, D65 simulator
F8	5000 K	Broadband fluorescent, D50 simulator
F9	4150 K	Broadband fluorescent, Cool White Deluxe
F10	5000 K	Narrowband fluorescent, Philips TL85, Ultralume 50
F11	4000 K	Narrowband fluorescent, Philips TL84, Ultralume 40
F12	3000 K	Narrowband fluorescent, Philips TL83, Ultralume 30

`Color.`**`NAMED_COLOR`**

The names and RGB values of the X11 colors supported by popular browsers, with the gray/grey spelling issues, fixed so that both work (e.g light*grey* and light*gray*).

Note: For *Gray*, *Green*, *Maroon* and *Purple*, the HTML/CSS values are used instead of the X11 ones (see X11/CSS clashes)

Reference: CSS3 Color module

CHAPTER 4

Conversion functions

The conversion functions are static methods of the `Color` class that let you convert a color stored as the list of its components rather than as a `Color` instance.

CHAPTER 5

Instantiation functions

The instantiation functions let you create a new instance of the `Color` class from the color components using the color system of your choice.

CHAPTER 6

Properties

The properties get the value of the instance in the specified color model.

The properties returning calculated values unless marked otherwise.

Note: All the properties are read-only. You need to make a copy of the instance to modify the color value.

`Color.`**`greyscale`**

CHAPTER 7

Manipulation methods

The manipulations methods let you create a new color by changing an existing color properties.

Note: The methods **do not** modify the current Color instance. They create a new instance or a tuple of new instances with the specified modifications.

CHAPTER 8

Generation methods

The generation methods let you create a color scheme by using a color as the start point.

All the method, appart from Gradient and MonochromeScheme, have a 'mode' parameter that let you choose which color wheel should be used to generate the scheme.

The following modes are available:

ryb The RYB color wheel, or *artistic color wheel*. While scientifically incorrect, it generally produces better schemes than RGB.

rgb The standard RGB color wheel.

CHAPTER 9

Blending methods

g

C

G