color

package | standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 0 | Imported by: 17,769

Details Valid go.mod file | Redistributable license | Tagged version | Stable version

Learn more

Repository cs.opensource.google/go/go

Links Report a Vulnerability

Documentation

Documentation

Overview

Package color implements a basic color library.

Index

Constants

This section is empty.

Variables

View Source

```
var (
    Black       = Gray16{0}
    White       = Gray16{0xffff}
    Transparent = Alpha16{0}
    Opaque      = Alpha16{0xffff}
)
```

Standard colors.

Functions

func CMYKToRGB

added in go1.5

```
func CMYKToRGB(c, m, y, k uint8) (uint8, uint8, uint8)
```

CMYKToRGB converts a CMYK quadruple to an RGB triple.

func RGBToCMYK

added in go1.5

```
func RGBToCMYK(r, g, b uint8) (uint8, uint8, uint8, uint8)
```

RGBToCMYK converts an RGB triple to a CMYK quadruple.

func RGBToYCbCr

```
func RGBToYCbCr(r, g, b uint8) (uint8, uint8, uint8)
```

RGBToYCbCr converts an RGB triple to a Y'CbCr triple.

func YCbCrToRGB

```
func YCbCrToRGB(y, cb, cr uint8) (uint8, uint8, uint8)
```

YCbCrToRGB converts a Y'CbCr triple to an RGB triple.

Types

type Alpha

```
type Alpha struct {
    A uint8
}
```

Alpha represents an 8-bit alpha color.

func (Alpha) RGBA

```
func (c Alpha) RGBA() (r, g, b, a uint32)
```

type Alpha16

```
type Alpha16 struct {
    A uint16
}
```

Alpha16 represents a 16-bit alpha color.

func (Alpha16) RGBA

```
func (c Alpha16) RGBA() (r, g, b, a uint32)
```

type CMYK

added in go1.5

```
type CMYK struct {
    C, M, Y, K uint8
}
```

CMYK represents a fully opaque CMYK color, having 8 bits for each of cyan, magenta, yellow and black.

It is not associated with any particular color profile.

func (CMYK) RGBA

added in go1.5

```
func (c CMYK) RGBA() (uint32, uint32, uint32, uint32)
```

type Color

```
type Color interface {
    // RGBA returns the alpha-premultiplied red, green, blue and alpha values
    // for the color. Each value ranges within [0, 0xffff], but is represented
    // by a uint32 so that multiplying by a blend factor up to 0xffff will not
    // overflow.
    //
    // An alpha-premultiplied color component c has been scaled by alpha (a),
    // so has valid values 0 <= c <= a.
    RGBA() (r, g, b, a uint32)
}
```

Color can convert itself to alpha-premultiplied 16-bits per channel RGBA. The conversion may be lossy.

type Gray

```
type Gray struct {
    Y uint8
}
```

Gray represents an 8-bit grayscale color.

func (Gray) RGBA

```
func (c Gray) RGBA() (r, g, b, a uint32)
```

type Gray16

```
type Gray16 struct {
    Y uint16
}
```

Gray16 represents a 16-bit grayscale color.

func (Gray16) RGBA

```
func (c Gray16) RGBA() (r, g, b, a uint32)
```

type Model

```
type Model interface {
    Convert(c Color) Color
}
```

Model can convert any Color to one from its own color model. The conversion may be lossy.

```
var (
    RGBAModel    Model = ModelFunc(rgbaModel)
    RGBA64Model  Model = ModelFunc(rgba64Model)
    NRGBAModel   Model = ModelFunc(nrgbaModel)
    NRGBA64Model Model = ModelFunc(nrgba64Model)
    AlphaModel   Model = ModelFunc(alphaModel)
    Alpha16Model Model = ModelFunc(alpha16Model)
    GrayModel    Model = ModelFunc(grayModel)
    Gray16Model  Model = ModelFunc(gray16Model)
)
```

Models for the standard color types.

```
var CMYKModel Model = ModelFunc(cmykModel)
```

CMYKModel is the Model for CMYK colors.

```
var NYCbCrAModel Model = ModelFunc(nYCbCrAModel)
```

NYCbCrAModel is the Model for non-alpha-premultiplied Y'CbCr-with-alpha colors.

```
var YCbCrModel Model = ModelFunc(yCbCrModel)
```

YCbCrModel is the Model for Y'CbCr colors.

func ModelFunc

```
func ModelFunc(f func(Color) Color) Model
```

ModelFunc returns a Model that invokes f to implement the conversion.

type NRGBA

```
type NRGBA struct {
    R, G, B, A uint8
}
```

NRGBA represents a non-alpha-premultiplied 32-bit color.

func (NRGBA) RGBA

```
func (c NRGBA) RGBA() (r, g, b, a uint32)
```

type NRGBA64

```
type NRGBA64 struct {
    R, G, B, A uint16
```

```
}
```

NRGBA64 represents a non-alpha-premultiplied 64-bit color, having 16 bits for each of red, green, blue and alpha.

func (NRGBA64) RGBA

```
func (c NRGBA64) RGBA() (r, g, b, a uint32)
```

type NYCbCrA

added in go1.6

```
type NYCbCrA struct {
    YCbCr
    A uint8
}
```

NYCbCrA represents a non-alpha-premultiplied Y'CbCr-with-alpha color, having 8 bits each for one luma, two chroma and one alpha component.

func (NYCbCrA) RGBA

added in go1.6

```
func (c NYCbCrA) RGBA() (uint32, uint32, uint32, uint32)
```

type Palette

```
type Palette []Color
```

Palette is a palette of colors.

func (Palette) Convert

```
func (p Palette) Convert(c Color) Color
```

Convert returns the palette color closest to c in Euclidean R,G,B space.

func (Palette) Index

```
func (p Palette) Index(c Color) int
```

Index returns the index of the palette color closest to c in Euclidean R,G,B,A space.

type RGBA

```
type RGBA struct {
    R, G, B, A uint8
}
```

RGBA represents a traditional 32-bit alpha-premultiplied color, having 8 bits for each of red, green, blue and alpha.

An alpha-premultiplied color component C has been scaled by alpha (A), so has valid values 0 <= C <= A.

func (RGBA) RGBA

```
func (c RGBA) RGBA() (r, g, b, a uint32)
```

type RGBA64

```
type RGBA64 struct {
    R, G, B, A uint16
}
```

RGBA64 represents a 64-bit alpha-premultiplied color, having 16 bits for each of red, green, blue and alpha.

An alpha-premultiplied color component C has been scaled by alpha (A), so has valid values 0 <= C <= A.

func (RGBA64) RGBA

```
func (c RGBA64) RGBA() (r, g, b, a uint32)
```

type YCbCr

```
type YCbCr struct {
    Y, Cb, Cr uint8
}
```

YCbCr represents a fully opaque 24-bit Y'CbCr color, having 8 bits each for one luma and two chroma components.

JPEG, VP8, the MPEG family and other codecs use this color model. Such codecs often use the terms YUV and Y'CbCr interchangeably, but strictly speaking, the term YUV applies only to analog video signals, and Y' (luma) is Y (luminance) after applying gamma correction.

Conversion between RGB and Y'CbCr is lossy and there are multiple, slightly different formulae for converting between the two. This package follows the JFIF specification at https://www.w3.org/Graphics/JPEG/jfif3.pdf.

func (YCbCr) RGBA

```
func (c YCbCr) RGBA() (uint32, uint32, uint32, uint32)
```

Source Files

View all

color.go ycbcr.go

Directories

palette

Package palette provides standard color palettes.

Report an Issue