

[Discover Packages](#) > [Standard library](#) > [image](#) 

image





package

standard library

Version: [go1.20.1](#) **Latest** | Published: Feb 14, 2023 | License: [BSD-3-Clause](#) | Imports: 8 |

Imported by: 25,429

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 image implements a basic 2-D image library.

The fundamental interface is called Image. An Image contains colors, which are described in the image/color package.

Values of the Image interface are created either by calling functions such as NewRGBA and NewPaletted, or by calling Decode on an io.Reader containing image data in a format such as GIF, JPEG or PNG. Decoding any particular image format requires the prior registration of a decoder function. Registration is typically automatic as a side effect of initializing that format's package so that, to decode a PNG image, it suffices to have

```
import _ "image/png"
```

in a program's main package. The `_` means to import a package purely for its initialization side effects.

See "The Go image package" for more details: https://golang.org/doc/articles/image_package.html

[▶ Example](#)[▶ Example \(DecodeConfig\)](#)

Index

Variables

```
func RegisterFormat(name, magic string, decode func(io.Reader) (Image, error), ...)
```

type Alpha

```
func NewAlpha(r Rectangle) *Alpha
func (p *Alpha) AlphaAt(x, y int) color.Alpha
func (p *Alpha) At(x, y int) color.Color
func (p *Alpha) Bounds() Rectangle
func (p *Alpha) ColorModel() color.Model
func (p *Alpha) Opaque() bool
func (p *Alpha) PixOffset(x, y int) int
func (p *Alpha) RGBA64At(x, y int) color.RGBA64
func (p *Alpha) Set(x, y int, c color.Color)
func (p *Alpha) SetAlpha(x, y int, c color.Alpha)
func (p *Alpha) SetRGBA64(x, y int, c color.RGBA64)
func (p *Alpha) SubImage(r Rectangle) Image
```

type Alpha16

```
func NewAlpha16(r Rectangle) *Alpha16
func (p *Alpha16) Alpha16At(x, y int) color.Alpha16
func (p *Alpha16) At(x, y int) color.Color
func (p *Alpha16) Bounds() Rectangle
func (p *Alpha16) ColorModel() color.Model
func (p *Alpha16) Opaque() bool
func (p *Alpha16) PixOffset(x, y int) int
func (p *Alpha16) RGBA64At(x, y int) color.RGBA64
func (p *Alpha16) Set(x, y int, c color.Color)
func (p *Alpha16) SetAlpha16(x, y int, c color.Alpha16)
func (p *Alpha16) SetRGBA64(x, y int, c color.RGBA64)
func (p *Alpha16) SubImage(r Rectangle) Image
```

type CMYK

```
func NewCMYK(r Rectangle) *CMYK
func (p *CMYK) At(x, y int) color.Color
func (p *CMYK) Bounds() Rectangle
func (p *CMYK) CMYKAt(x, y int) color.CMYK
func (p *CMYK) ColorModel() color.Model
func (p *CMYK) Opaque() bool
func (p *CMYK) PixOffset(x, y int) int
func (p *CMYK) RGBA64At(x, y int) color.RGBA64
func (p *CMYK) Set(x, y int, c color.Color)
func (p *CMYK) SetCMYK(x, y int, c color.CMYK)
func (p *CMYK) SetRGBA64(x, y int, c color.RGBA64)
func (p *CMYK) SubImage(r Rectangle) Image
```

type Config

```
func DecodeConfig(r io.Reader) (Config, string, error)
```

type Gray

```
func NewGray(r Rectangle) *Gray
func (p *Gray) At(x, y int) color.Color
```

```
func (p *Gray) Bounds() Rectangle
func (p *Gray) ColorModel() color.Model
func (p *Gray) GrayAt(x, y int) color.Gray
func (p *Gray) Opaque() bool
func (p *Gray) PixOffset(x, y int) int
func (p *Gray) RGBA64At(x, y int) color.RGBA64
func (p *Gray) Set(x, y int, c color.Color)
func (p *Gray) SetGray(x, y int, c color.Gray)
func (p *Gray) SetRGBA64(x, y int, c color.RGBA64)
func (p *Gray) SubImage(r Rectangle) Image
```

type Gray16

```
func NewGray16(r Rectangle) *Gray16
func (p *Gray16) At(x, y int) color.Color
func (p *Gray16) Bounds() Rectangle
func (p *Gray16) ColorModel() color.Model
func (p *Gray16) Gray16At(x, y int) color.Gray16
func (p *Gray16) Opaque() bool
func (p *Gray16) PixOffset(x, y int) int
func (p *Gray16) RGBA64At(x, y int) color.RGBA64
func (p *Gray16) Set(x, y int, c color.Color)
func (p *Gray16) SetGray16(x, y int, c color.Gray16)
func (p *Gray16) SetRGBA64(x, y int, c color.RGBA64)
func (p *Gray16) SubImage(r Rectangle) Image
```

type Image

```
func Decode(r io.Reader) (Image, string, error)
```

type NRGBA

```
func NewNRGBA(r Rectangle) *NRGBA
func (p *NRGBA) At(x, y int) color.Color
func (p *NRGBA) Bounds() Rectangle
func (p *NRGBA) ColorModel() color.Model
func (p *NRGBA) NRGBAAt(x, y int) color.NRGBA
func (p *NRGBA) Opaque() bool
func (p *NRGBA) PixOffset(x, y int) int
func (p *NRGBA) RGBA64At(x, y int) color.RGBA64
func (p *NRGBA) Set(x, y int, c color.Color)
func (p *NRGBA) SetNRGBA(x, y int, c color.NRGBA)
func (p *NRGBA) SetRGBA64(x, y int, c color.RGBA64)
func (p *NRGBA) SubImage(r Rectangle) Image
```

type NRGBA64

```
func NewNRGBA64(r Rectangle) *NRGBA64
func (p *NRGBA64) At(x, y int) color.Color
func (p *NRGBA64) Bounds() Rectangle
func (p *NRGBA64) ColorModel() color.Model
func (p *NRGBA64) NRGBA64At(x, y int) color.NRGBA64
func (p *NRGBA64) Opaque() bool
func (p *NRGBA64) PixOffset(x, y int) int
```

```
func (p *NRGBA64) RGBA64At(x, y int) color.RGBA64
func (p *NRGBA64) Set(x, y int, c color.Color)
func (p *NRGBA64) SetNRGBA64(x, y int, c color.NRGBA64)
func (p *NRGBA64) SetRGBA64(x, y int, c color.RGBA64)
func (p *NRGBA64) SubImage(r Rectangle) Image
```

type NYCbCrA

```
func NewNYCbCrA(r Rectangle, subsampleRatio YCbCrSubsampleRatio) *NYCbCrA
func (p *NYCbCrA) AOffset(x, y int) int
func (p *NYCbCrA) At(x, y int) color.Color
func (p *NYCbCrA) ColorModel() color.Model
func (p *NYCbCrA) NYCbCrAAt(x, y int) color.NYCbCrA
func (p *NYCbCrA) Opaque() bool
func (p *NYCbCrA) RGBA64At(x, y int) color.RGBA64
func (p *NYCbCrA) SubImage(r Rectangle) Image
```

type Paletted

```
func NewPaletted(r Rectangle, p color.Palette) *Paletted
func (p *Paletted) At(x, y int) color.Color
func (p *Paletted) Bounds() Rectangle
func (p *Paletted) ColorIndexAt(x, y int) uint8
func (p *Paletted) ColorModel() color.Model
func (p *Paletted) Opaque() bool
func (p *Paletted) PixOffset(x, y int) int
func (p *Paletted) RGBA64At(x, y int) color.RGBA64
func (p *Paletted) Set(x, y int, c color.Color)
func (p *Paletted) SetColorIndex(x, y int, index uint8)
func (p *Paletted) SetRGBA64(x, y int, c color.RGBA64)
func (p *Paletted) SubImage(r Rectangle) Image
```

type PalettedImage

type Point

```
func Pt(X, Y int) Point
func (p Point) Add(q Point) Point
func (p Point) Div(k int) Point
func (p Point) Eq(q Point) bool
func (p Point) In(r Rectangle) bool
func (p Point) Mod(r Rectangle) Point
func (p Point) Mul(k int) Point
func (p Point) String() string
func (p Point) Sub(q Point) Point
```

type RGBA

```
func NewRGBA(r Rectangle) *RGBA
func (p *RGBA) At(x, y int) color.Color
func (p *RGBA) Bounds() Rectangle
func (p *RGBA) ColorModel() color.Model
func (p *RGBA) Opaque() bool
func (p *RGBA) PixOffset(x, y int) int
func (p *RGBA) RGBA64At(x, y int) color.RGBA64
```

```
func (p *RGBA) RGBAAt(x, y int) color.RGBA
func (p *RGBA) Set(x, y int, c color.Color)
func (p *RGBA) SetRGBA(x, y int, c color.RGBA)
func (p *RGBA) SetRGBA64(x, y int, c color.RGBA64)
func (p *RGBA) SubImage(r Rectangle) Image
```

type RGBA64

```
func NewRGBA64(r Rectangle) *RGBA64
func (p *RGBA64) At(x, y int) color.Color
func (p *RGBA64) Bounds() Rectangle
func (p *RGBA64) ColorModel() color.Model
func (p *RGBA64) Opaque() bool
func (p *RGBA64) PixOffset(x, y int) int
func (p *RGBA64) RGBA64At(x, y int) color.RGBA64
func (p *RGBA64) Set(x, y int, c color.Color)
func (p *RGBA64) SetRGBA64(x, y int, c color.RGBA64)
func (p *RGBA64) SubImage(r Rectangle) Image
```

type RGBA64Image

type Rectangle

```
func Rect(x0, y0, x1, y1 int) Rectangle
func (r Rectangle) Add(p Point) Rectangle
func (r Rectangle) At(x, y int) color.Color
func (r Rectangle) Bounds() Rectangle
func (r Rectangle) Canon() Rectangle
func (r Rectangle) ColorModel() color.Model
func (r Rectangle) Dx() int
func (r Rectangle) Dy() int
func (r Rectangle) Empty() bool
func (r Rectangle) Eq(s Rectangle) bool
func (r Rectangle) In(s Rectangle) bool
func (r Rectangle) Inset(n int) Rectangle
func (r Rectangle) Intersect(s Rectangle) Rectangle
func (r Rectangle) Overlaps(s Rectangle) bool
func (r Rectangle) RGBA64At(x, y int) color.RGBA64
func (r Rectangle) Size() Point
func (r Rectangle) String() string
func (r Rectangle) Sub(p Point) Rectangle
func (r Rectangle) Union(s Rectangle) Rectangle
```

type Uniform

```
func NewUniform(c color.Color) *Uniform
func (c *Uniform) At(x, y int) color.Color
func (c *Uniform) Bounds() Rectangle
func (c *Uniform) ColorModel() color.Model
func (c *Uniform) Convert(color.Color) color.Color
func (c *Uniform) Opaque() bool
func (c *Uniform) RGBA() (r, g, b, a uint32)
func (c *Uniform) RGBA64At(x, y int) color.RGBA64
```

type YCbCr

```
func NewYCbCr(r Rectangle, subsampleRatio YCbCrSubsampleRatio) *YCbCr
```

```
func (p *YCbCr) At(x, y int) color.Color
```

```
func (p *YCbCr) Bounds() Rectangle
```

```
func (p *YCbCr) COffset(x, y int) int
```

```
func (p *YCbCr) ColorModel() color.Model
```

```
func (p *YCbCr) Opaque() bool
```

```
func (p *YCbCr) RGBA64At(x, y int) color.RGBA64
```

```
func (p *YCbCr) SubImage(r Rectangle) Image
```

```
func (p *YCbCr) YCbCrAt(x, y int) color.YCbCr
```

```
func (p *YCbCr) YOffset(x, y int) int
```

type YCbCrSubsampleRatio

```
func (s YCbCrSubsampleRatio) String() string
```

Examples

Package

Package (DecodeConfig)

Constants

This section is empty.

Variables

[View Source](#)

```
var (  
    // Black is an opaque black uniform image.  
    Black = NewUniform(color.Black)  
    // White is an opaque white uniform image.  
    White = NewUniform(color.White)  
    // Transparent is a fully transparent uniform image.  
    Transparent = NewUniform(color.Transparent)  
    // Opaque is a fully opaque uniform image.  
    Opaque = NewUniform(color.Opaque)  
)
```

[View Source](#)

```
var ErrFormat = errors.New("image: unknown format")
```

ErrFormat indicates that decoding encountered an unknown format.

Functions

func RegisterFormat

```
func RegisterFormat(name, magic string, decode func(io.Reader) (Image, error), decodeC  
onfig func(io.Reader) (Config, error))
```

RegisterFormat registers an image format for use by Decode. Name is the name of the format, like "jpeg" or "png". Magic is the magic prefix that identifies the format's encoding. The magic string can contain "?" wildcards that each match any one byte. Decode is the function that decodes the encoded image. DecodeConfig is the function that decodes just its configuration.

Types

type Alpha

```
type Alpha struct {  
    // Pix holds the image's pixels, as alpha values. The pixel at  
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*1].  
    Pix []uint8  
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.  
    Stride int  
    // Rect is the image's bounds.  
    Rect Rectangle  
}
```

Alpha is an in-memory image whose At method returns color.Alpha values.

func NewAlpha

```
func NewAlpha(r Rectangle) *Alpha
```

NewAlpha returns a new Alpha image with the given bounds.

func (*Alpha) AlphaAt

added in go1.4

```
func (p *Alpha) AlphaAt(x, y int) color.Alpha
```

func (*Alpha) At

```
func (p *Alpha) At(x, y int) color.Color
```

func (*Alpha) Bounds

```
func (p *Alpha) Bounds() Rectangle
```

func (*Alpha) ColorModel

```
func (p *Alpha) ColorModel() color.Model
```

func (*Alpha) Opaque

```
func (p *Alpha) Opaque() bool
```

Opaque scans the entire image and reports whether it is fully opaque.

func (*Alpha) **PixOffset**

```
func (p *Alpha) PixOffset(x, y int) int
```

PixOffset returns the index of the first element of Pix that corresponds to the pixel at (x, y).

func (*Alpha) **RGBA64At**

added in go1.17

```
func (p *Alpha) RGBA64At(x, y int) color.RGBA64
```

func (*Alpha) **Set**

```
func (p *Alpha) Set(x, y int, c color.Color)
```

func (*Alpha) **SetAlpha**

```
func (p *Alpha) SetAlpha(x, y int, c color.Alpha)
```

func (*Alpha) **SetRGBA64**

added in go1.17

```
func (p *Alpha) SetRGBA64(x, y int, c color.RGBA64)
```

func (*Alpha) **SubImage**

```
func (p *Alpha) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type **Alpha16**

```
type Alpha16 struct {  
    // Pix holds the image's pixels, as alpha values in big-endian format. The pixel at  
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*2].  
    Pix []uint8  
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.  
    Stride int  
    // Rect is the image's bounds.  
    Rect Rectangle  
}
```

Alpha16 is an in-memory image whose At method returns color.Alpha16 values.

func **NewAlpha16**

```
func NewAlpha16(r Rectangle) *Alpha16
```


NewAlpha16 returns a new Alpha16 image with the given bounds.

func (*Alpha16) Alpha16At

added in go1.4

```
func (p *Alpha16) Alpha16At(x, y int) color.Alpha16
```

func (*Alpha16) At

```
func (p *Alpha16) At(x, y int) color.Color
```

func (*Alpha16) Bounds

```
func (p *Alpha16) Bounds() Rectangle
```

func (*Alpha16) ColorModel

```
func (p *Alpha16) ColorModel() color.Model
```

func (*Alpha16) Opaque

```
func (p *Alpha16) Opaque() bool
```

Opaque scans the entire image and reports whether it is fully opaque.

func (*Alpha16) PixOffset

```
func (p *Alpha16) PixOffset(x, y int) int
```

PixOffset returns the index of the first element of Pix that corresponds to the pixel at (x, y).

func (*Alpha16) RGBA64At

added in go1.17

```
func (p *Alpha16) RGBA64At(x, y int) color.RGBA64
```

func (*Alpha16) Set

```
func (p *Alpha16) Set(x, y int, c color.Color)
```

func (*Alpha16) SetAlpha16

```
func (p *Alpha16) SetAlpha16(x, y int, c color.Alpha16)
```

func (*Alpha16) SetRGBA64

added in go1.17

```
func (p *Alpha16) SetRGBA64(x, y int, c color.RGBA64)
```

func (*Alpha16) SubImage

```
func (p *Alpha16) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type CMYK

added in go1.5

```
type CMYK struct {  
    // Pix holds the image's pixels, in C, M, Y, K order. The pixel at  
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*4].  
    Pix []uint8  
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.  
    Stride int  
    // Rect is the image's bounds.  
    Rect Rectangle  
}
```

CMYK is an in-memory image whose At method returns color.CMYK values.

func NewCMYK

added in go1.5

```
func NewCMYK(r Rectangle) *CMYK
```

NewCMYK returns a new CMYK image with the given bounds.

func (*CMYK) At

added in go1.5

```
func (p *CMYK) At(x, y int) color.Color
```

func (*CMYK) Bounds

added in go1.5

```
func (p *CMYK) Bounds() Rectangle
```

func (*CMYK) CMYKAt

added in go1.5

```
func (p *CMYK) CMYKAt(x, y int) color.CMYK
```

func (*CMYK) ColorModel

added in go1.5

```
func (p *CMYK) ColorModel() color.Model
```

func (*CMYK) Opaque

added in go1.5

```
func (p *CMYK) Opaque() bool
```

Opaque scans the entire image and reports whether it is fully opaque.

func (*CMYK) **PixOffset**

added in go1.5

```
func (p *CMYK) PixOffset(x, y int) int
```

PixOffset returns the index of the first element of Pix that corresponds to the pixel at (x, y).

func (*CMYK) **RGBA64At**

added in go1.17

```
func (p *CMYK) RGBA64At(x, y int) color.RGBA64
```

func (*CMYK) **Set**

added in go1.5

```
func (p *CMYK) Set(x, y int, c color.Color)
```

func (*CMYK) **SetCMYK**

added in go1.5

```
func (p *CMYK) SetCMYK(x, y int, c color.CMYK)
```

func (*CMYK) **SetRGBA64**

added in go1.17

```
func (p *CMYK) SetRGBA64(x, y int, c color.RGBA64)
```

func (*CMYK) **SubImage**

added in go1.5

```
func (p *CMYK) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type **Config**

```
type Config struct {  
    ColorModel    color.Model  
    Width, Height int  
}
```

Config holds an image's color model and dimensions.

func **DecodeConfig**

```
func DecodeConfig(r io.Reader) (Config, string, error)
```

DecodeConfig decodes the color model and dimensions of an image that has been encoded in a registered format. The string returned is the format name used during format registration. Format registration is typically done by an init function in the codec-specific package.

type **Gray**

```
type Gray struct {  
    // Pix holds the image's pixels, as gray values. The pixel at  
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*1].  
    Pix []uint8  
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.  
    Stride int  
    // Rect is the image's bounds.  
    Rect Rectangle  
}
```

Gray is an in-memory image whose `At` method returns `color.Gray` values.

func **NewGray**

```
func NewGray(r Rectangle) *Gray
```

`NewGray` returns a new `Gray` image with the given bounds.

func (*Gray) **At**

```
func (p *Gray) At(x, y int) color.Color
```

func (*Gray) **Bounds**

```
func (p *Gray) Bounds() Rectangle
```

func (*Gray) **ColorModel**

```
func (p *Gray) ColorModel() color.Model
```

func (*Gray) **GrayAt**

added in go1.4

```
func (p *Gray) GrayAt(x, y int) color.Gray
```

func (*Gray) **Opaque**

```
func (p *Gray) Opaque() bool
```

`Opaque` scans the entire image and reports whether it is fully opaque.

func (*Gray) **PixOffset**

```
func (p *Gray) PixOffset(x, y int) int
```

`PixOffset` returns the index of the first element of `Pix` that corresponds to the pixel at `(x, y)`.

func (*Gray) **RGBA64At**

added in go1.17

```
func (p *Gray) RGBA64At(x, y int) color.RGBA64
```

func (*Gray) **Set**

```
func (p *Gray) Set(x, y int, c color.Color)
```

func (*Gray) **SetGray**

```
func (p *Gray) SetGray(x, y int, c color.Gray)
```

func (*Gray) **SetRGBA64**

added in go1.17

```
func (p *Gray) SetRGBA64(x, y int, c color.RGBA64)
```

func (*Gray) **SubImage**

```
func (p *Gray) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type **Gray16**

```
type Gray16 struct {  
    // Pix holds the image's pixels, as gray values in big-endian format. The pixel at  
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*2].  
    Pix []uint8  
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.  
    Stride int  
    // Rect is the image's bounds.  
    Rect Rectangle  
}
```

Gray16 is an in-memory image whose At method returns color.Gray16 values.

func **NewGray16**

```
func NewGray16(r Rectangle) *Gray16
```

NewGray16 returns a new Gray16 image with the given bounds.

func (*Gray16) **At**

```
func (p *Gray16) At(x, y int) color.Color
```

func (*Gray16) Bounds

```
func (p *Gray16) Bounds() Rectangle
```

func (*Gray16) ColorModel

```
func (p *Gray16) ColorModel() color.Model
```

func (*Gray16) Gray16At

added in go1.4

```
func (p *Gray16) Gray16At(x, y int) color.Gray16
```

func (*Gray16) Opaque

```
func (p *Gray16) Opaque() bool
```

Opaque scans the entire image and reports whether it is fully opaque.

func (*Gray16) PixOffset

```
func (p *Gray16) PixOffset(x, y int) int
```

PixOffset returns the index of the first element of Pix that corresponds to the pixel at (x, y).

func (*Gray16) RGBA64At

added in go1.17

```
func (p *Gray16) RGBA64At(x, y int) color.RGBA64
```

func (*Gray16) Set

```
func (p *Gray16) Set(x, y int, c color.Color)
```

func (*Gray16) SetGray16

```
func (p *Gray16) SetGray16(x, y int, c color.Gray16)
```

func (*Gray16) SetRGBA64

added in go1.17

```
func (p *Gray16) SetRGBA64(x, y int, c color.RGBA64)
```

func (*Gray16) SubImage

```
func (p *Gray16) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type Image

```
type Image interface {
    // ColorModel returns the Image's color model.
    ColorModel() color.Model
    // Bounds returns the domain for which At can return non-zero color.
    // The bounds do not necessarily contain the point (0, 0).
    Bounds() Rectangle
    // At returns the color of the pixel at (x, y).
    // At(Bounds().Min.X, Bounds().Min.Y) returns the upper-left pixel of the grid.
    // At(Bounds().Max.X-1, Bounds().Max.Y-1) returns the lower-right one.
    At(x, y int) color.Color
}
```

Image is a finite rectangular grid of color.Color values taken from a color model.

func Decode

```
func Decode(r io.Reader) (Image, string, error)
```

Decode decodes an image that has been encoded in a registered format. The string returned is the format name used during format registration. Format registration is typically done by an init function in the codec- specific package.

type NRGBA

```
type NRGBA struct {
    // Pix holds the image's pixels, in R, G, B, A order. The pixel at
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*4].
    Pix []uint8
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.
    Stride int
    // Rect is the image's bounds.
    Rect Rectangle
}
```

NRGBA is an in-memory image whose At method returns color.NRGBA values.

func NewNRGBA

```
func NewNRGBA(r Rectangle) *NRGBA
```

NewNRGBA returns a new NRGBA image with the given bounds.

func (*NRGBA) At

```
func (p *NRGBA) At(x, y int) color.Color
```

func (*NRGBA) Bounds

```
func (p *NRGBA) Bounds() Rectangle
```

func (*NRGBA) ColorModel

```
func (p *NRGBA) ColorModel() color.Model
```

func (*NRGBA) NRGBAAt

added in go1.4

```
func (p *NRGBA) NRGBAAt(x, y int) color.NRGBA
```

func (*NRGBA) Opaque

```
func (p *NRGBA) Opaque() bool
```

Opaque scans the entire image and reports whether it is fully opaque.

func (*NRGBA) PixOffset

```
func (p *NRGBA) PixOffset(x, y int) int
```

PixOffset returns the index of the first element of Pix that corresponds to the pixel at (x, y).

func (*NRGBA) RGBA64At

added in go1.17

```
func (p *NRGBA) RGBA64At(x, y int) color.RGBA64
```

func (*NRGBA) Set

```
func (p *NRGBA) Set(x, y int, c color.Color)
```

func (*NRGBA) SetNRGBA

```
func (p *NRGBA) SetNRGBA(x, y int, c color.NRGBA)
```

func (*NRGBA) SetRGBA64

added in go1.17

```
func (p *NRGBA) SetRGBA64(x, y int, c color.RGBA64)
```

func (*NRGBA) SubImage

```
func (p *NRGBA) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type NRGBA64


```

type NRGBA64 struct {
    // Pix holds the image's pixels, in R, G, B, A order and big-endian format. The pixel
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*8].
    Pix []uint8
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.
    Stride int
    // Rect is the image's bounds.
    Rect Rectangle
}

```

NRGBA64 is an in-memory image whose `At` method returns `color.NRGBA64` values.

func NewNRGBA64

```
func NewNRGBA64(r Rectangle) *NRGBA64
```

`NewNRGBA64` returns a new NRGBA64 image with the given bounds.

func (*NRGBA64) At

```
func (p *NRGBA64) At(x, y int) color.Color
```

func (*NRGBA64) Bounds

```
func (p *NRGBA64) Bounds() Rectangle
```

func (*NRGBA64) ColorModel

```
func (p *NRGBA64) ColorModel() color.Model
```

func (*NRGBA64) NRGBA64At

added in go1.4

```
func (p *NRGBA64) NRGBA64At(x, y int) color.NRGBA64
```

func (*NRGBA64) Opaque

```
func (p *NRGBA64) Opaque() bool
```

`Opaque` scans the entire image and reports whether it is fully opaque.

func (*NRGBA64) PixOffset

```
func (p *NRGBA64) PixOffset(x, y int) int
```

`PixOffset` returns the index of the first element of `Pix` that corresponds to the pixel at `(x, y)`.

func (*NRGBA64) RGBA64At

added in go1.17

```
func (p *NRGBA64) RGBA64At(x, y int) color.RGBA64
```

func (*NRGBA64) Set

```
func (p *NRGBA64) Set(x, y int, c color.Color)
```

func (*NRGBA64) SetNRGBA64

```
func (p *NRGBA64) SetNRGBA64(x, y int, c color.NRGBA64)
```

func (*NRGBA64) SetRGBA64

added in go1.17

```
func (p *NRGBA64) SetRGBA64(x, y int, c color.RGBA64)
```

func (*NRGBA64) SubImage

```
func (p *NRGBA64) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type NYCbCrA

added in go1.6

```
type NYCbCrA struct {  
    YCbCr  
    A      []uint8  
    AStride int  
}
```

NYCbCrA is an in-memory image of non-alpha-premultiplied Y'CbCr-with-alpha colors. A and AStride are analogous to the Y and YStride fields of the embedded YCbCr.

func NewNYCbCrA

added in go1.6

```
func NewNYCbCrA(r Rectangle, subsampleRatio YCbCrSubsampleRatio) *NYCbCrA
```

NewNYCbCrA returns a new NYCbCrA image with the given bounds and subsample ratio.

func (*NYCbCrA) AOffset

added in go1.6

```
func (p *NYCbCrA) AOffset(x, y int) int
```

AOffset returns the index of the first element of A that corresponds to the pixel at (x, y).

func (*NYCbCrA) At

added in go1.6

```
func (p *NYCbCrA) At(x, y int) color.Color
```

func (*NYCbCrA) ColorModel

added in go1.6

```
func (p *NYCbCrA) ColorModel() color.Model
```

func (*NYCbCrA) NYCbCrAAt

added in go1.6

```
func (p *NYCbCrA) NYCbCrAAt(x, y int) color.NYCbCrA
```

func (*NYCbCrA) Opaque

added in go1.6

```
func (p *NYCbCrA) Opaque() bool
```

Opaque scans the entire image and reports whether it is fully opaque.

func (*NYCbCrA) RGBA64At

added in go1.17

```
func (p *NYCbCrA) RGBA64At(x, y int) color.RGBA64
```

func (*NYCbCrA) SubImage

added in go1.6

```
func (p *NYCbCrA) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type Paletted

```
type Paletted struct {
    // Pix holds the image's pixels, as palette indices. The pixel at
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*1].
    Pix []uint8
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.
    Stride int
    // Rect is the image's bounds.
    Rect Rectangle
    // Palette is the image's palette.
    Palette color.Palette
}
```

Paletted is an in-memory image of uint8 indices into a given palette.

func NewPaletted

```
func NewPaletted(r Rectangle, p color.Palette) *Paletted
```

NewPaletted returns a new Paletted image with the given width, height and palette.

func (*Paletted) At

```
func (p *Paletted) At(x, y int) color.Color
```

func (*Paletted) Bounds

```
func (p *Paletted) Bounds() Rectangle
```

func (*Paletted) ColorIndexAt

```
func (p *Paletted) ColorIndexAt(x, y int) uint8
```

func (*Paletted) ColorModel

```
func (p *Paletted) ColorModel() color.Model
```

func (*Paletted) Opaque

```
func (p *Paletted) Opaque() bool
```

Opaque scans the entire image and reports whether it is fully opaque.

func (*Paletted) PixOffset

```
func (p *Paletted) PixOffset(x, y int) int
```

PixOffset returns the index of the first element of Pix that corresponds to the pixel at (x, y).

func (*Paletted) RGBA64At

added in go1.17

```
func (p *Paletted) RGBA64At(x, y int) color.RGBA64
```

func (*Paletted) Set

```
func (p *Paletted) Set(x, y int, c color.Color)
```

func (*Paletted) SetColorIndex

```
func (p *Paletted) SetColorIndex(x, y int, index uint8)
```

func (*Paletted) SetRGBA64

added in go1.17

```
func (p *Paletted) SetRGBA64(x, y int, c color.RGBA64)
```

func (*Paletted) SubImage

```
func (p *Paletted) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type PalettedImage

```
type PalettedImage interface {  
    // ColorIndexAt returns the palette index of the pixel at (x, y).  
    ColorIndexAt(x, y int) uint8  
    Image  
}
```

PalettedImage is an image whose colors may come from a limited palette. If m is a PalettedImage and m.ColorModel() returns a color.Palette p, then m.At(x, y) should be equivalent to p[m.ColorIndexAt(x, y)]. If m's color model is not a color.Palette, then ColorIndexAt's behavior is undefined.

type Point

```
type Point struct {  
    X, Y int  
}
```

A Point is an X, Y coordinate pair. The axes increase right and down.

```
var ZP Point
```

ZP is the zero Point.

Deprecated: Use a literal image.Point{} instead.

func Pt

```
func Pt(X, Y int) Point
```

Pt is shorthand for Point{X, Y}.

func (Point) Add

```
func (p Point) Add(q Point) Point
```

Add returns the vector p+q.

func (Point) Div

```
func (p Point) Div(k int) Point
```

Div returns the vector p/k.

func (Point) Eq

```
func (p Point) Eq(q Point) bool
```

Eq reports whether p and q are equal.

func (Point) In

```
func (p Point) In(r Rectangle) bool
```

In reports whether p is in r.

func (Point) Mod

```
func (p Point) Mod(r Rectangle) Point
```

Mod returns the point q in r such that p.X-q.X is a multiple of r's width and p.Y-q.Y is a multiple of r's height.

func (Point) Mul

```
func (p Point) Mul(k int) Point
```

Mul returns the vector p*k.

func (Point) String

```
func (p Point) String() string
```

String returns a string representation of p like "(3,4)".

func (Point) Sub

```
func (p Point) Sub(q Point) Point
```

Sub returns the vector p-q.

type RGBA

```
type RGBA struct {  
    // Pix holds the image's pixels, in R, G, B, A order. The pixel at  
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*4].  
    Pix []uint8  
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.  
    Stride int  
    // Rect is the image's bounds.  
    Rect Rectangle  
}
```

RGBA is an in-memory image whose `At` method returns `color.RGBA` values.

func `NewRGBA`

```
func NewRGBA(r Rectangle) *RGBA
```

`NewRGBA` returns a new `RGBA` image with the given bounds.

func (`*RGBA`) `At`

```
func (p *RGBA) At(x, y int) color.Color
```

func (`*RGBA`) `Bounds`

```
func (p *RGBA) Bounds() Rectangle
```

func (`*RGBA`) `ColorModel`

```
func (p *RGBA) ColorModel() color.Model
```

func (`*RGBA`) `Opaque`

```
func (p *RGBA) Opaque() bool
```

`Opaque` scans the entire image and reports whether it is fully opaque.

func (`*RGBA`) `PixOffset`

```
func (p *RGBA) PixOffset(x, y int) int
```

`PixOffset` returns the index of the first element of `Pix` that corresponds to the pixel at (x, y).

func (`*RGBA`) `RGBA64At`

added in go1.17

```
func (p *RGBA) RGBA64At(x, y int) color.RGBA64
```

func (`*RGBA`) `RGBAAt`

added in go1.4

```
func (p *RGBA) RGBAAt(x, y int) color.RGBA
```

func (`*RGBA`) `Set`

```
func (p *RGBA) Set(x, y int, c color.Color)
```

func (`*RGBA`) `SetRGBA`

```
func (p *RGBA) SetRGBA(x, y int, c color.RGBA)
```

func (*RGBA) SetRGBA64

added in go1.17

```
func (p *RGBA) SetRGBA64(x, y int, c color.RGBA64)
```

func (*RGBA) SubImage

```
func (p *RGBA) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type RGBA64

```
type RGBA64 struct {  
    // Pix holds the image's pixels, in R, G, B, A order and big-endian format. The pixel  
    // (x, y) starts at Pix[(y-Rect.Min.Y)*Stride + (x-Rect.Min.X)*8].  
    Pix []uint8  
    // Stride is the Pix stride (in bytes) between vertically adjacent pixels.  
    Stride int  
    // Rect is the image's bounds.  
    Rect Rectangle  
}
```

RGBA64 is an in-memory image whose At method returns color.RGBA64 values.

func NewRGBA64

```
func NewRGBA64(r Rectangle) *RGBA64
```

NewRGBA64 returns a new RGBA64 image with the given bounds.

func (*RGBA64) At

```
func (p *RGBA64) At(x, y int) color.Color
```

func (*RGBA64) Bounds

```
func (p *RGBA64) Bounds() Rectangle
```

func (*RGBA64) ColorModel

```
func (p *RGBA64) ColorModel() color.Model
```

func (*RGBA64) Opaque

```
func (p *RGBA64) Opaque() bool
```


Opaque scans the entire image and reports whether it is fully opaque.

func (*RGBA64) [PixOffset](#)

```
func (p *RGBA64) PixOffset(x, y int) int
```

PixOffset returns the index of the first element of Pix that corresponds to the pixel at (x, y).

func (*RGBA64) [RGBA64At](#)

added in go1.4

```
func (p *RGBA64) RGBA64At(x, y int) color.RGBA64
```

func (*RGBA64) [Set](#)

```
func (p *RGBA64) Set(x, y int, c color.Color)
```

func (*RGBA64) [SetRGBA64](#)

```
func (p *RGBA64) SetRGBA64(x, y int, c color.RGBA64)
```

func (*RGBA64) [SubImage](#)

```
func (p *RGBA64) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

type [RGBA64Image](#)

added in go1.17

```
type RGBA64Image interface {  
    // RGBA64At returns the RGBA64 color of the pixel at (x, y). It is  
    // equivalent to calling At(x, y).RGBA() and converting the resulting  
    // 32-bit return values to a color.RGBA64, but it can avoid allocations  
    // from converting concrete color types to the color.Color interface type.  
    RGBA64At(x, y int) color.RGBA64  
    Image  
}
```

RGBA64Image is an Image whose pixels can be converted directly to a color.RGBA64.

type [Rectangle](#)

```
type Rectangle struct {  
    Min, Max Point  
}
```

A Rectangle contains the points with Min.X ≤ X < Max.X, Min.Y ≤ Y < Max.Y. It is well-formed if Min.X ≤ Max.X and likewise for Y. Points are always well-formed. A rectangle's methods always return well-

formed outputs for well-formed inputs.

A Rectangle is also an Image whose bounds are the rectangle itself. At returns color.Opaque for points in the rectangle and color.Transparent otherwise.

```
var ZR Rectangle
```

ZR is the zero Rectangle.

Deprecated: Use a literal image.Rectangle{} instead.

func Rect

```
func Rect(x0, y0, x1, y1 int) Rectangle
```

Rect is shorthand for Rectangle{Pt(x0, y0), Pt(x1, y1)}. The returned rectangle has minimum and maximum coordinates swapped if necessary so that it is well-formed.

func (Rectangle) Add

```
func (r Rectangle) Add(p Point) Rectangle
```

Add returns the rectangle r translated by p.

func (Rectangle) At

added in go1.5

```
func (r Rectangle) At(x, y int) color.Color
```

At implements the Image interface.

func (Rectangle) Bounds

added in go1.5

```
func (r Rectangle) Bounds() Rectangle
```

Bounds implements the Image interface.

func (Rectangle) Canon

```
func (r Rectangle) Canon() Rectangle
```

Canon returns the canonical version of r. The returned rectangle has minimum and maximum coordinates swapped if necessary so that it is well-formed.

func (Rectangle) ColorModel

added in go1.5

```
func (r Rectangle) ColorModel() color.Model
```

ColorModel implements the Image interface.

func (Rectangle) Dx

```
func (r Rectangle) Dx() int
```

Dx returns r's width.

func (Rectangle) Dy

```
func (r Rectangle) Dy() int
```

Dy returns r's height.

func (Rectangle) Empty

```
func (r Rectangle) Empty() bool
```

Empty reports whether the rectangle contains no points.

func (Rectangle) Eq

```
func (r Rectangle) Eq(s Rectangle) bool
```

Eq reports whether r and s contain the same set of points. All empty rectangles are considered equal.

func (Rectangle) In

```
func (r Rectangle) In(s Rectangle) bool
```

In reports whether every point in r is in s.

func (Rectangle) Inset

```
func (r Rectangle) Inset(n int) Rectangle
```

Inset returns the rectangle r inset by n, which may be negative. If either of r's dimensions is less than 2*n then an empty rectangle near the center of r will be returned.

func (Rectangle) Intersect

```
func (r Rectangle) Intersect(s Rectangle) Rectangle
```

Intersect returns the largest rectangle contained by both r and s. If the two rectangles do not overlap then the zero rectangle will be returned.

func (Rectangle) Overlaps

```
func (r Rectangle) Overlaps(s Rectangle) bool
```

Overlaps reports whether r and s have a non-empty intersection.

func (Rectangle) **RGBA64At**

added in go1.17

```
func (r Rectangle) RGBA64At(x, y int) color.RGBA64
```

RGBA64At implements the RGBA64Image interface.

func (Rectangle) **Size**

```
func (r Rectangle) Size() Point
```

Size returns r's width and height.

func (Rectangle) **String**

```
func (r Rectangle) String() string
```

String returns a string representation of r like "(3,4)-(6,5)".

func (Rectangle) **Sub**

```
func (r Rectangle) Sub(p Point) Rectangle
```

Sub returns the rectangle r translated by -p.

func (Rectangle) **Union**

```
func (r Rectangle) Union(s Rectangle) Rectangle
```

Union returns the smallest rectangle that contains both r and s.

type **Uniform**

```
type Uniform struct {  
    c color.Color  
}
```

Uniform is an infinite-sized Image of uniform color. It implements the color.Color, color.Model, and Image interfaces.

func **NewUniform**

```
func NewUniform(c color.Color) *Uniform
```

NewUniform returns a new Uniform image of the given color.

func (*Uniform) **At**

```
func (c *Uniform) At(x, y int) color.Color
```

func (*Uniform) Bounds

```
func (c *Uniform) Bounds() Rectangle
```

func (*Uniform) ColorModel

```
func (c *Uniform) ColorModel() color.Model
```

func (*Uniform) Convert

```
func (c *Uniform) Convert(color.Color) color.Color
```

func (*Uniform) Opaque

```
func (c *Uniform) Opaque() bool
```

Opaque scans the entire image and reports whether it is fully opaque.

func (*Uniform) RGBA

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

func (*Uniform) RGBA64At

added in go1.17

```
func (c *Uniform) RGBA64At(x, y int) color.RGBA64
```

type YCbCr

```
type YCbCr struct {  
    Y, Cb, Cr      []uint8  
    YStride        int  
    CStride        int  
    SubsampleRatio YCbCrSubsampleRatio  
    Rect           Rectangle  
}
```

YCbCr is an in-memory image of Y'CbCr colors. There is one Y sample per pixel, but each Cb and Cr sample can span one or more pixels. YStride is the Y slice index delta between vertically adjacent pixels. CStride is the Cb and Cr slice index delta between vertically adjacent pixels that map to separate chroma samples. It is not an absolute requirement, but YStride and len(Y) are typically multiples of 8, and:

```
For 4:4:4, CStride == YStride/1 && len(Cb) == len(Cr) == len(Y)/1.  
For 4:2:2, CStride == YStride/2 && len(Cb) == len(Cr) == len(Y)/2.  
For 4:2:0, CStride == YStride/2 && len(Cb) == len(Cr) == len(Y)/4.  
For 4:4:0, CStride == YStride/1 && len(Cb) == len(Cr) == len(Y)/2.
```

```
For 4:1:1, CStride == YStride/4 && len(Cb) == len(Cr) == len(Y)/4.  
For 4:1:0, CStride == YStride/4 && len(Cb) == len(Cr) == len(Y)/8.
```

func NewYCbCr

```
func NewYCbCr(r Rectangle, subsampleRatio YCbCrSubsampleRatio) *YCbCr
```

NewYCbCr returns a new YCbCr image with the given bounds and subsample ratio.

func (*YCbCr) At

```
func (p *YCbCr) At(x, y int) color.Color
```

func (*YCbCr) Bounds

```
func (p *YCbCr) Bounds() Rectangle
```

func (*YCbCr) COffset

```
func (p *YCbCr) COffset(x, y int) int
```

COffset returns the index of the first element of Cb or Cr that corresponds to the pixel at (x, y).

func (*YCbCr) ColorModel

```
func (p *YCbCr) ColorModel() color.Model
```

func (*YCbCr) Opaque

```
func (p *YCbCr) Opaque() bool
```

func (*YCbCr) RGBA64At

added in go1.17

```
func (p *YCbCr) RGBA64At(x, y int) color.RGBA64
```

func (*YCbCr) SubImage

```
func (p *YCbCr) SubImage(r Rectangle) Image
```

SubImage returns an image representing the portion of the image p visible through r. The returned value shares pixels with the original image.

func (*YCbCr) YCbCrAt

added in go1.4

```
func (p *YCbCr) YCbCrAt(x, y int) color.YCbCr
```

func (*YCbCr) YOffset

```
func (p *YCbCr) YOffset(x, y int) int
```

YOffset returns the index of the first element of Y that corresponds to the pixel at (x, y).

type YCbCrSubsampleRatio

```
type YCbCrSubsampleRatio int
```

YCbCrSubsampleRatio is the chroma subsample ratio used in a YCbCr image.

```
const (  
    YCbCrSubsampleRatio444 YCbCrSubsampleRatio = iota  
    YCbCrSubsampleRatio422  
    YCbCrSubsampleRatio420  
    YCbCrSubsampleRatio440  
    YCbCrSubsampleRatio411  
    YCbCrSubsampleRatio410  
)
```

func (YCbCrSubsampleRatio) String

```
func (s YCbCrSubsampleRatio) String() string
```



Source Files

[View all](#) 

[format.go](#)

[geom.go](#)

[image.go](#)

[names.go](#)

[ycbcr.go](#)



Directories

[Expand all](#)

▶ [color](#)

Package color implements a basic color library.

[draw](#)

Package draw provides image composition functions.

[gif](#)

Package gif implements a GIF image decoder and encoder.

[jpeg](#)

Package jpeg implements a JPEG image decoder and encoder.

[png](#)

Package png implements a PNG image decoder and encoder.

Why Go

[Use Cases](#)

[Case Studies](#)

Get Started

[Playground](#)

[Tour](#)

[Stack Overflow](#)

[Help](#)

Packages

[Standard Library](#)

[About Go Packages](#)

About

[Download](#)

[Blog](#)

[Issue Tracker](#)

[Release Notes](#)

[Brand Guidelines](#)

[Code of Conduct](#)

Connect

[Twitter](#)

[GitHub](#)

[Slack](#)

[r/golang](#)

[Meetup](#)

[Golang Weekly](#)

[Copyright](#)

[Terms of Service](#)

[Privacy Policy](#)

[Report an Issue](#)



Google