NSColor

Inherits From:	NSResponder : NSObject
Conforms To:	NSCoding NSCopying NSObject (NSObject)
Declared In:	AppKit/NSColor.h

Class at a Glance

Purpose

An NSColor object represents a color, which is defined in a *color space*, each point of which has a set of components (such as red, green, and blue) that uniquely define a color.

Principal Attributes

- Color space
- Color components

Creation

colorWith... methods
Individual color methods

Commonly Used Methods

colorSpaceName	Gets the NSColor's color space.
set	Sets the drawing color.

Class Description

An NSColor object represents a color. The color can be a grayscale value and can include alpha (opacity) information. By sending a **set** message to an NSColor instance, you set the color for the current PostScript drawing context. This causes subsequently drawn graphics to have the color represented by the NSColor instance.

Color Spaces

A color is defined in some particular *color space*. A color space consists of a set of dimensions—such as red, green, and blue in the case of RGB space. Each point in the space represents a unique color, and the point's location along each dimension is called a *component*. An individual color is usually specified by the numeric values of its components, which range from 0.0 to 1.0. For instance, a pure red is specified in RGB space by the component values 1.0, 0.0, and 0.0.

Some color spaces include an alpha component, which defines the color's opacity. An alpha value of 1.0 means completely opaque, and 0.0 means completely transparent. The alpha component is ignored when the color is used on a device that doesn't support alpha, such as a printer.

There are three kinds of color spaces in NEXTSTEP:

- *Device-dependent*. This means that a given color might not look the same on different displays and printers.
- *Device-independent*, also known as *calibrated*. With this sort of color space, a given color should look the same on all devices.
- *Named*. The "named color space" has components that aren't numeric values, but simply names in various catalogs of colors. Named colors come with lookup tables that provide the ability to generate the correct color on a given device.

NEXTSTEP includes six different color spaces, referred to by these enumeration constants:

NSDeviceCMYKColorSpace	Cyan, magenta, yellow, black, and alpha components
NSDeviceWhiteColorSpace	White and alpha components
NSDeviceRGBColorSpace	Red, green, blue, and alpha components Hue, saturation, brightness, and alpha components
NSCalibratedWhiteColorSpace	White and alpha components
NSCalibratedRGBColorSpace	Red, green, blue, and alpha components Hue, saturation, brightness, and alpha components
NSNamedColorSpace	Catalog name and color name components

(Color spaces whose names start with "NSDevice" are device-dependent; those with "NSCalibrated" are device-independent.)

Color Components

There's usually no need to retrieve the individual components of a color, but when needed, you can either retrieve a set of components (using such methods as **getRed:green:blue:alpha:**) or an individual component (using such methods as **redComponent**). However, it's illegal to ask an NSColor for components that aren't defined for its color space. You can identify the color space by sending a **colorSpaceName** message to the NSColor object. If you need to ask an NSColor for components that aren't in its color space (for instance, when you've gotten the color from the color panel), first convert the color to the appropriate color space using the **colorUsingColorSpaceName:** method. If the color is already in the specified color space, you get the same color back; otherwise you get a conversion that's usually lossy or that's correct only for the current device. You get back **nil** if the specified conversion can't be done.

Creating Subclasses

Subclasses of NSColor need to implement the **colorSpaceName** and **set** methods, as well as the methods that return the components for that color space and the methods in the NSCoding protocol. Some other methods—such as **colorWithAlphaComponent:**, **isEqual:**, and **colorUsingColorSpaceName:device:**—may also be implemented if they make sense for the color space. Mutable subclasses (if any) should additionally implement **copyWithZone:** to provide a true copy.

Adopted Protocols

NSCoding	– encodeWithCoder: – initWithCoder:
NSCopying	– copyWithZone:

Method Types

Creating an NSColor object from Component Values

+ colorWithCalibratedHue:saturation:brightness:alpha:
+ colorWithCalibratedRed:green:blue:alpha:
+ colorWithCalibratedWhite:alpha:alpha
+ colorWithCatalogName:colorName:
+ colorWithDeviceCyan:magenta:yellow:black:alpha:
+ colorWithDeviceHue:saturation:brightness:alpha:
+ colorWithDeviceRed:green:blue:alpha:
+ colorWithDeviceWhite:alpha:

Creating an NSColor With Preset Components	+ blackColor + blueColor + brownColor + clearColor + cyanColor + darkGrayColor + grayColor + greenColor + lightGrayColor + magentaColor + orangeColor + purpleColor + redColor + whiteColor + yellowColor
Ignoring Alpha Components	+ ignoresAlpha + setIgnoresAlpha:
Copying and Pasting	+ colorFromPasteboard: – writeToPasteboard:
Retrieving a Set of Components	– getCyan:magenta:yellow:black:alpha: – getHue:saturation:brightness:alpha: – getRed:green:blue:alpha: – getWhite:alpha:
Retrieving Individual Components	– alphaComponent – blackComponent – blueComponent – brightnessComponent – catalogNameComponent – colorNameComponent – cyanComponent – greenComponent – hueComponent – localizedCatalogNameComponent – localizedColorNameComponent – magentaComponent – redComponent – saturationComponent – whiteComponent – yellowComponent

Converting to Another Color Space	– colorSpaceName – colorUsingColorSpaceName: – colorUsingColorSpaceName:device:
Changing the Color	– blendedColorWithFraction:ofColor: – colorWithAlphaComponent:
Drawing	– drawSwatchInRect: – set

Class Methods

blackColor

+ (NSColor *)**blackColor**

Returns an NSColor in NSCalibratedWhiteColorSpace whose grayscale value is 0.0 and whose alpha value is 1.0.

See also: – blackComponent

blueColor

+ (NSColor *)**blueColor**

Returns an NSColor in NSCalibratedRGBColorSpace whose RGB value is 0.0, 0.0, 1.0 and whose alpha value is 1.0.

See also: – blueComponent

brownColor

+ (NSColor *)**brownColor**

Returns an NSColor in NSCalibratedRGBColorSpace whose RGB value is 0.6, 0.4, 0.2 and whose alpha value is 1.0.

clearColor

+ (NSColor *)**clearColor**

Returns an NSColor in NSCalibratedWhiteColorSpace whose grayscale and alpha values are both 0.0.

colorFromPasteboard:

+ (NSColor *)**colorFromPasteboard:**(NSPasteboard *)*pasteBoard*

Returns the NSColor currently on the pasteboard, or **nil** if the pasteboard doesn't contain color data. The returned color's alpha component is set to 1.0 if **ignoresAlpha** returns YES.

This method invokes **colorWithAlphaComponent:**.

See also: – writeToPasteboard:

colorWithCalibratedHue:saturation:brightness:alpha:

+ (NSColor *)**colorWithCalibratedHue:**(float)*hue*
saturation:(float)*saturation*
brightness:(float)*brightness*
alpha:(float)*alpha*

Creates and returns a new NSColor whose color space is NSCalibratedRGBColorSpace, whose opacity value is *alpha*, and whose components in HSB space would be *hue*, *saturation*, and *brightness*. All values are not legal: Values less than 0.0 are set to 0.0, and values greater than 1.0 are set to 1.0.

See also: **+ colorWithCalibratedRed:green:blue:alpha:**, **+ colorWithDeviceHue:saturation:brightness:alpha:**, **– getHue:saturation:brightness:alpha:**

colorWithCalibratedRed:green:blue:alpha:

+ (NSColor *)**colorWithCalibratedRed:**(float)*red*
green:(float)*green*
blue:(float)*blue*
alpha:(float)*alpha*

Creates and returns a new NSColor whose color space is NSCalibratedRGBColorSpace, whose opacity value is *alpha*, and whose RGB components are *red*, *green*, and *blue*. All values are not legal: Values less than 0.0 are set to 0.0, and values greater than 1.0 are set to 1.0.

See also: **+ colorWithCalibratedHue:saturation:brightness:alpha:**, **+ colorWithDeviceRed:green:blue:alpha:**, **– getRed:green:blue:alpha:**

colorWithCalibratedWhite:alpha:

+ (NSColor *)**colorWithCalibratedWhite:**(float)*white*
  alpha:(float)*alpha*

Creates and returns a new NSColor whose color space is NSCalibratedWhiteColorSpace, whose opacity value is *alpha*, and whose grayscale value is *white*. All values are not legal: Values less than 0.0 are set to 0.0, and values greater than 1.0 are set to 1.0.

See also: **+ colorWithDeviceWhite:alpha:**, **– getWhite:alpha:**

colorWithCatalogName:colorName:

+ (NSColor *)**colorWithCatalogName:**(NSString *)*listName*
  colorName:(NSString *)*colorName*

Creates and returns a new NSColor whose color space is NSNamedColorSpace, by finding the color named *colorName* in the catalog named *listName*, which may be a standard catalog.

See also: **– catalogNameComponent**, **– colorNameComponent**, **– localizedCatalogNameComponent**

colorWithDeviceCyan:magenta:yellow:black:alpha:

+ (NSColor *)**colorWithDeviceCyan:**(float)*cyan*
  magenta:(float)*magenta*
  yellow:(float)*yellow*
  black:(float)*black*
  alpha:(float)*alpha*

Creates and returns a new NSColor whose color space is NSDeviceCMYKColorSpace, whose opacity value is *alpha*, and whose CMYK components are *cyan*, *magenta*, *yellow*, and *black*. All values are not legal: Values less than 0.0 are set to 0.0, and values greater than 1.0 are set to 1.0. In PostScript, this colorspace corresponds directly to the device-dependent operator *setcmykcolor*.

See also: – **getCyan:magenta:yellow:black:alpha:**

colorWithDeviceHue:saturation:brightness:alpha:

+ (NSColor *)**colorWithDeviceHue:**(float)*hue*
  saturation:(float)*saturation*
  brightness:(float)*brightness*
  alpha:(float)*alpha*

Creates and returns a new NSColor whose color space is NSDeviceRGBColorSpace, whose opacity value is *alpha*, and whose components in HSB space would be *hue*, *saturation*, and *brightness*. All values are not legal: Values less than 0.0 are set to 0.0, and values greater than 1.0 are set to 1.0. In PostScript, this colorspace corresponds directly to the device-dependent operator *setrgbcolor*.

See also: **+ colorWithCalibratedHue:saturation:brightness:alpha:**, **– colorWithDeviceRed:green:blue:alpha:**, **– getHue:saturation:brightness:alpha:**

colorWithDeviceRed:green:blue:alpha:

+ (NSColor *)**colorWithDeviceRed:**(float)*red*
green:(float)*green*
blue:(float)*blue*
alpha:(float)*alpha*

Creates and returns a new NSColor whose color space is NSDeviceRGBColorSpace, whose opacity value is *alpha*, and whose RGB components are *red*, *green*, and *blue*. All values are not legal: Values less than 0.0 are set to 0.0, and values greater than 1.0 are set to 1.0. In PostScript, this colorspace corresponds directly to the device-dependent operator *setrgbcolor*.

See also: **+ colorWithCalibratedRed:green:blue:alpha:**, **– colorWithDeviceHue:saturation:brightness:alpha:**, **– getRed:green:blue:alpha:**

colorWithDeviceWhite:alpha:

+ (NSColor *)**colorWithDeviceWhite:**(float)*white*
alpha:(float)*alpha*

Creates and returns a new NSColor whose color space is NSDeviceWhiteColorSpace, whose opacity value is *alpha*, and whose grayscale value is *white*. All values are not legal: Values less than 0.0 are set to 0.0, and values greater than 1.0 are set to 1.0. In PostScript, this colorspace corresponds directly to the device-dependent operator *setgray*.

See also: **+ colorWithCalibratedWhite:alpha:**, **– getWhite:alpha:**

cyanColor

+ (NSColor *)**cyanColor**

Returns an NSColor in NSCalibratedRGBColorSpace whose RGB value is 0.0, 1.0, 1.0 and whose alpha value is 1.0.

See also: – **cyanComponent**

darkGrayColor

+ (NSColor *)**darkGrayColor**

Returns an NSColor in NSCalibratedWhiteColorSpace whose grayscale value is 1/3 and whose alpha value is 1.0.

grayColor

+ (NSColor *)**grayColor**

Returns an NSColor in NSCalibratedWhiteColorSpace whose grayscale value is 0.5 and whose alpha value is 1.0.

greenColor

+ (NSColor *)**greenColor**

Returns an NSColor in NSCalibratedRGBColorSpace whose RGB value is 0.0, 1.0, 0.0 and whose alpha value is 1.0.

See also: – **greenComponent**

ignoresAlpha

+ (BOOL)**ignoresAlpha**

Returns YES if the application doesn't support alpha. This value returned is consulted when an application imports alpha (through color dragging, for instance). The value determines whether the color panel has an opacity slider. This value is YES by default, indicating that the opacity components of imported colors will be set to 1.0. If an application wants alpha, it can invoke the **setIgnoresAlpha** method with a parameter of NO.

See also: **+ setIgnoresAlpha**, **– alphaComponent**

lightGrayColor

+ (NSColor *)**lightGrayColor**

Returns an NSColor in NSCalibratedWhiteColorSpace whose grayscale value is 2/3 and whose alpha value is 1.0.

magentaColor

+ (NSColor *)**magentaColor**

Returns an NSColor in NSCalibratedRGBColorSpace whose RGB value is 1.0, 0.0, 1.0 and whose alpha value is 1.0.

See also: **– magentaComponent**

orangeColor

+ (NSColor *)**orangeColor**

Returns an NSColor in NSCalibratedRGBColorSpace whose RGB value is 1.0, 0.5, 0.0 and whose alpha value is 1.0.

purpleColor

+ (NSColor *)**purpleColor**

Returns an NSColor in NSCalibratedRGBColorSpace whose RGB value is 0.5, 0.0, 0.5 and whose alpha value is 1.0.

redColor

+ (NSColor *)**redColor**

Returns an NSColor in NSCalibratedRGBColorSpace whose RGB value is 1.0, 0.0, 0.0 and whose alpha value is 1.0.

See also: – **redComponent**

setIgnoresAlpha:

+ (void)**setIgnoresAlpha:**(BOOL)*flag*

If *flag* is YES, The application won't support alpha. In this case, no opacity slider is displayed in the color panel, and colors dragged in or pasted have their alpha values set to 1.0. Applications which need to import alpha should invoke this method with *flag* set to NO and explicitly make colors opaque in cases where it matters to them.

See also: **+ ignoresAlpha**, **– alphaComponent**

whiteColor

+ (NSColor *)**whiteColor**

Returns an NSColor in NSCalibratedWhiteColorSpace whose grayscale and alpha values are both 1.0.

See also: – whiteComponent

yellowColor

+ (NSColor *)**yellowColor**

Returns an NSColor in NSCalibratedRGBColorSpace whose RGB value is 1.0, 1.0, 0.0 and whose alpha value is 1.0.

See also: – **yellowComponent**

Instance Methods

alphaComponent

– (float)**alphaComponent**

Returns the alpha (opacity) component. Returns 1.0 (opaque) for colors that do not have alpha components.

See also: **– getCyan:magenta:yellow:black:alpha:**, **– getHue:saturation:brightness:alpha:**, **– getRed:green:blue:alpha:**, **– getWhite:alpha:**

blackComponent

– (float)**blackComponent**

Returns the black component. It's an error if the receiver isn't a CMYK color.

See also: **+ blackColor**, **– getCyan:magenta:yellow:black:alpha:**

blendedColorWithFraction:ofColor:

– (NSColor *)**blendedColorWithFraction:**(float)*fraction*
    ofColor:(NSColor *)*color*

Returns a newly created NSColor in NSCalibratedRGBColorSpace whose component values are a weighted sum of the receiver's and *color*'s. The method converts *color* and a copy of the receiver to RGB, and then sets each component of the returned color to *fraction* of *color*'s value plus 1 – *fraction* of the receiver's. If the colors can't be converted to NSCalibratedRGBColorSpace, **nil** is returned.

This method invokes **colorUsingColorSpaceName:**.

blueComponent

– (float)**blueComponent**

Returns the blue component. It’s an error if the receiver isn’t an RGB color.

See also: **+ blueColor**, **– getRed:green:blue:alpha:**

brightnessComponent

– (float)**brightnessComponent**

Returns the brightness component of the HSB color equivalent to the receiver. It’s an error if the receiver isn’t an RGB color.

See also: – **getHue:saturation:brightness:alpha:**

catalogNameComponent

– (NSString *)**catalogNameComponent**

Returns the name of the catalog containing this color, or **nil** if the receiver’s color space isn’t NSNamedColorSpace.

See also: **+ colorWithCatalogName:colorName:**, **– colorNameComponent**, **– localizedCatalogNameComponent**

colorNameComponent

– (NSString *)**colorNameComponent**

Returns the name of this color, or **nil** if the receiver’s color space isn’t NSNamedColorSpace.

See also: **+ colorWithCatalogName:colorName:**, **– catalogNameComponent**, **– localizedCatalogNameComponent**

colorSpaceName

– (NSString *)**colorSpaceName**

Returns the name of the NSColor's color space. This method should be implemented in subclasses of NSColor.

See also: **– colorUsingColorSpaceName:**, **– colorUsingColorSpaceName:device:**

colorUsingColorSpaceName:

– (NSColor *)**colorUsingColorSpaceName:**(NSString *)*colorSpace*

Returns a newly created NSColor whose color is the same as the receiver's, except that the new NSColor is in the color space named *colorSpace*. If *colorSpace* is **nil**, the most appropriate color space is used.

This method invokes **colorUsingColorSpaceName:device:** with a **nil** device, indicating that the color is appropriate for the current device (the current window if drawing, or the current printer if printing).

Returns **nil** if the specified conversion cannot be done.

See also: – **colorSpaceName:**

colorUsingColorSpaceName:device:

– (NSColor *)**colorUsingColorSpaceName:**(NSString *)*colorSpace*
device:(NSDictionary *)*deviceDescription*

Returns a newly created NSColor whose color is the same as the receiver's, except that the new NSColor is in the color space named *colorSpace* and is specific to the device described by *deviceDescription*. Device descriptions can be obtained from windows, screens, and printers with the **deviceDescription** method. If *colorSpace* is **nil**, the most appropriate color space is used.

If *deviceDescription* is **nil,** the current device (as obtained from the currently lockFocus'ed view's window or, if printing, the current printer) is used.

This method invokes **colorSpaceName.**

Returns **nil** if the specified conversion cannot be done.

See also: **– colorSpaceName:**, **– colorUsingColorSpaceName:**

colorWithAlphaComponent:

– (NSColor *)**colorWithAlphaComponent:**(float)*alpha*

Returns a newly created NSColor that has the same color space and component values as the receiver, except that its alpha component is *alpha*. If the receiver's color space doesn't include an alpha component, the receiver is returned. A subclass which has explicit opacity components should override this method to return a color with the specified *alpha*.

See also: **– alphaComponent**, **– blendedColorWithFraction:ofColor:**

cyanComponent

– (float)**cyanComponent**

Returns the cyan component. It's an error if the receiver isn't a CMYK color.

See also: **+ cyanColor**, **– getCyan:magenta:yellow:black:alpha:**

drawSwatchInRect:

– (void)**drawSwatchInRect:**(NSRect)*rect*

Draws the current color in the rectangle *rect*. Subclasses adorn the rectangle in some manner to indicate the type of color. This method is invoked by color wells, swatches, and other user-interface objects that need to display colors.

getCyan:magenta:yellow:black:alpha:

– (void)**getCyan:**(float *)*cyan*
magenta:(float *)*magenta*
yellow:(float *)*yellow*
black:(float *)*black*
alpha:(float *)*alpha*

Returns the CMYK and alpha values in the respective arguments. If NULL is passed in as an argument, the method doesn't set that value. It's an error if the receiver isn't a CMYK color.

See also: **– alphaComponent**, **– blackComponent**, **– cyanComponent**, **– magentaComponent**, **– yellowComponent**

getHue:saturation:brightness:alpha:

– (void)**getHue:**(float *)*hue*
 saturation:(float *)*saturation*
 brightness:(float *)*brightness*
 alpha:(float *)*alpha*

Returns the HSB and alpha values in the respective arguments. If NULL is passed in as an argument, the method doesn't set that value. It's an error if the receiver isn't an RGB color.

See also: **– alphaComponent**, **– brightnessComponent**, **– hueComponent**, **– saturationComponent**

getRed:green:blue:alpha:

– (void)**getRed:**(float *)*red*
 green:(float *)*green*
 blue:(float *)*blue*
 alpha:(float *)*alpha*

Returns the RGB and alpha values in the respective arguments. If NULL is passed in as an argument, the method doesn't set that value. It's an error if the receiver isn't an RGB color.

See also: **– alphaComponent**, **– blueComponent**, **– greenComponent**, **– redComponent**

getWhite:alpha:

– (void)**getWhite:**(float *)*white*
 alpha:(float *)*alpha*

Returns the grayscale and alpha values in the respective arguments. If NULL is passed in as an argument, the method doesn't set that value. It's an error if the receiver isn't a grayscale color.

See also: **– alphaComponent**, **– whiteComponent**

greenComponent

– (float)**greenComponent**

Returns the green component. It's an error if the receiver isn't an RGB color.

See also: **+ greenColor**, **– getRed:green:blue:alpha:**

hueComponent

– (float)**hueComponent**

Returns the hue component of the HSB color equivalent to the receiver. It’s an error if the receiver isn’t an RGB color.

See also: **– getHue:saturation:brightness:alpha:**

localizedCatalogNameComponent

– (NSString *)**localizedCatalogNameComponent**

Like **catalogNameComponent**, but returns a localized string.

See also: **+ colorWithCatalogName:colorName:**, **– colorNameComponent**

localizedColorNameComponent

– (NSString *)**localizedColorNameComponent**

Like **colorNameComponent**, but returns a localized string.

See also: **+ colorWithCatalogName:colorName:**, **– catalogNameComponent**, **– colorNameComponent**, **– localizedCatalogNameComponent**

magentaComponent

– (float)**magentaComponent**

Returns the magenta component. It’s an error if the receiver isn’t a CMYK color.

See also: **+ magentaColor**, **– getCyan:magenta:yellow:black:alpha:**

redComponent

– (float)**redComponent**

Returns the red component. It’s an error if the receiver isn’t an RGB color.

See also: **+ redColor**, **– getRed:green:blue:alpha:**

saturationComponent

– (float)**saturationComponent**

Returns the saturation component of the HSB color equivalent to the receiver. It’s an error if the receiver isn’t an RGB color.

See also: – **getHue:saturation:brightness:alpha:**

set

– (void)**set**

Sets the color of subsequent PostScript drawing to the color that the receiver represents. If the application is drawing to the screen rather than printing, this method also sets the current drawing context’s alpha value to the value returned by **alphaComponent**; if the color doesn’t know about alpha, it’s set to 1.0. This method should be implemented in subclasses.

whiteComponent

– (float)**whiteComponent**

Returns the white component. It’s an error if the receiver isn’t a grayscale color.

See also: + **whiteColor**, **– getWhite:alpha:**

writeToPasteboard:

– (void)**writeToPasteboard:**(NSPasteboard *)*pasteBoard*

Writes the receiver’s data to the pasteboard, unless the pasteboard doesn’t support color data (in which case the method does nothing).

See also: + **colorFromPasteboard:**

yellowComponent

– (float)**yellowComponent**

Returns the yellow component. It’s an error if the receiver isn’t a CMYK color.

See also: + **yellowColor**, **– getCyan:magenta:yellow:black:alpha:**