

---

# UIColor Class Reference

Cocoa > Graphics & Imaging



2006-12-05



Apple Inc.  
© 2006 Apple Computer, Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
408-996-1010

Apple, the Apple logo, Cocoa, Mac, Mac OS, and Quartz are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY**

**DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

---

## **UIColor Class Reference 5**

---

Overview	5
Tasks	6
Initializing Color Objects	6
Creating Color Objects	6
Getting Color Components	6
Class Methods	7
colorWithCGColor:	7
colorWithRed:green:blue:	7
colorWithRed:green:blue:alpha:	8
colorWithString:	9
Instance Methods	9
alpha	9
blue	10
colorSpace	10
components	10
green	11
initWithCGColor:	11
numberOfComponents	11
red	12
stringRepresentation	12

---

## **Document Revision History 13**

---

## **Index 15**

---



# UIColor Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSCoding NSCopying NSObject (NSObject)
<b>Framework</b>	Library/Frameworks/QuartzCore.framework
<b>Availability</b>	Mac OS X v10.4 and later
<b>Declared in</b>	UIColor.h
<b>Companion guides</b>	Core Image Programming Guide Color Management Overview
<b>Related sample code</b>	UIColorAnnotation

## Overview

The `UIColor` class contains color values and the color space for which the color values are valid. You use `UIColor` objects in conjunction with other Core Image classes, such as `CIFilter`, `CIContext`, and `CIImage`, to take advantage of the built-in Core Image filters when processing images.

A color space defines a one-, two-, three-, or four-dimensional environment whose color components represent intensity values. A color component is also referred to as a color channel. An RGB color space, for example, is a three-dimensional color space whose stimuli are the red, green, and blue intensities that make up a given color. Regardless of the color space, in Core Image, color values range from 0.0 to 1.0, with 0.0 representing an absence of that component (0 percent) and 1.0 representing 100 percent.

Colors also have an alpha component that represents the opacity of the color, with 0.0 meaning completely transparent and 1.0 meaning completely opaque. If a color does not have an explicit alpha component, Core Image paints the color as if the alpha component equals 1.0. You always provide unmultiplied color components to Core Image and Core Image provides unmultiplied color components to you. Core Image multiplies each color component with the alpha value in order to optimize calculations. For more information on multiplied alpha values see *Core Image Programming Guide*.

## Tasks

### Initializing Color Objects

- `initWithCGColor:` (page 11)  
Initializes a color object with a Quartz color.

### Creating Color Objects

- + `colorWithCGColor:` (page 7)  
Creates a color object from a Quartz color.
- + `colorWithRed:green:blue:` (page 7)  
Creates a color object using the specified RGB color component values
- + `colorWithRed:green:blue:alpha:` (page 8)  
Creates a color object using the specified RGBA color component values.
- + `colorWithString:` (page 9)  
Creates a color object using the RGBA color component values specified by a string.

### Getting Color Components

- `alpha` (page 9)  
Returns the alpha value of the color.
- `blue` (page 10)  
Returns the blue component of the color.
- `colorSpace` (page 10)  
Returns the Quartz 2D color space associated with the color.
- `components` (page 10)  
Returns the color components of the color.
- `green` (page 11)  
Returns the green component of the color.
- `numberOfComponents` (page 11)  
Returns the number of color components in the color.
- `red` (page 12)  
Returns the red component of the color.
- `stringRepresentation` (page 12)  
Returns a formatted string that specifies the components of the color.

## Class Methods

### colorWithCGColor:

Creates a color object from a Quartz color.

```
+ (CIColor *) colorWithCGColor:(CGColorRef)c
```

#### Parameters

*c*

A Quartz color (`CGColorRef` object) created using a Quartz color creation function such as `CGColorCreate`.

#### Return Value

A Core Image color object that represents a Quartz color.

#### Discussion

A `CGColorRef` object is the fundamental opaque data type used internally by Quartz to represent colors. For more information on Quartz 2D color and color spaces, see *Quartz 2D Programming Guide*.

You can pass a `CGColorRef` object that represents any color space, including CMYK, but Core Image converts all color spaces to the Core Image working color space before it passes the color space to the filter kernel. The Core Image working color space uses three color components plus alpha.

#### Availability

Mac OS X v10.4 and later.

#### See Also

+ [colorWithRed:green:blue:](#) (page 7)

+ [colorWithRed:green:blue:alpha:](#) (page 8)

+ [colorWithString:](#) (page 9)

#### Declared In

`CIColor.h`

### colorWithRed:green:blue:

Creates a color object using the specified RGB color component values

```
+ (CIColor *) colorWithRed:(CGFloat)r green:(CGFloat)g blue:(CGFloat)b
```

#### Parameters

*r*

The value of the red component.

*g*

The value of the green component.

*b*

The value of the blue component.

**Return Value**

A Core Image color object that represents an RGB color in the color space specified by the Quartz 2D constant `kCGColorSpaceGenericRGB`.

**Availability**

Mac OS X v10.4 and later.

**See Also**

- + [colorWithCGColor:](#) (page 7)
- + [colorWithRed:green:blue:alpha:](#) (page 8)
- + [colorWithString:](#) (page 9)

**Declared In**

`CIColor.h`

**colorWithRed:green:blue:alpha:**

Creates a color object using the specified RGBA color component values.

```
+ (CIColor *) colorWithRed:(CGFloat)r green:(CGFloat)g blue:(CGFloat)b
    alpha:(CGFloat)a
```

**Parameters**

- r*  
The value of the red component.
- g*  
The value of the green component.
- b*  
The value of the blue component.
- a*  
The value of the alpha component.

**Return Value**

A Core Image color object that represents an RGB color in the color space specified by the Quartz 2D constant `kCGColorSpaceGenericRGB` and an alpha value.

**Availability**

Mac OS X v10.4 and later.

**See Also**

- + [colorWithCGColor:](#) (page 7)
- + [colorWithRed:green:blue:](#) (page 7)
- + [colorWithString:](#) (page 9)

**Related Sample Code**

`CIAnnotation`

**Declared In**

`CIColor.h`



**colorWithString:**

Creates a color object using the RGBA color component values specified by a string.

```
+ (CIColor *)colorWithString:(NSString *)representation
```

**Parameters**

*representation*

A string that is in one of the formats returned by the `stringRepresentation` method. For example, the string:

```
@"0.5 0.7 0.3 1.0"
```

indicates an RGB color whose components are 50% red, 70% green, 30% blue, and 100% opaque (alpha value of 1.0). The string representation always has four components—red, green, blue, and alpha. The default value for the alpha component is 1.0.

**Return Value**

A Core Image color object that represents an RGB color in the color space specified by the Quartz 2D constant `kCGColorSpaceGenericRGB`.

**Availability**

Mac OS X v10.4 and later.

**See Also**

- + [colorWithCGColor:](#) (page 7)
- + [colorWithRed:green:blue:](#) (page 7)
- + [colorWithRed:green:blue:alpha:](#) (page 8)

**Declared In**

CIColor.h

## Instance Methods

**alpha**

Returns the alpha value of the color.

```
- (CGFloat)alpha
```

**Return Value**

The alpha value. A color created without an explicit alpha value has an alpha of 1.0 by default.

**Availability**

Mac OS X v10.4 and later.

**See Also**

- [components](#) (page 10)

**Declared In**

CIColor.h

## blue

Returns the blue component of the color.

- (CGFloat)blue

### Return Value

The unmultiplied blue component of the color.

### Availability

Mac OS X v10.4 and later.

### See Also

- [components](#) (page 10)

### Declared In

CIColor.h

## colorSpace

Returns the Quartz 2D color space associated with the color.

- (CGColorSpaceRef)colorSpace

### Return Value

The Quartz 2D color space (CGColorSpaceRef object). You are responsible for disposing of this color space by calling the Quartz 2D function CGColorSpaceRelease.

### Availability

Mac OS X v10.4 and later.

### See Also

- [components](#) (page 10)

### Declared In

CIColor.h

## components

Returns the color components of the color.

- (const CGFloat \*)components

### Return Value

An array of color components, specified as floating-point values in the range of 0.0 through 1.0. This array includes an alpha component if there is one.

### Availability

Mac OS X v10.4 and later.

### See Also

- [numberOfComponents](#) (page 11)

- [stringRepresentation](#) (page 12)

**Declared In**

CIColor.h

**green**

Returns the green component of the color.

- (CGFloat)green

**Return Value**

The unmultiplied green component of the color.

**Availability**

Mac OS X v10.4 and later.

**See Also**- [components](#) (page 10)**Declared In**

CIColor.h

**initWithCGColor:**

Initializes a color object with a Quartz color.

- (id)initWithCGColor:(CGColorRef)c

**Parameters**

c

A Quartz color (CGColorRef) created using a Quartz color creation function such as CGColorCreate.

**Discussion**

A CGColorRef object is the fundamental opaque data type used internally by Quartz to represent colors. For more information on Quartz 2D color and color spaces, see *Quartz 2D Programming Guide*.

You can pass a CGColorRef object that represents any color space, including CMYK, but Core Image converts all color spaces to the Core Image working color space before it passes the color space to the filter kernel. The Core Image working color space uses three color components plus alpha.

**Availability**

Mac OS X v10.4 and later.

**Declared In**

CIColor.h

**numberOfComponents**

Returns the number of color components in the color.

- (size\_t)numberOfComponents

**Return Value**

The number of color components, which includes an alpha component if there is one.

**Availability**

Mac OS X v10.4 and later.

**See Also**

- [components](#) (page 10)

**Declared In**

CIColor.h

**red**

Returns the red component of the color.

- (CGFloat)red

**Return Value**

The unpremultiplied red component of the color.

**Availability**

Mac OS X v10.4 and later.

**See Also**

- [components](#) (page 10)

**Declared In**

CIColor.h

**stringRepresentation**

Returns a formatted string that specifies the components of the color.

- (NSString \*)stringRepresentation

**Return Value**

The formatted string.

**Discussion**

The string representation always has four components—red, green, blue, and alpha. The default value for the alpha component is 1.0. For example, this string:

```
@"0.5 0.7 0.3 1.0"
```

indicates an RGB color whose components are 50% red, 70% green, 30% blue, and 100% opaque (alpha value of 1.0).

**Availability**

Mac OS X v10.4 and later.

**See Also**

- [components](#) (page 10)

**Declared In**

CIColor.h

# Document Revision History

---

This table describes the changes to *CIColor Class Reference*.

Date	Notes
2006-12-05	Revised class meta information.
2006-05-23	First publication of this content as a separate document.
	Added parameter descriptions and updated Class Description.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

alpha [instance method 9](#)

## B

---

blue [instance method 10](#)

## C

---

colorSpace [instance method 10](#)  
colorWithCGColor: [class method 7](#)  
colorWithRed:green:blue: [class method 7](#)  
colorWithRed:green:blue:alpha: [class method 8](#)  
colorWithString: [class method 9](#)  
components [instance method 10](#)

## G

---

green [instance method 11](#)

## I

---

initWithCGColor: [instance method 11](#)

## N

---

numberOfComponents [instance method 11](#)

## R

---

red [instance method 12](#)

## S

---

stringRepresentation [instance method 12](#)