
Core Graphics Framework Reference

Graphics & Animation: 2D Drawing



2009-05-14



Apple Inc.
© 2009 Apple 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, Carbon, Cocoa, iPhone, Mac, Mac OS, Pages, Quartz, QuickDraw, QuickTime, Spaces, and TrueType are trademarks of Apple Inc., registered in the United States and other countries.

Adobe, Acrobat, and PostScript are trademarks or registered trademarks of Adobe Systems Incorporated in the U.S. and/or other countries.

IOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

OpenGL is a registered trademark of Silicon Graphics, Inc.

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

Introduction	Introduction	9
---------------------	-------------------------------------	----------

Part I	Opaque Types	11
---------------	-------------------------------------	-----------

Chapter 1	CGBitmapContext Reference	13
------------------	--	-----------

[Overview](#) 13
[Functions by Task](#) 13
[Functions](#) 14
[Data Types](#) 22

Chapter 2	CGColor Reference	23
------------------	--	-----------

[Overview](#) 23
[Functions by Task](#) 23
[Functions](#) 24
[Data Types](#) 30

Chapter 3	CGColorSpace Reference	33
------------------	---	-----------

[Overview](#) 33
[Functions by Task](#) 34
[Functions](#) 35
[Data Types](#) 46
[Constants](#) 47

Chapter 4	CGContext Reference	51
------------------	--	-----------

[Overview](#) 51
[Functions by Task](#) 51
[Functions](#) 58
[Data Types](#) 132
[Constants](#) 133

Chapter 5	CGDataConsumer Reference	143
------------------	---	------------

[Overview](#) 143
[Functions by Task](#) 143
[Functions](#) 144
[Callbacks](#) 146
[Data Types](#) 148

Chapter 6 CGDataProvider Reference 151

Overview 151
Functions 151
Callbacks by Task 156
Callbacks 157
Data Types 164

Chapter 7 CGFont Reference 169

Overview 169
Functions by Task 169
Functions 171
Data Types 185
Constants 186

Chapter 8 CGFunction Reference 189

Overview 189
Functions by Task 189
Functions 190
Callbacks 192
Data Types 193

Chapter 9 CGGradient Reference 195

Overview 195
Functions by Task 195
Functions 196
Data Types 199
Constants 199

Chapter 10 CGImage Reference 201

Overview 201
Functions by Task 201
Functions 203
Data Types 218
Constants 218

Chapter 11 CGLayer Reference 223

Overview 223
Functions by Task 223
Functions 224
Data Types 228

Chapter 12 **CGPath Reference 229**

Overview 229
Functions by Task 229
Functions 231
Callbacks 247
Data Types 248
Constants 249

Chapter 13 **CGPattern Reference 253**

Overview 253
Functions by Task 253
Functions 254
Callbacks 256
Data Types 258
Constants 259

Chapter 14 **CGPDFArray Reference 261**

Overview 261
Functions 261
Data Types 267

Chapter 15 **CGPDFContentStream Reference 269**

Overview 269
Functions by Task 269
Functions 270
Data Types 273

Chapter 16 **CGPDFContext Reference 275**

Overview 275
Functions by Task 275
Functions 276
Constants 281

Chapter 17 **CGPDFDictionary Reference 285**

Overview 285
Functions by Task 285
Functions 286
Callbacks 292
Data Types 293

Chapter 18 **CGPDFDocument Reference 295**

Overview 295
Functions by Task 295
Functions 296
Data Types 304

Chapter 19 **CGPDFObject Reference 305**

Overview 305
Functions 305
Data Types 306
Constants 307

Chapter 20 **CGPDFOperatorTable Reference 311**

Overview 311
Functions by Task 311
Functions 312
Callbacks 313
Data Types 314

Chapter 21 **CGPDFPage Reference 315**

Overview 315
Functions by Task 315
Functions 316
Data Types 321
Constants 321

Chapter 22 **CGPDFScanner Reference 323**

Overview 323
Functions by Task 323
Functions 324
Data Types 331

Chapter 23 **CGPDFStream Reference 333**

Overview 333
Functions 333
Data Types 334
Constants 334

Chapter 24 **CGPDFString Reference 337**

Overview 337
Functions by Task 337
Functions 338
Data Types 339

Chapter 25 **CGShading Reference 341**

Overview 341
Functions by Task 341
Functions 342
Data Types 345

Part II **Constants 347**

Chapter 26 **Core Graphics Constants Reference 349**

Overview 349
Result Codes 349

Part III **Other References 353**

Chapter 27 **CGAffineTransform Reference 355**

Overview 355
Functions by Task 355
Functions 356
Data Types 366
Constants 367

Chapter 28 **CGGeometry Reference 369**

Overview 369
Functions by Task 369
Functions 372
Data Types 389
Constants 391

Document Revision History 395

Introduction

Framework:	CoreGraphics/CoreGraphics.h
Declared in	CABase.h CGAffineTransform.h CABase.h CGBitmapContext.h CGColor.h CGColorSpace.h CGContext.h CGDataConsumer.h CGDataProvider.h CGFont.h CGFunction.h CGGeometry.h CGGradient.h CGImage.h CGLayer.h CGPDFArray.h CGPDFContentStream.h CGPDFContext.h CGPDFDictionary.h CGPDFDocument.h CGPDFObject.h CGPDFOperatorTable.h CGPDFPage.h CGPDFScanner.h CGPDFStream.h CGPDFString.h CGPath.h CGPattern.h CGShading.h

The Core Graphics framework is a C-based API that is based on the Quartz advanced drawing engine. It provides low-level, lightweight 2D rendering with unmatched output fidelity. You use this framework to handle path-based drawing, transformations, color management, offscreen rendering, patterns, gradients and shadings, image data management, image creation, masking, and PDF document creation, display, and parsing.

Opaque Types

CGBitmapContext Reference

Derived From:	CGContextRef (page 132)
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGBitmapContext.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGBitmapContext.h` header file defines functions that create and operate on a Quartz bitmap graphics context. A bitmap graphics context is a type of [CGContextRef](#) (page 132) that you can use for drawing bits to memory. The functions in this reference operate only on Quartz bitmap graphics contexts created using the function [CGBitmapContextCreate](#) (page 14).

The number of components for each pixel in a bitmap graphics context is specified by a color space (defined by a [CGColorSpaceRef](#) (page 46), which includes RGB, grayscale, and CMYK, and which also may specify a destination color profile). The bitmap graphics context specifies whether the bitmap should contain an alpha channel, and how the bitmap is generated.

Functions by Task

Creating Bitmap Contexts

[CGBitmapContextCreate](#) (page 14)

Creates a bitmap graphics context.

[CGBitmapContextCreateWithData](#) (page 16)

Creates a bitmap graphics context with the specified callback function.

[CGBitmapContextCreateImage](#) (page 15)

Creates and returns a Quartz image from the pixel data in a bitmap graphics context.

Getting Information About Bitmap Contexts

These functions return the values of attributes specified when a bitmap context is created.

[CGBitmapContextGetBitmapInfo](#) (page 18)

Obtains the bitmap information associated with a bitmap graphics context.

[CGBitmapContextGetAlphaInfo](#) (page 18)

Returns the alpha information associated with the context, which indicates how a bitmap context handles the alpha component.

[CGBitmapContextGetBitsPerComponent](#) (page 19)

Returns the bits per component of a bitmap context.

[CGBitmapContextGetBitsPerPixel](#) (page 19)

Returns the bits per pixel of a bitmap context.

[CGBitmapContextGetBytesPerRow](#) (page 20)

Returns the bytes per row of a bitmap context.

[CGBitmapContextGetColorSpace](#) (page 20)

Returns the color space of a bitmap context.

[CGBitmapContextGetData](#) (page 21)

Returns a pointer to the image data associated with a bitmap context.

[CGBitmapContextGetHeight](#) (page 21)

Returns the height in pixels of a bitmap context.

[CGBitmapContextGetWidth](#) (page 21)

Returns the width in pixels of a bitmap context.

Functions

CGBitmapContextCreate

Creates a bitmap graphics context.

```
CGContextRef CGBitmapContextCreate (
    void *data,
    size_t width,
    size_t height,
    size_t bitsPerComponent,
    size_t bytesPerRow,
    CGColorSpaceRef colorspace,
    CGBitmapInfo bitmapInfo
);
```

Parameters

data

A pointer to the destination in memory where the drawing is to be rendered. The size of this memory block should be at least $(\text{bytesPerRow} \times \text{height})$ bytes.

In iOS 4.0 and later, and Mac OS X v10.6 and later, you can pass `NULL` if you want Quartz to allocate memory for the bitmap. This frees you from managing your own memory, which reduces memory leak issues.

width

The width, in pixels, of the required bitmap.

height

The height, in pixels, of the required bitmap.

bitsPerComponent

The number of bits to use for each component of a pixel in memory. For example, for a 32-bit pixel format and an RGB color space, you would specify a value of 8 bits per component. For the list of supported pixel formats, see “Supported Pixel Formats” in the “Graphics Contexts” chapter of *Quartz 2D Programming Guide*.

bytesPerRow

The number of bytes of memory to use per row of the bitmap.

colorspace

The color space to use for the bitmap context. Note that indexed color spaces are not supported for bitmap graphics contexts.

bitmapInfo

Constants that specify whether the bitmap should contain an alpha channel, the alpha channel's relative location in a pixel, and information about whether the pixel components are floating-point or integer values. The constants for specifying the alpha channel information are declared with the `CGImageAlphaInfo` type but can be passed to this parameter safely. You can also pass the other constants associated with the `CGBitmapInfo` type. (See *CGImage Reference* for a description of the `CGBitmapInfo` and `CGImageAlphaInfo` constants.)

For an example of how to specify the color space, bits per pixel, bits per pixel component, and bitmap information using the `CGBitmapContextCreate` function, see “Creating a Bitmap Graphics Context” in the “Graphics Contexts” chapter of *Quartz 2D Programming Guide*.

Return Value

A new bitmap context, or `NULL` if a context could not be created. You are responsible for releasing this object using [CGContextRelease](#) (page 93).

Discussion

When you call this function, Quartz creates a bitmap drawing environment—that is, a bitmap context—to your specifications. When you draw into this context, Quartz renders your drawing as bitmapped data in the specified block of memory.

The pixel format for a new bitmap context is determined by three parameters—the number of bits per component, the color space, and an alpha option (expressed as a [Image Bitmap Information](#) (page 220) constant). The alpha value determines the opacity of a pixel when it is drawn.

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch
GLImageProcessing
PVRTextureLoader
Reflection
TheElements

Declared In

`CGBitmapContext.h`

CGBitmapContextCreateImage

Creates and returns a Quartz image from the pixel data in a bitmap graphics context.

```
CGImageRef CGContextCreateImage (
    CGContextRef c
);
```

Parameters

c
A bitmap graphics context.

Return Value

A CGImage object that contains a snapshot of the bitmap graphics context or `NULL` if the image is not created.

Discussion

The CGImage object returned by this function is created by a copy operation. Subsequent changes to the bitmap graphics context do not affect the contents of the returned image. In some cases the copy operation actually follows copy-on-write semantics, so that the actual physical copy of the bits occur only if the underlying data in the bitmap graphics context is modified. As a consequence, you may want to use the resulting image and release it before you perform additional drawing into the bitmap graphics context. In this way, you can avoid the actual physical copy of the data.

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch
GLImageProcessing
Reflection
TheElements

Declared In

CGBitmapContext.h

CGBitmapContextCreateWithData

Creates a bitmap graphics context with the specified callback function.


```
CGContextRef CGBitmapContextCreateWithData(
    void *data,
    size_t width,
    size_t height,
    size_t bitsPerComponent,
    size_t bytesPerRow,
    CGColorSpaceRef space,
    CGBitmapInfo bitmapInfo,
    CGBitmapContextReleaseDataCallback releaseCallback,
    void *releaseInfo
);
```

Parameters

data

A pointer to the destination in memory where the drawing is to be rendered. The size of this memory block should be at least `(bytesPerRow*height)` bytes.

In iOS 4.0 and later, and Mac OS X v10.6 and later, you can pass `NULL` if you do not care where the data is stored. This frees you from managing your own memory, which reduces memory leak issues. Quartz has more flexibility when it manages data storage for you. For example, it's possible for Quartz to use OpenGL for rendering if it takes care of the memory. Do not pass `NULL` if you are running on earlier operating systems.

width

The width, in pixels, of the required bitmap.

height

The height, in pixels, of the required bitmap.

bitsPerComponent

The number of bits to use for each component of a pixel in memory. For example, for a 32-bit pixel format and an RGB color space, you would specify a value of 8 bits per component. For the list of supported pixel formats, see “Supported Pixel Formats” in the “Graphics Contexts” chapter of *Quartz 2D Programming Guide*.

bytesPerRow

The number of bytes of memory to use per row of the bitmap.

colorspace

The color space to use for the bitmap context. Note that indexed color spaces are not supported for bitmap graphics contexts.

bitmapInfo

Constants that specify whether the bitmap should contain an alpha channel, the alpha channel's relative location in a pixel, and information about whether the pixel components are floating-point or integer values. The constants for specifying the alpha channel information are declared with the `CGImageAlphaInfo` type but can be passed to this parameter safely. You can also pass the other constants associated with the `CGBitmapInfo` type. (See *CGImage Reference* for a description of the `CGBitmapInfo` and `CGImageAlphaInfo` constants.)

For an example of how to specify the color space, bits per pixel, bits per pixel component, and bitmap information using the `CGBitmapContextCreate` function, see “Creating a Bitmap Graphics Context” in the *Graphics Contexts* chapter of *Quartz 2D Programming Guide*.

releaseCallback

The custom release function to call when it is time to release the bitmap data. For the syntax of this function, see the description of the `CGBitmapContextReleaseDataCallback` (page 22) data type. You may specify `NULL` for this parameter.

releaseInfo

A pointer to any data you want passed to the your custom release callback.

Return Value

A new bitmap context, or `NULL` if a context could not be created. You are responsible for releasing this object using [CGContextRelease](#) (page 93).

Discussion

When you call this function, Quartz creates a bitmap drawing environment—that is, a bitmap context—to your specifications. When you draw into this context, Quartz renders your drawing as bitmapped data in the specified block of memory.

The pixel format for a new bitmap context is determined by three parameters—the number of bits per component, the color space, and an alpha option (expressed as a [Image Bitmap Information](#) (page 220) constant). The alpha value determines the opacity of a pixel when it is drawn.

Availability

Available in iOS 4.0 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetAlphaInfo

Returns the alpha information associated with the context, which indicates how a bitmap context handles the alpha component.

```
CGImageAlphaInfo CGBitmapContextGetAlphaInfo (
    CGContextRef c
);
```

Parameters

context

A bitmap context.

Return Value

A bitmap information constant. If the specified context is not a bitmap context, [kCGImageAlphaNone](#) (page 219) is returned. See [CGImageAlphaInfo](#) (renamed to [CGBitmapInfo](#) in Mac OS X v10.4) for more information about values.

Discussion

Every bitmap context contains an attribute that specifies whether the bitmap contains an alpha component, and how it is generated. The alpha component determines the opacity of a pixel when it is drawn.

Availability

Available in iOS 2.0 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetBitmapInfo

Obtains the bitmap information associated with a bitmap graphics context.

```
CGBitmapInfo CGBitmapContextGetBitmapInfo (  
    CGContextRef c  
);
```

Parameters

c

A bitmap graphics context.

Return Value

The bitmap info of the bitmap graphics context or 0 if *c* is not a bitmap graphics context. See *CGImage Reference* for a description of the [Image Bitmap Information](#) (page 220) constants that can be returned.

Discussion

The `CGBitmapInfo` data returned by the function specifies whether the bitmap contains an alpha channel and how the alpha channel is generated, along with whether the components are floating-point or integer.

Availability

Available in iOS 2.0 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetBitsPerComponent

Returns the bits per component of a bitmap context.

```
size_t CGBitmapContextGetBitsPerComponent (  
    CGContextRef c  
);
```

Parameters

context

The bitmap context to examine.

Return Value

The number of bits per component in the specified context, or 0 if the context is not a bitmap context.

Availability

Available in iOS 2.0 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetBitsPerPixel

Returns the bits per pixel of a bitmap context.

```
size_t CGBitmapContextGetBitsPerPixel (  
    CGContextRef c  
);
```

Parameters

context

The bitmap context to examine.

Return Value

The number of bits per pixel in the specified context, or 0 if the context is not a bitmap context.

Availability

Available in iOS 2.0 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetBytesPerRow

Returns the bytes per row of a bitmap context.

```
size_t CGBitmapContextGetBytesPerRow (  
    CGContextRef c  
) ;
```

Parameters

context

The bitmap context to examine.

Return Value

The number of bytes per row of the specified context, or 0 if the context is not a bitmap context.

Availability

Available in iOS 2.0 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetColorSpace

Returns the color space of a bitmap context.

```
CGColorSpaceRef CGBitmapContextGetColorSpace (  
    CGContextRef c  
) ;
```

Parameters

context

The bitmap context to examine.

Return Value

The color space of the specified context, or `NULL` if the context is not a bitmap context. You are responsible for retaining and releasing this object as necessary.

Availability

Available in iOS 2.0 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetData

Returns a pointer to the image data associated with a bitmap context.

```
void * CGBitmapContextGetData (  
    CGContextRef c  
);
```

Parameters

context

The bitmap context to examine.

Return Value

A pointer to the specified bitmap context's image data, or `NULL` if the context is not a bitmap context.

Discussion

If you provided the memory for the bitmap data, you can use this method to get that data pointer. If you passed `NULL` for the data pointer when creating your bitmap context, it is safe to get the data pointer in iOS 4.0 and later and Mac OS X v10.6 and later only. In earlier versions of the operating system, passing `NULL` for the data parameter is not supported and may lead to crashes when attempting to access this data using this function.

Availability

Available in iOS 2.0 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetHeight

Returns the height in pixels of a bitmap context.

```
size_t CGBitmapContextGetHeight (  
    CGContextRef c  
);
```

Parameters

context

The bitmap context to examine.

Return Value

The height in pixels of the specified context, or 0 if the context is not a bitmap context.

Availability

Available in iOS 2.0 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetWidth

Returns the width in pixels of a bitmap context.

```
size_t CGContextGetWidth (
    CGContextRef c
);
```

Parameters

context

The bitmap context to examine.

Return Value

The width in pixels of the specified context, or 0 if the context is not a bitmap context.

Availability

Available in iOS 2.0 and later.

Declared In

CGBitmapContext.h

Data Types

CGBitmapContextReleaseDataCallback

A callback function used to release data associate with the bitmap context.

```
typedef void (*CGBitmapContextReleaseDataCallback)(void *releaseInfo, void *data);
```

Discussion

The *releaseInfo* parameter contains the contextual data that you passed to the [CGBitmapContextCreateWithData](#) (page 16) function. The *data* parameter contains a pointer to the bitmap data for you to release.

Availability

Available in iOS 4.0 and later.

Declared In

CGBitmapContext.h

CGColor Reference

Derived From:	CType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGColor.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGColorRef` opaque type contains a set of components (such as red, green, and blue) that uniquely define a color, and a color space that specifies how those components should be interpreted. Quartz color objects provide a fast and convenient way to manage and set colors, especially colors that are used repeatedly. Quartz drawing operations use color objects for setting fill and stroke colors, managing alpha, and setting color with a pattern.

See also these related references: *CGContext Reference*, *CGColorSpace Reference*, and *CGPattern Reference*.

Functions by Task

Retaining and Releasing Color Objects

[CGColorRelease](#) (page 29)

Decrements the retain count of a Quartz color.

[CGColorRetain](#) (page 30)

Increments the retain count of a Quartz color.

Creating Quartz Colors

[CGColorCreate](#) (page 24)

Creates a Quartz color using a list of intensity values (including alpha) and an associated color space.

[CGColorCreateCopy](#) (page 25)

Creates a copy of an existing Quartz color.

[CGColorCreateCopyWithAlpha](#) (page 25)

Creates a copy of an existing Quartz color, substituting a new alpha value.

[CGColorCreateWithPattern](#) (page 26)

Creates a Quartz color using a list of intensity values (including alpha), a pattern color space, and a pattern.

Getting Information about Quartz Colors

[CGColorEqualToColor](#) (page 26)

Indicates whether two colors are equal.

[CGColorGetAlpha](#) (page 27)

Returns the value of the alpha component associated with a Quartz color.

[CGColorGetColorSpace](#) (page 27)

Returns the color space associated with a Quartz color.

[CGColorGetComponents](#) (page 28)

Returns the values of the color components (including alpha) associated with a Quartz color.

[CGColorGetNumberOfComponents](#) (page 28)

Returns the number of color components (including alpha) associated with a Quartz color.

[CGColorGetPattern](#) (page 29)

Returns the pattern associated with a Quartz color in a pattern color space.

[CGColorGetTypeID](#) (page 29)

Returns the Core Foundation type identifier for a Quartz color data type.

Functions

CGColorCreate

Creates a Quartz color using a list of intensity values (including alpha) and an associated color space.

```
CGColorRef CGColorCreate (
    CGColorSpaceRef colorspace,
    const CGFloat components[]
);
```

Parameters

colorspace

A color space for the new color. Quartz retains this object; upon return, you may safely release it.

components

An array of intensity values describing the color. The array should contain $n+1$ values that correspond to the n color components in the specified color space, followed by the alpha component. Each component value should be in the range appropriate for the color space. Values outside this range will be clamped to the nearest correct value.

Return Value

A new Quartz color. You are responsible for releasing this object using [CGColorRelease](#) (page 29).

Availability**Related Sample Code**

AccelerometerGraph

HazardMap

Declared In

CGColor.h

CGColorCreateCopy

Creates a copy of an existing Quartz color.

```
CGColorRef CGColorCreateCopy (  
    CGColorRef color  
);
```

Parameters*color*

A Quartz color.

Return ValueA copy of the specified color. You are responsible for releasing this object using [CGColorRelease](#) (page 29).**Availability****Declared In**

CGColor.h

CGColorCreateCopyWithAlpha

Creates a copy of an existing Quartz color, substituting a new alpha value.

```
CGColorRef CGColorCreateCopyWithAlpha (  
    CGColorRef color,  
    CGFloat alpha  
);
```

Parameters*color*

The Quartz color to copy.

*alpha*A value that specifies the desired opacity of the copy. Values outside the range $[0, 1]$ are clamped to 0 or 1.**Return Value**A copy of the specified color, using the specified alpha value. You are responsible for releasing this object using [CGColorRelease](#) (page 29).**Availability****Related Sample Code**

avTouch

SpeakHere

Declared In

CGColor.h

CGColorCreateWithPattern

Creates a Quartz color using a list of intensity values (including alpha), a pattern color space, and a pattern.

```
CGColorRef CGColorCreateWithPattern (
    CGColorSpaceRef colorspace,
    CGPatternRef pattern,
    const CGFloat components[]
);
```

Parameters

colorspace

A pattern color space for the new color. Quartz retains the color space you pass in. On return, you may safely release it.

pattern

A pattern for the new color object. Quartz retains the pattern you pass in. On return, you may safely release it.

components

An array of intensity values describing the color. The array should contain $n + 1$ values that correspond to the n color components in the specified color space, followed by the alpha component. Each component value should be in the range appropriate for the color space. Values outside this range will be clamped to the nearest correct value.

Return Value

A new Quartz color. You are responsible for releasing this object using [CGColorRelease](#) (page 29).

Availability

Related Sample Code

QuartzDemo

Declared In

CGColor.h

CGColorEqualToColor

Indicates whether two colors are equal.

```
bool CGColorEqualToColor (
    CGColorRef color1,
    CGColorRef color2
);
```

Parameters

color1

The first Quartz color to compare.

color2

The second Quartz color to compare.

Return Value

A Boolean value that, if `true`, indicates that the specified colors are equal. If the colors are not equal, the value is `false`.

Discussion

Two colors are equal if they have equal color spaces and numerically equal color components.

Availability

Declared In

CGColor.h

CGColorGetAlpha

Returns the value of the alpha component associated with a Quartz color.

```
CGFloat CGColorGetAlpha (
    CGColorRef color
);
```

Parameters

color

A Quartz color.

Return Value

An alpha intensity value in the range `[0,1]`. The value represents the opacity of the color.

Availability

Declared In

CGColor.h

CGColorGetColorSpace

Returns the color space associated with a Quartz color.

```
CGColorSpaceRef CGColorGetColorSpace (
    CGColorRef color
);
```

Parameters

color

A Quartz color.

Return Value

The Quartz color space for the specified color. You are responsible for retaining and releasing it as needed.

Availability

Related Sample Code

QuartzDemo

Declared In

CGColor.h

CGColorGetComponents

Returns the values of the color components (including alpha) associated with a Quartz color.

```
const CGFloat * CGColorGetComponents (
    CGColorRef color
);
```

Parameters*color*

A Quartz color.

Return Value

An array of intensity values for the color components (including alpha) associated with the specified color. The size of the array is one more than the number of components of the color space for the color.

Availability**Related Sample Code**

avTouch

QuartzDemo

SpeakHere

Declared In

CGColor.h

CGColorGetNumberOfComponents

Returns the number of color components (including alpha) associated with a Quartz color.

```
size_t CGColorGetNumberOfComponents (
    CGColorRef color
);
```

Parameters*color*

A Quartz color.

Return Value

The number of color components (including alpha) associated with the specified color. This number is one more than the number of components of the color space for the color.

Availability**Related Sample Code**

avTouch

SpeakHere

Declared In

CGColor.h

CGColorGetPattern

Returns the pattern associated with a Quartz color in a pattern color space.

```
CGPatternRef CGColorGetPattern (  
    CGColorRef color  
);
```

Parameters

color

A Quartz color.

Return Value

The pattern for the specified color. You are responsible for retaining and releasing the pattern as needed.

Availability

Declared In

CGColor.h

CGColorGetTypeID

Returns the Core Foundation type identifier for a Quartz color data type.

```
CTypeID CGColorGetTypeID (  
    void  
);
```

Return Value

The Core Foundation type identifier for CGColorRef.

Availability

Declared In

CGColor.h

CGColorRelease

Decrements the retain count of a Quartz color.

```
void CGColorRelease (  
    CGColorRef color  
);
```

Parameters

color

The Quartz color to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `color` parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Related Sample Code

avTouch

HazardMap

QuartzDemo

SpeakHere

Declared In

CGColor.h

CGColorRetain

Increments the retain count of a Quartz color.

```
CGColorRef CGColorRetain (  
    CGColorRef color  
);
```

Parameters*color*

The Quartz color to retain.

Return Value

The same color you passed in as the *color* parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the *color* parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Declared In

CGColor.h

Data Types

CGColorRef

An opaque type that represents a color used in Quartz 2D drawing.

```
typedef struct CGColor *CGColorRef;
```

Discussion

`CGColorRef` is the fundamental data type used internally by Quartz to represent colors. `CGColor` objects, and the functions that operate on them, provide a fast and convenient way of managing and setting colors directly, especially colors that are reused (such as black for text).

In Mac OS X version 10.3 and later, `CGColorRef` is derived from `CTypeRef` and inherits the properties that all Core Foundation types have in common. For more information, see `CType Reference`.

Availability

Available in iOS 2.0 and later.

Declared In

CGColor.h

CGColorSpace Reference

Derived From:	<i>CType Reference</i>
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGColorSpace.h
Companion guides	Quartz 2D Programming Guide CGColor Reference CGContext Reference

Overview

The `CGColorSpaceRef` opaque type encapsulates color space information that is used to specify how Quartz interprets color information. A color space specifies how color values are interpreted. A color space is multi-dimensional, and each dimension represents a specific color component. For example, the colors in an RGB color space have three dimensions or components—red, green, and blue. The intensity of each component is represented by floating point values—their range and meaning depends on the color space in question.

Different types of devices (scanners, monitors, printers) operate within different color spaces (RGB, CMYK, grayscale). Additionally, two devices of the same type (for example, color displays from different manufacturers) may operate within the same kind of color space, yet still produce a different range of colors, or gamut. Color spaces that are correctly specified ensure that an image has a consistent appearance regardless of the output device.

Quartz supports several kinds of color spaces:

- Calibrated color spaces ensure that colors appear the same when displayed on different devices. The visual appearance of the color is preserved, as far as the capabilities of the device allow.
- Device-dependent color spaces are tied to the system of color representation of a particular device. Device color spaces are not recommended when high-fidelity color preservation is important.
- Special color spaces—indexed and pattern. An indexed color space contains a color table with up to 256 entries and a base color space to which the color table entries are mapped. Each entry in the color table specifies one color in the base color space. A pattern color space is used when stroking or filling with a pattern. Pattern color spaces are supported in Mac OS X version 10.1 and later.

Functions by Task

Creating Device-Independent Color Spaces

[CGColorSpaceCreateCalibratedGray](#) (page 35)

Creates a calibrated grayscale color space.

[CGColorSpaceCreateCalibratedRGB](#) (page 36)

Creates a calibrated RGB color space.

[CGColorSpaceCreateICCBased](#) (page 38)

Creates a device-independent color space that is defined according to the ICC color profile specification.

[CGColorSpaceCreateWithICCProfile](#) (page 42)

Creates an ICC-based color space using the ICC profile contained in the specified data.

[CGColorSpaceCreateLab](#) (page 40)

Creates a device-independent color space that is relative to human color perception, according to the CIE L*a*b* standard.

Creating Generic or Device-Dependent Color Spaces

In Mac OS X v10.4 and later, the color space returned by each of these functions is no longer device-dependent and is replaced by a generic counterpart.

[CGColorSpaceCreateDeviceCMYK](#) (page 37)

Creates a device-dependent CMYK color space.

[CGColorSpaceCreateDeviceGray](#) (page 37)

Creates a device-dependent grayscale color space.

[CGColorSpaceCreateDeviceRGB](#) (page 38)

Creates a device-dependent RGB color space.

Creating Special Color Spaces

[CGColorSpaceCreateIndexed](#) (page 39)

Creates an indexed color space, consisting of colors specified by a color lookup table.

[CGColorSpaceCreatePattern](#) (page 41)

Creates a pattern color space.

[CGColorSpaceCreateWithName](#) (page 42)

Creates a specified type of Quartz color space.

Getting Information About Color Spaces

[CGColorSpaceGetNumberOfComponents](#) (page 44)

Returns the number of color components in a color space.

[CGColorSpaceGetTypeID](#) (page 45)

Returns the Core Foundation type identifier for Quartz color spaces.

[CGColorSpaceGetModel](#) (page 44)

Returns the color space model of the provided color space.

[CGColorSpaceGetBaseColorSpace](#) (page 42)

Returns the base color space of a pattern or indexed color space.

[CGColorSpaceGetColorTableCount](#) (page 43)

Returns the number of entries in the color table of an indexed color space.

[CGColorSpaceGetColorTable](#) (page 43)

Copies the entries in the color table of an indexed color space.

Retaining and Releasing Color Spaces

[CGColorSpaceRelease](#) (page 45)

Decrements the retain count of a color space.

[CGColorSpaceRetain](#) (page 46)

Increments the retain count of a color space.

Functions

CGColorSpaceCreateCalibratedGray

Creates a calibrated grayscale color space.

```
CGColorSpaceRef CGColorSpaceCreateCalibratedGray(  
    const CGFloat whitePoint[3],  
    const CGFloat blackPoint[3],  
    CGFloat gamma  
);
```

Parameters

whitePoint

An array of 3 numbers specifying the tristimulus value, in the CIE 1931 XYZ-space, of the diffuse white point.

blackPoint

An array of 3 numbers specifying the tristimulus value, in CIE 1931 XYZ-space, of the diffuse black point.

gamma

The gamma value appropriate for the imaging device.

Return Value

A new calibrated gray color space. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns `NULL`.

Discussion

Creates a device-independent grayscale color space that represents colors relative to a reference white point. This white point is based on the whitest light that can be generated by the output device. Colors in a device-independent color space should appear the same when displayed on different devices, to the extent that the capabilities of the device allow.

Availability

Available in iOS 2.0 and later.

Declared In

CGColorSpace.h

CGColorSpaceCreateCalibratedRGB

Creates a calibrated RGB color space.

```
CGColorSpaceRef CGColorSpaceCreateCalibratedRGB(  
    const CGFloat whitePoint[3],  
    const CGFloat blackPoint[3],  
    const CGFloat gamma[3],  
    const CGFloat matrix[9]  
);
```

Parameters

whitePoint

An array of 3 numbers specifying the tristimulus value, in the CIE 1931 XYZ-space, of the diffuse white point.

blackPoint

An array of 3 numbers specifying the tristimulus value, in CIE 1931 XYZ-space, of the diffuse black point.

gamma

An array of 3 numbers specifying the gamma for the red, green, and blue components of the color space.

matrix

An array of 9 numbers specifying the linear interpretation of the gamma-modified RGB values of the color space with respect to the final XYZ representation.

Return Value

A new calibrated RGB color space. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns `NULL`.

Discussion

Creates a device-independent RGB color space that represents colors relative to a reference white point. This white point is based on the whitest light that can be generated by the output device. Colors in a device-independent color space should appear the same when displayed on different devices, to the extent that the capabilities of the device allow.

For color spaces that require a detailed gamma, such as the piecewise transfer function used in sRGB or ITU-R BT.709, you may want to use the function [CGColorSpaceCreateICCBased](#) (page 38) instead, because it can accurately represent these gamma curves.

Availability

Available in iOS 2.0 and later.

Declared In

CGColorSpace.h

CGColorSpaceCreateDeviceCMYK

Creates a device-dependent CMYK color space.

```
CGColorSpaceRef CGColorSpaceCreateDeviceCMYK(  
    void  
);
```

Return Value

A device-dependent CMYK color space. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns NULL.

Discussion

In Mac OS X v10.4 and later, this color space is no longer device-dependent and is replaced by the generic counterpart—`kCGColorSpaceGenericCMYK`—described in “Color Space Names”. If you use this function in Mac OS X v10.4 and later, colors are mapped to the generic color spaces. If you want to bypass color matching, use the color space of the destination context.

Colors in a device-dependent color space are not transformed or otherwise modified when displayed on an output device—that is, there is no attempt to maintain the visual appearance of a color. As a consequence, colors in a device color space often appear different when displayed on different output devices. For this reason, device color spaces are not recommended when color preservation is important.

Availability

Available in iOS 2.0 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceCreateDeviceGray

Creates a device-dependent grayscale color space.

```
CGColorSpaceRef CGColorSpaceCreateDeviceGray(  
    void  
);
```

Return Value

A device-dependent gray color space. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns NULL.

Discussion

In Mac OS X v10.4 and later, this color space is no longer device-dependent and is replaced by the generic counterpart—`kCGColorSpaceGenericGray`—described in “Color Space Names”. If you use this function in Mac OS X v10.4 and later, colors are mapped to the generic color spaces. If you want to bypass color matching, use the color space of the destination context.

Colors in a device-dependent color space are not transformed or otherwise modified when displayed on an output device—that is, there is no attempt to maintain the visual appearance of a color. As a consequence, colors in a device color space often appear different when displayed on different output devices. For this reason, device color spaces are not recommended when color preservation is important.

Availability

Available in iOS 2.0 and later.

Related Sample Code

AccelerometerGraph
Reflection
TheElements

Declared In

CGColorSpace.h

CGColorSpaceCreateDeviceRGB

Creates a device-dependent RGB color space.

```
CGColorSpaceRef CGColorSpaceCreateDeviceRGB(  
    void  
);
```

Return Value

A device-dependent RGB color space. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns `NULL`.

Discussion

In Mac OS X v10.4 and later, this color space is no longer device-dependent and is replaced by the generic counterpart—`kCGColorSpaceGenericRGB`—described in “Color Space Names”. If you use this function in Mac OS X v10.4 and later, colors are mapped to the generic color spaces. If you want to bypass color matching, use the color space of the destination context.

Colors in a device-dependent color space are not transformed or otherwise modified when displayed on an output device—that is, there is no attempt to maintain the visual appearance of a color. As a consequence, colors in a device color space often appear different when displayed on different output devices. For this reason, device color spaces are not recommended when color preservation is important.

Availability

Available in iOS 2.0 and later.

Related Sample Code

GLImageProcessing
HazardMap
PVRTextureLoader
QuartzDemo
SpeakHere

Declared In

CGColorSpace.h

CGColorSpaceCreateICCBased

Creates a device-independent color space that is defined according to the ICC color profile specification.

```
CGColorSpaceRef CGColorSpaceCreateICCBased(
    size_t nComponents,
    const CGFloat *range,
    CGDataProviderRef profile,
    CGColorSpaceRef alternate
);
```

Parameters*nComponents*

The number of color components in the color space defined by the ICC profile data. This must match the number of components actually in the ICC profile and must equal 1, 3, or 4.

range

An array of numbers that specify the minimum and maximum valid values of the corresponding color components. The size of the array is two times the number of components. If $c[k]$ is the k th color component, the valid range is $\text{range}[2*k] \leq c[k] \leq \text{range}[2*k+1]$.

profile

A data provider that supplies the ICC profile.

alternateSpace

An alternate color space to use in case the ICC profile is not supported. The alternate color space must have *nComponents* color components. You must supply an alternate color space. If this parameter is `NULL`, then the function returns `NULL`.

Return Value

A new ICC-based color space object. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns `NULL`.

Discussion

This function creates an ICC-based color space from an ICC color profile, as defined by the International Color Consortium. ICC profiles define the reproducible color gamut (the range of colors supported by a device) and other characteristics of a particular output device, providing a way to accurately transform the color space of one device to the color space of another. The ICC profile is usually provided by the manufacturer of the device. Additionally, some color monitors and printers contain electronically embedded ICC profile information, as do some bitmap formats such as TIFF. Colors in a device-independent color space should appear the same when displayed on different devices, to the extent that the capabilities of the device allow.

You may want to use this function for a color space that requires a detailed gamma, such as the piecewise transfer function used in sRGB or ITU-R BT.709, because this function can accurately represent these gamma curves.

Availability

Available in iOS 2.0 and later.

See Also

[CGColorSpaceCreateWithICCProfile](#) (page 42)

Declared In

`CGColorSpace.h`

CGColorSpaceCreateIndexed

Creates an indexed color space, consisting of colors specified by a color lookup table.

```
CGColorSpaceRef CGColorSpaceCreateIndexed(
    CGColorSpaceRef baseSpace,
    size_t lastIndex,
    const unsigned char *colorTable
);
```

Parameters*baseSpace*

The color space on which the color table is based.

lastIndex

The maximum valid index value for the color table. The value must be less than or equal to 255.

colorTable

An array of $m \times (\text{lastIndex} + 1)$ bytes, where m is the number of color components in the base color space. Each byte is an unsigned integer in the range 0 to 255 that is scaled to the range of the corresponding color component in the base color space.

Return Value

A new indexed color space object. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns `NULL`.

Discussion

An indexed color space contains a color table with up to 255 entries, and a base color space to which the color table entries are mapped. Each entry in the color table specifies one color in the base color space. A value in an indexed color space is treated as an index into the color table of the color space. The data in the table is in meshed format. (For example, for an RGB color space the values are R, G, B, R, G, B, and so on.)

Availability

Available in iOS 2.0 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceCreateLab

Creates a device-independent color space that is relative to human color perception, according to the CIE $L^*a^*b^*$ standard.

```
CGColorSpaceRef CGColorSpaceCreateLab(
    const CGFloat whitePoint[3],
    const CGFloat blackPoint[3],
    const CGFloat range[4]
);
```

Parameters*whitePoint*

An array of 3 numbers that specify the tristimulus value, in the CIE 1931 XYZ-space, of the diffuse white point.

blackPoint

An array of 3 numbers that specify the tristimulus value, in CIE 1931 XYZ-space, of the diffuse black point.

range

An array of 4 numbers that specify the range of valid values for the a* and b* components of the color space. The a* component represents values running from green to red, and the b* component represents values running from blue to yellow.

Return Value

A new L*a*b* color space. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns `NULL`.

Discussion

The CIE L*a*b* space is a nonlinear transformation of the Munsell color notation system (a system which specifies colors by hue, value, and saturation—or “chroma”—values), designed to match perceived color difference with quantitative distance in color space. The L* component represents the lightness value, the a* component represents values running from green to red, and the b* component represents values running from blue to yellow. This roughly corresponds to the way the human brain is thought to decode colors. Colors in a device-independent color space should appear the same when displayed on different devices, to the extent that the capabilities of the device allow.

Availability

Available in iOS 2.0 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceCreatePattern

Creates a pattern color space.

```
CGColorSpaceRef CGColorSpaceCreatePattern(
    CGColorSpaceRef baseSpace
);
```

Parameters

baseSpace

For masking patterns, the underlying color space that specifies the colors to be painted through the mask. For colored patterns, you should pass `NULL`.

Return Value

A new pattern color space. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns `NULL`.

Discussion

For information on creating and using patterns, see *Quartz 2D Programming Guide* and *CGPattern Reference*. Quartz retains the color space you pass in. Upon return, you may safely release it by calling [CGColorSpaceRelease](#) (page 45).

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

`CGColorSpace.h`

CGColorSpaceCreateWithICCProfile

Creates an ICC-based color space using the ICC profile contained in the specified data.

```
CGColorSpaceRef CGColorSpaceCreateWithICCProfile(  
    CFDataRef data  
);
```

Parameters

data

The data containing the ICC profile to set for the new color space.

Return Value

A new color space based on the specified profile.

Availability

Available in iOS 4.0 and later.

See Also

[CGColorSpaceCreateICCBased](#) (page 38)

Declared In

CGColorSpace.h

CGColorSpaceCreateWithName

Creates a specified type of Quartz color space.

```
CGColorSpaceRef CGColorSpaceCreateWithName(  
    CFStringRef name  
);
```

Parameters

name

A color space name. See “Color Space Names” for a list of the valid Quartz-defined names.

Return Value

A new generic color space. You are responsible for releasing this object by calling [CGColorSpaceRelease](#) (page 45). If unsuccessful, returns NULL.

Discussion

You can use this function to create a generic color space. For more information, see “Color Space Names”.

Availability

Available in iOS 2.0 and later.

Declared In

CGColorSpace.h

CGColorSpaceGetBaseColorSpace

Returns the base color space of a pattern or indexed color space.

```
CGColorSpace CGColorSpaceGetBaseColorSpace(  
    CGColorSpaceRef space  
);
```

Parameters*space*

A color space object for a pattern or indexed color space.

Return Value

The base color space if the *space* parameter is a pattern or indexed color space; otherwise, `NULL`.

Availability

Available in iOS 2.0 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceGetColorTable

Copies the entries in the color table of an indexed color space.

```
void CGColorSpaceGetColorTable(  
    CGColorSpaceRef space,  
    uint8_t *table);  
);
```

Parameters*space*

A color space object for an indexed color space.

table

The array pointed to by *table* should be at least as large as the number of entries in the color table.

On output, the array contains the table data in the same format as that passed to

[CGColorSpaceCreateIndexed](#) (page 39).

Discussion

This function does nothing if the color space is not an indexed color space. To determine whether a color space is an indexed color space, call the function [CGColorSpaceGetModel](#) (page 44).

Availability

Available in iOS 2.0 and later.

See Also

[CGColorSpaceGetColorTableCount](#) (page 43)

Declared In

`CGColorSpace.h`

CGColorSpaceGetColorTableCount

Returns the number of entries in the color table of an indexed color space.

```
size_t CGColorSpaceGetColorTableCount(  
    CGColorSpaceRef space  
);
```

Parameters

space

A color space object for an indexed color space.

Return Value

The number of entries in the color table of the *space* parameter if the color space is an indexed color space; otherwise, returns 0.

Availability

Available in iOS 2.0 and later.

See Also

[CGColorSpaceGetColorTable](#) (page 43)

Declared In

CGColorSpace.h

CGColorSpaceGetModel

Returns the color space model of the provided color space.

```
CGColorSpaceModel CGColorSpaceGetModel(  
    CGColorSpaceRef space  
);
```

Parameters

space

A color space object.

Return Value

One of the constants described in “[Color Space Models](#)” (page 47).

Availability

Available in iOS 2.0 and later.

Related Sample Code

GLImageProcessing

QuartzDemo

Declared In

CGColorSpace.h

CGColorSpaceGetNumberOfComponents

Returns the number of color components in a color space.

```
size_t CGColorSpaceGetNumberOfComponents(  
    CGColorSpaceRef cs  
);
```

Parameters*cs*

The Quartz color space to examine.

Return Value

The number of color components in the specified color space, not including the alpha value. For example, for an RGB color space, `CGColorSpaceGetNumberOfComponents` returns a value of 3.

Discussion

A color space defines an n-dimensional space whose dimensions (or components) represent intensity values. For example, you specify colors in RGB space as three intensity values: red, green, and blue. You can use the `CGColorSpaceGetNumberOfComponents` function to obtain the number of components in a given color space.

Availability

Available in iOS 2.0 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceGetTypeID

Returns the Core Foundation type identifier for Quartz color spaces.

```
CTypeID CGColorSpaceGetTypeID(  
    void  
);
```

Return Value

The identifier for the opaque type [CGColorSpaceRef](#) (page 46).

Availability

Available in iOS 2.0 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceRelease

Decrements the retain count of a color space.

```
void CGColorSpaceRelease(  
    CGColorSpaceRef cs  
);
```

Parameters*cs*

The Quartz color space to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `cs` parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Related Sample Code

AccelerometerGraph

PVRTextureLoader

QuartzDemo

Reflection

TheElements

Declared In

CGColorSpace.h

CGColorSpaceRetain

Increments the retain count of a color space.

```
CGColorSpaceRef CGColorSpaceRetain(  
    CGColorSpaceRef cs  
);
```

Parameters

cs

The Quartz color space to retain.

Return Value

The same color space you passed in as the *cs* parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the *cs* parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Declared In

CGColorSpace.h

Data Types

CGColorSpaceRef

An opaque type that encapsulates color space information.

```
typedef struct CGColorSpace *CGColorSpaceRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGColorSpace.h

Constants

Color Space Models

Models for color spaces.

```
enum CGColorSpaceModel {
    kCGColorSpaceModelUnknown = -1,
    kCGColorSpaceModelMonochrome,
    kCGColorSpaceModelRGB,
    kCGColorSpaceModelCMYK,
    kCGColorSpaceModelLab,
    kCGColorSpaceModelDeviceN,
    kCGColorSpaceModelIndexed,
    kCGColorSpaceModelPattern
};
typedef int32_t CGColorSpaceModel;
```

Constants

`kCGColorSpaceModelUnknown`

An unknown color space model.

Available in iOS 2.0 and later.

Declared in `CGColorSpace.h`.

`kCGColorSpaceModelMonochrome`

A monochrome color space model.

Available in iOS 2.0 and later.

Declared in `CGColorSpace.h`.

`kCGColorSpaceModelRGB`

An RGB color space model.

Available in iOS 2.0 and later.

Declared in `CGColorSpace.h`.

`kCGColorSpaceModelCMYK`

A CMYK color space model.

Available in iOS 2.0 and later.

Declared in `CGColorSpace.h`.

`kCGColorSpaceModelLab`

A Lab color space model.

Available in iOS 2.0 and later.

Declared in `CGColorSpace.h`.

`kCGColorSpaceModelDeviceN`

A DeviceN color space model.

Available in iOS 2.0 and later.

Declared in `CGColorSpace.h`.

`kCGColorSpaceModelIndexed`
 An indexed color space model.
 Available in iOS 2.0 and later.
 Declared in `CGColorSpace.h`.

`kCGColorSpaceModelPattern`
 A pattern color space model.
 Available in iOS 2.0 and later.
 Declared in `CGColorSpace.h`.

Color Rendering Intents

Handling options for colors that are not located within the destination color space of a graphics context.

```
enum CGColorRenderingIntent {
    kCGRenderingIntentDefault,
    kCGRenderingIntentAbsoluteColorimetric,
    kCGRenderingIntentRelativeColorimetric,
    kCGRenderingIntentPerceptual,
    kCGRenderingIntentSaturation
};
typedef enum CGColorRenderingIntent CGColorRenderingIntent;
```

Constants

`kCGRenderingIntentDefault`
 The default rendering intent for the graphics context.
 Available in iOS 2.0 and later.
 Declared in `CGColorSpace.h`.

`kCGRenderingIntentAbsoluteColorimetric`
 Map colors outside of the gamut of the output device to the closest possible match inside the gamut of the output device. This can produce a clipping effect, where two different color values in the gamut of the graphics context are mapped to the same color value in the output device's gamut. Unlike the relative colorimetric, absolute colorimetric does not modify colors inside the gamut of the output device.
 Available in iOS 2.0 and later.
 Declared in `CGColorSpace.h`.

`kCGRenderingIntentRelativeColorimetric`
 Map colors outside of the gamut of the output device to the closest possible match inside the gamut of the output device. This can produce a clipping effect, where two different color values in the gamut of the graphics context are mapped to the same color value in the output device's gamut. The relative colorimetric shifts all colors (including those within the gamut) to account for the difference between the white point of the graphics context and the white point of the output device.
 Available in iOS 2.0 and later.
 Declared in `CGColorSpace.h`.

`kCGColorRenderingIntentPerceptual`

Preserve the visual relationship between colors by compressing the gamut of the graphics context to fit inside the gamut of the output device. Perceptual intent is good for photographs and other complex, detailed images.

Available in iOS 2.0 and later.

Declared in `CGColorSpace.h`.

`kCGColorRenderingIntentSaturation`

Preserve the relative saturation value of the colors when converting into the gamut of the output device. The result is an image with bright, saturated colors. Saturation intent is good for reproducing images with low detail, such as presentation charts and graphs.

Available in iOS 2.0 and later.

Declared in `CGColorSpace.h`.

Discussion

The rendering intent specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context. It determines the exact method used to map colors from one color space to another. If you do not explicitly set the rendering intent by calling the function `CGContextSetRenderingIntent` (page 114), the graphics context uses the relative colorimetric rendering intent, except when drawing sampled images.

CGContext Reference

Derived From:	<i>CType Reference</i>
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGContext.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGContextRef` opaque type represents a Quartz 2D drawing destination. A graphics context contains drawing parameters and all device-specific information needed to render the paint on a page to the destination, whether the destination is a window in an application, a bitmap image, a PDF document, or a printer. You can obtain a graphics context by using Quartz graphics context creation functions or by using higher-level functions provided in the Carbon, Cocoa, or Printing frameworks. Quartz provides creation functions for various flavors of Quartz graphics contexts including bitmap images and PDF. The Cocoa framework provides functions for obtaining window graphics contexts. The Printing framework provides functions that obtain a graphics context appropriate for the destination printer.

Functions by Task

Managing Graphics Contexts

[CGContextFlush](#) (page 87)

Forces all pending drawing operations in a window context to be rendered immediately to the destination device.

[CGContextGetTypeID](#) (page 91)

Returns the type identifier for Quartz graphics contexts.

[CGContextRelease](#) (page 93)

Decrements the retain count of a graphics context.

[CGContextRetain](#) (page 95)

Increments the retain count of a graphics context.

[CGContextSynchronize](#) (page 131)

Marks a window context for update.

Saving and Restoring the Current Graphics State

[CGContextSaveGState](#) (page 96)

Pushes a copy of the current graphics state onto the graphics state stack for the context.

[CGContextRestoreGState](#) (page 94)

Sets the current graphics state to the state most recently saved.

Getting and Setting Graphics State Parameters

[CGContextGetInterpolationQuality](#) (page 88)

Returns the current level of interpolation quality for a graphics context.

[CGContextSetFlatness](#) (page 107)

Sets the accuracy of curved paths in a graphics context.

[CGContextSetInterpolationQuality](#) (page 110)

Sets the level of interpolation quality for a graphics context.

[CGContextSetLineCap](#) (page 111)

Sets the style for the endpoints of lines drawn in a graphics context.

[CGContextSetLineDash](#) (page 111)

Sets the pattern for dashed lines in a graphics context.

[CGContextSetLineJoin](#) (page 112)

Sets the style for the joins of connected lines in a graphics context.

[CGContextSetLineWidth](#) (page 112)

Sets the line width for a graphics context.

[CGContextSetMiterLimit](#) (page 113)

Sets the miter limit for the joins of connected lines in a graphics context.

[CGContextSetPatternPhase](#) (page 114)

Sets the pattern phase of a context.

[CGContextSetFillPattern](#) (page 106)

Sets the fill pattern in the specified graphics context.

[CGContextSetRenderingIntent](#) (page 114)

Sets the rendering intent in the current graphics state.

[CGContextSetShouldAntialias](#) (page 118)

Sets anti-aliasing on or off for a graphics context.

[CGContextSetStrokePattern](#) (page 121)

Sets the stroke pattern in the specified graphics context.

[CGContextSetBlendMode](#) (page 101)

Sets how Quartz composites sample values for a graphics context.

[CGContextSetAllowsAntialiasing](#) (page 98)

Sets whether or not to allow anti-aliasing for a graphics context.

[CGContextSetAllowsFontSmoothing](#) (page 99)

Sets whether or not to allow font smoothing for a graphics context.

[CGContextSetShouldSmoothFonts](#) (page 118)

Enables or disables font smoothing in a graphics context.

[CGContextSetAllowsFontSubpixelPositioning](#) (page 100)

Sets whether or not to allow subpixel positioning for a graphics context

[CGContextSetShouldSubpixelPositionFonts](#) (page 119)

Enables or disables subpixel positioning in a graphics context.

[CGContextSetAllowsFontSubpixelQuantization](#) (page 100)

Sets whether or not to allow subpixel quantization for a graphics context

[CGContextSetShouldSubpixelQuantizeFonts](#) (page 119)

Enables or disables subpixel quantization in a graphics context.

Constructing Paths

These functions are used to define the geometry of the current path. For more information on how paths are defined, see *CGPath Reference*.

[CGContextAddArc](#) (page 58)

Adds an arc of a circle to the current path, possibly preceded by a straight line segment

[CGContextAddArcToPoint](#) (page 60)

Adds an arc of a circle to the current path, using a radius and tangent points.

[CGContextAddCurveToPoint](#) (page 61)

Appends a cubic Bézier curve from the current point, using the provided control points and end point

.

[CGContextAddLines](#) (page 62)

Adds a sequence of connected straight-line segments to the current path.

[CGContextAddLineToPoint](#) (page 63)

Appends a straight line segment from the current point to the provided point .

[CGContextAddPath](#) (page 64)

Adds a previously created Quartz path object to the current path in a graphics context.

[CGContextCopyPath](#) (page 77)

Returns a Quartz path object built from the current path information in a graphics context.

[CGContextAddQuadCurveToPoint](#) (page 64)

Appends a quadratic Bézier curve from the current point, using a control point and an end point you specify.

[CGContextAddRect](#) (page 65)

Adds a rectangular path to the current path.

[CGContextAddRects](#) (page 66)

Adds a set rectangular paths to the current path.

[CGContextBeginPath](#) (page 67)

Creates a new empty path in a graphics context.

[CGContextClosePath](#) (page 73)

Closes and terminates the current path's subpath.

[CGContextMoveToPoint](#) (page 92)

Begins a new subpath at the point you specify.

[CGContextAddEllipseInRect](#) (page 62)

Adds an ellipse that fits inside the specified rectangle.

Painting Paths

These functions are used to stroke along or fill in the current path.

[CGContextClearRect](#) (page 69)

Paints a transparent rectangle.

[CGContextDrawPath](#) (page 79)

Draws the current path using the provided drawing mode.

[CGContextEOFillPath](#) (page 84)

Paints the area within the current path, using the even-odd fill rule.

[CGContextFillPath](#) (page 85)

Paints the area within the current path, using the nonzero winding number rule.

[CGContextFillRect](#) (page 86)

Paints the area contained within the provided rectangle, using the fill color in the current graphics state.

[CGContextFillRects](#) (page 87)

Paints the areas contained within the provided rectangles, using the fill color in the current graphics state.

[CGContextFillEllipseInRect](#) (page 85)

Paints the area of the ellipse that fits inside the provided rectangle, using the fill color in the current graphics state.

[CGContextStrokePath](#) (page 129)

Paints a line along the current path.

[CGContextStrokeRect](#) (page 130)

Paints a rectangular path.

[CGContextStrokeRectWithWidth](#) (page 131)

Paints a rectangular path, using the specified line width.

[CGContextReplacePathWithStrokedPath](#) (page 94)

Replaces the path in the graphics context with the stroked version of the path.

[CGContextStrokeEllipseInRect](#) (page 128)

Strokes an ellipse that fits inside the specified rectangle.

[CGContextStrokeLineSegments](#) (page 129)

Strokes a sequence of line segments.

Getting Information About Paths

[CGContextIsPathEmpty](#) (page 91)

Indicates whether the current path contains any subpaths.

[CGContextGetPathCurrentPoint](#) (page 89)

Returns the current point in a non-empty path.

[CGContextGetPathBoundingBox](#) (page 89)

Returns the smallest rectangle that contains the current path.

[CGContextPathContainsPoint](#) (page 93)

Checks to see whether the specified point is contained in the current path.

Modifying Clipping Paths

[CGContextClip](#) (page 70)

Modifies the current clipping path, using the nonzero winding number rule.

[CGContextEOClip](#) (page 84)

Modifies the current clipping path, using the even-odd rule.

[CGContextClipToRect](#) (page 72)

Sets the clipping path to the intersection of the current clipping path with the area defined by the specified rectangle.

[CGContextClipToRects](#) (page 72)

Sets the clipping path to the intersection of the current clipping path with the region defined by an array of rectangles.

[CGContextGetClipBoundingBox](#) (page 88)

Returns the bounding box of a clipping path.

[CGContextClipToMask](#) (page 71)

Maps a mask into the specified rectangle and intersects it with the current clipping area of the graphics context.

Setting Color, Color Space, and Shadow Values

[CGContextSetAlpha](#) (page 101)

Sets the opacity level for objects drawn in a graphics context.

[CGContextSetCMYKFillColor](#) (page 102)

Sets the current fill color to a value in the DeviceCMYK color space.

[CGContextSetFillColor](#) (page 104)

Sets the current fill color.

[CGContextSetCMYKStrokeColor](#) (page 103)

Sets the current stroke color to a value in the DeviceCMYK color space.

[CGContextSetFillColorSpace](#) (page 105)

Sets the fill color space in a graphics context.

[CGContextSetFillColorWithColor](#) (page 106)

Sets the current fill color in a graphics context, using a Quartz color.

[CGContextSetGrayFillColor](#) (page 108)

Sets the current fill color to a value in the DeviceGray color space.

[CGContextSetGrayStrokeColor](#) (page 109)

Sets the current stroke color to a value in the DeviceGray color space.

[CGContextSetRGBFillColor](#) (page 115)

Sets the current fill color to a value in the DeviceRGB color space.

[CGContextSetRGBStrokeColor](#) (page 116)

Sets the current stroke color to a value in the DeviceRGB color space.

[CGContextSetShadow](#) (page 116)

Enables shadowing in a graphics context.

[CGContextSetShadowWithColor](#) (page 117)

Enables shadowing with color a graphics context.

[CGContextSetStrokeColor](#) (page 120)

Sets the current stroke color.

[CGContextSetStrokeColorSpace](#) (page 120)

Sets the stroke color space in a graphics context.

[CGContextSetStrokeColorWithColor](#) (page 121)

Sets the current stroke color in a context, using a Quartz color.

Transforming User Space

These functions allow you to examine and change the current transformation matrix (CTM) in a graphics context.

[CGContextConcatCTM](#) (page 74)

Transforms the user coordinate system in a context using a specified matrix.

[CGContextGetCTM](#) (page 88)

Returns the current transformation matrix.

[CGContextRotateCTM](#) (page 95)

Rotates the user coordinate system in a context.

[CGContextScaleCTM](#) (page 97)

Changes the scale of the user coordinate system in a context.

[CGContextTranslateCTM](#) (page 132)

Changes the origin of the user coordinate system in a context.

Using Transparency Layers

[CGContextBeginTransparencyLayer](#) (page 68)

Begins a transparency layer.

[CGContextBeginTransparencyLayerWithRect](#) (page 69)

Begins a transparency layer whose contents are bounded by the specified rectangle.

[CGContextEndTransparencyLayer](#) (page 83)

Ends a transparency layer.

Drawing an Image to a Graphics Context

[CGContextDrawTiledImage](#) (page 82)

Repeatedly draws an image, scaled to the provided rectangle, to fill the current clip region.

[CGContextDrawImage](#) (page 78)

Draws an image into a graphics context.

Drawing PDF Content to a Graphics Context

[CGContextDrawPDFPage](#) (page 80)

Draws a page in the current user space of a PDF context.

Drawing With a Gradient

[CGContextDrawLinearGradient](#) (page 78)

Paints a gradient fill that varies along the line defined by the provided starting and ending points.

[CGContextDrawRadialGradient](#) (page 80)

Paints a gradient fill that varies along the area defined by the provided starting and ending circles.

Drawing With a Shading

[CGContextDrawShading](#) (page 81)

Fills the clipping path of a context with the specified shading.

Setting Up a Page-Based Graphics Context

[CGContextBeginPage](#) (page 67)

Starts a new page in a page-based graphics context.

[CGContextEndPage](#) (page 83)

Ends the current page in a page-based graphics context.

Drawing Glyphs

[CGContextShowGlyphs](#) (page 124)

Displays an array of glyphs at the current text position.

[CGContextShowGlyphsAtPoint](#) (page 124)

Displays an array of glyphs at a position you specify.

[CGContextShowGlyphsWithAdvances](#) (page 126)

Draws an array of glyphs with varying offsets.

[CGContextShowGlyphsAtPositions](#) (page 125)

Draws glyphs at the provided position.

Drawing Text

[CGContextGetTextMatrix](#) (page 90)

Returns the current text matrix.

[CGContextGetTextPosition](#) (page 90)

Returns the location at which text is drawn.

[CGContextSelectFont](#) (page 98)

Sets the font and font size in a graphics context.

[CGContextSetCharacterSpacing](#) (page 102)

Sets the current character spacing.

[CGContextSetFont](#) (page 107)

Sets the platform font in a graphics context.

[CGContextSetFontSize](#) (page 108)

Sets the current font size.

[CGContextSetTextDrawingMode](#) (page 122)

Sets the current text drawing mode.

[CGContextSetTextMatrix](#) (page 123)

Sets the current text matrix.

[CGContextSetTextPosition](#) (page 123)

Sets the location at which text is drawn.

[CGContextShowText](#) (page 127)

Displays a character array at the current text position, a point specified by the current text matrix.

[CGContextShowTextAtPoint](#) (page 127)

Displays a character string at a position you specify.

Converting Between Device Space and User Space

[CGContextGetUserSpaceToDeviceSpaceTransform](#) (page 91)

Returns an affine transform that maps user space coordinates to device space coordinates.

[CGContextConvertPointToDeviceSpace](#) (page 74)

Returns a point that is transformed from user space coordinates to device space coordinates.

[CGContextConvertPointToUserSpace](#) (page 75)

Returns a point that is transformed from device space coordinates to user space coordinates.

[CGContextConvertSizeToDeviceSpace](#) (page 76)

Returns a size that is transformed from user space coordinates to device space coordinates.

[CGContextConvertSizeToUserSpace](#) (page 77)

Returns a size that is transformed from device space coordinates to user space coordinates.

[CGContextConvertRectToDeviceSpace](#) (page 75)

Returns a rectangle that is transformed from user space coordinate to device space coordinates.

[CGContextConvertRectToUserSpace](#) (page 76)

Returns a rectangle that is transformed from device space coordinate to user space coordinates.

Functions

CGContextAddArc

Adds an arc of a circle to the current path, possibly preceded by a straight line segment

```
void CGContextAddArc (
    CGContextRef c,
    CGFloat x,
    CGFloat y,
    CGFloat radius,
    CGFloat startAngle,
    CGFloat endAngle,
    int clockwise
);
```

Parameters*c*

A graphics context.

x

The x-value, in user space coordinates, for the center of the arc.

y

The y-value, in user space coordinates, for the center of the arc.

radius

The radius of the arc, in user space coordinates.

startAngle

The angle to the starting point of the arc, measured in radians from the positive x-axis.

endAngle

The angle to the end point of the arc, measured in radians from the positive x-axis.

clockwise

Specify 1 to create a clockwise arc or 0 to create a counterclockwise arc.

Discussion

An arc is a segment of a circle with radius *r* centered at a point (*x*, *y*). When you call this function, you provide the center point, radius, and two angles in radians. Quartz uses this information to determine the end points of the arc, and then approximates the new arc using a sequence of cubic Bézier curves. The *clockwise* parameter determines the direction in which the arc is created; the actual direction of the final path is dependent on the current transformation matrix of the graphics context. For example, on iOS, a `UIView` flips the Y-coordinate by scaling the Y values by -1. In a flipped coordinate system, specifying a clockwise arc results in a counterclockwise arc after the transformation is applied.

If the current path already contains a subpath, Quartz adds a line connecting the current point to the starting point of the arc. If the current path is empty, Quartz creates a new new subpath with a starting point set to the starting point of the arc.

The ending point of the arc becomes the new current point of the path.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextAddArcToPoint](#) (page 60)

Related Sample Code

GLImageProcessing

HeadsUpUI

QuartzDemo

Declared In
CGContext.h

CGContextAddArcToPoint

Adds an arc of a circle to the current path, using a radius and tangent points.

```
void CGContextAddArcToPoint (
    CGContextRef c,
    CGFloat x1,
    CGFloat y1,
    CGFloat x2,
    CGFloat y2,
    CGFloat radius
);
```

Parameters

c

A graphics context whose current path is not empty.

x1

The x-value, in user space coordinates, for the end point of the first tangent line. The first tangent line is drawn from the current point to (x1,y1).

y1

The y-value, in user space coordinates, for the end point of the first tangent line. The first tangent line is drawn from the current point to (x1,y1).

x2

The x-value, in user space coordinates, for the end point of the second tangent line. The second tangent line is drawn from (x1,y1) to (x2,y2).

y2

The y-value, in user space coordinates, for the end point of the second tangent line. The second tangent line is drawn from (x1,y1) to (x2,y2).

radius

The radius of the arc, in user space coordinates.

Discussion

This function uses a sequence of cubic Bézier curves to create an arc that is tangent to the line from the current point to (x1,y1) and to the line from (x1,y1) to (x2,y2). The start and end points of the arc are located on the first and second tangent lines, respectively. The start and end points of the arc are also the “tangent points” of the lines.

If the current point and the first tangent point of the arc (the starting point) are not equal, Quartz appends a straight line segment from the current point to the first tangent point.

If the current point and the first tangent point of the arc (the starting point) are not equal, Quartz appends a straight line segment from the current point to the first tangent point.

The ending point of the arc becomes the new current point of the path.

Availability

Available in iOS 2.0 and later.

See Also[CGContextAddArc](#) (page 58)[CGContextAddArcToPoint](#) (page 60)**Related Sample Code**

GLImageProcessing

QuartzDemo

Declared In

CGContext.h

CGContextAddCurveToPoint

Appends a cubic Bézier curve from the current point, using the provided control points and end point .

```
void CGContextAddCurveToPoint (
    CGContextRef c,
    CGFloat cp1x,
    CGFloat cp1y,
    CGFloat cp2x,
    CGFloat cp2y,
    CGFloat x,
    CGFloat y
);
```

Parameters*c*

A graphics context whose current path is not empty.

cp1x

The x-value, in user space coordinates, for the first control point of the curve.

cp1y

The y-value, in user space coordinates, for the first control point of the curve.

cp2x

The x-value, in user space coordinates, for the second control point of the curve.

cp2y

The y-value, in user space coordinates, for the second control point of the curve.

x

The x-value, in user space coordinates, at which to end the curve.

y

The y-value, in user space coordinates, at which to end the curve.

Discussion

This function appends a cubic curve to the current path. On return, the current point is set to the end point of that segment.

Availability

Available in iOS 2.0 and later.

See Also[CGContextAddQuadCurveToPoint](#) (page 64)[CGContextAddArcToPoint](#) (page 60)

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextAddEllipseInRect

Adds an ellipse that fits inside the specified rectangle.

```
void CGContextAddEllipseInRect (
    CGContextRef context,
    CGRect rect
);
```

Parameters*context*

A graphics context.

rect

A rectangle that defines the area for the ellipse to fit in.

Discussion

The ellipse is approximated by a sequence of Bézier curves. Its center is the midpoint of the rectangle defined by the *rect* parameter. If the rectangle is square, then the ellipse is circular with a radius equal to one-half the width (or height) of the rectangle. If the *rect* parameter specifies a rectangular shape, then the major and minor axes of the ellipse are defined by the width and height of the rectangle.

The ellipse forms a complete subpath of the path—that is, the ellipse drawing starts with a move-to operation and ends with a close-subpath operation, with all moves oriented in the clockwise direction.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextAddLines

Adds a sequence of connected straight-line segments to the current path.

```
void CGContextAddLines (
    CGContextRef c,
    const CGPoint points[],
    size_t count
);
```

Parameters*c*

A graphics context .

points

An array of values that specify the start and end points of the line segments to draw. Each point in the array specifies a position in user space. The first point in the array specifies the initial starting point.

count

The number of elements in the *points* array.

Discussion

This is a convenience function that adds a sequence of connected line segments to a path, using the following operation:

```
CGContextMoveToPoint (c, points[0].x, points[0].y);
for (k = 1; k < count; k++) {
    CGContextAddLineToPoint (c, points[k].x, points[k].y);
}
```

Availability

Available in iOS 2.0 and later.

See Also

[CGContextAddLineToPoint](#) (page 63)

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextAddLineToPoint

Appends a straight line segment from the current point to the provided point .

```
void CGContextAddLineToPoint (
    CGContextRef c,
    CGFloat x,
    CGFloat y
);
```

Parameters

c

A graphics context whose current path is not empty.

x

The x-value, in user space coordinates, for the end of the line segment.

y

The y-value, in user space coordinates, for the end of the line segment.

Discussion

After adding the line segment, the current point is set to the endpoint of the line segment.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextAddLines](#) (page 62)

Related Sample Code

AccelerometerGraph

GKRocket

GLImageProcessing

QuartzDemo

Teslameter

Declared In

CGContext.h

CGContextAddPath

Adds a previously created Quartz path object to the current path in a graphics context.

```
void CGContextAddPath (
    CGContextRef context,
    CGPathRef path
);
```

Parameters*context*

A graphics context .

path

A previously created Quartz path object. See *CGPath Reference*.

Discussion

If the source path is non-empty, then its path elements are appended in order onto the current path. Quartz applies the current transformation matrix (CTM) to the points before adding them to the path.

After the call completes, the start point and current point of the path are those of the last subpath in *path*.

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch

Breadcrumb

WeatherMap

Declared In

CGContext.h

CGContextAddQuadCurveToPoint

Appends a quadratic Bézier curve from the current point, using a control point and an end point you specify.


```
void CGContextAddQuadCurveToPoint (
    CGContextRef c,
    CGFloat cpx,
    CGFloat cpy,
    CGFloat x,
    CGFloat y
);
```

Parameters*c*

A graphics context whose current path is not empty.

cpx

The x-coordinate of the user space for the control point of the curve.

cpy

The y-coordinate of the user space for the control point of the curve.

x

The x-coordinate of the user space at which to end the curve.

y

The y-coordinate of the user space at which to end the curve.

Discussion

This function appends a quadratic curve to the current subpath. After adding the segment, the current point is set to the end point of the curve.

Availability

Available in iOS 2.0 and later.

See Also[CGContextAddCurveToPoint](#) (page 61)[CGContextAddArcToPoint](#) (page 60)**Related Sample Code**

QuartzDemo

Declared In

CGContext.h

CGContextAddRect

Adds a rectangular path to the current path.

```
void CGContextAddRect (
    CGContextRef c,
    CGRect rect
);
```

Parameters*c*

A graphics context.

rect

A rectangle, specified in user space coordinates.

Discussion

This is a convenience function that adds a rectangle to a path, using the following sequence of operations:

```
// start at origin
CGContextMoveToPoint (c, CGRectGetMinX(rect), CGRectGetMinY(rect));

// add bottom edge
CGContextAddLineToPoint (c, CGRectGetMaxX(rect), CGRectGetMinY(rect));

// add right edge
CGContextAddLineToPoint (c, CGRectGetMaxX(rect), CGRectGetMaxY(rect));

// add top edge
CGContextAddLineToPoint (c, CGRectGetMinX(rect), CGRectGetMaxY(rect));

// add left edge and close
CGContextClosePath (c);
```

Availability

Available in iOS 2.0 and later.

See Also

[CGContextAddRects](#) (page 66)

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextAddRects

Adds a set rectangular paths to the current path.

```
void CGContextAddRects (
    CGContextRef c,
    const CGRect rects[],
    size_t count
);
```

Parameters

c

A graphics context.

rects

An array of rectangles, specified in user space coordinates.

count

The number of rectangles in the *rects* array.

Discussion

This is a convenience function that adds an array of rectangles to a path, using the following operation:

```
for (k = 0; k < count; k++) {
    CGContextAddRect (c, m, rects[k]);
}
```

Availability

Available in iOS 2.0 and later.

See Also

[CGContextAddRect](#) (page 65)

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextBeginPage

Starts a new page in a page-based graphics context.

```
void CGContextBeginPage (
    CGContextRef c,
    const CGRect *mediaBox
);
```

Parameters

c

A page-based graphics context such as a PDF context. If you specify a context that does not support multiple pages, this function does nothing.

mediaBox

A Quartz rectangle defining the bounds of the new page, expressed in units of the default user space, or `NULL`. These bounds supersede any supplied for the media box when you created the context. If you pass `NULL`, Quartz uses the rectangle you supplied for the media box when the graphics context was created.

Discussion

When using a graphics context that supports multiple pages, you should call this function together with [CGContextEndPage](#) (page 83) to delineate the page boundaries in the output. In other words, each page should be bracketed by calls to `CGContextBeginPage` and `CGContextEndPage`. Quartz ignores all drawing operations performed outside a page boundary in a page-based context.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextBeginPath

Creates a new empty path in a graphics context.

```
void CGContextBeginPath (
    CGContextRef c
);
```

Parameters*c*

A graphics context.

Discussion

A graphics context can have only a single path in use at any time. If the specified context already contains a current path when you call this function, Quartz discards the old path and any data associated with it.

The current path is not part of the graphics state. Consequently, saving and restoring the graphics state has no effect on the current path.

Availability

Available in iOS 2.0 and later.

See Also[CGContextClosePath](#) (page 73)**Related Sample Code**

HeadsUpUI

Teslameter

Declared In

CGContext.h

CGContextBeginTransparencyLayer

Begins a transparency layer.

```
void CGContextBeginTransparencyLayer (
    CGContextRef context,
    CFDictionaryRef auxiliaryInfo
);
```

Parameters*context*

A graphics context.

auxiliaryInfo

A dictionary that specifies any additional information, or NULL.

Discussion

Until a corresponding call to [CGContextEndTransparencyLayer](#) (page 83), all subsequent drawing operations in the specified context are composited into a fully transparent backdrop (which is treated as a separate destination buffer from the context).

After a call to [CGContextEndTransparencyLayer](#), the result is composited into the context using the global alpha and shadow state of the context. This operation respects the clipping region of the context.

After a call to this function, all of the parameters in the graphics state remain unchanged with the exception of the following:

- The global alpha is set to 1.

- The shadow is turned off.

Ending the transparency layer restores these parameters to their previous values. Quartz maintains a transparency layer stack for each context, and transparency layers may be nested.

Tip: For best performance, make sure that you set the smallest possible clipping area for the objects in the transparency layer prior to calling `CGContextBeginTransparencyLayer`.

Availability

Available in iOS 2.0 and later.

Declared In

`CGContext.h`

CGContextBeginTransparencyLayerWithRect

Begins a transparency layer whose contents are bounded by the specified rectangle.

```
void CGContextBeginTransparencyLayerWithRect(  
    CGContextRef context,  
    CGRect rect,  
    CFDictionaryRef auxiliaryInfo  
);
```

Parameters

context

A graphics context.

rect

The rectangle, specified in user space, that bounds the transparency layer.

auxiliaryInfo

A dictionary that specifies any additional information, or `NULL`.

Discussion

This function is identical to [CGContextBeginTransparencyLayer](#) (page 68) except that the content of the transparency layer is within the bounds of the provided rectangle.

Availability

Available in iOS 2.0 and later.

Declared In

`CGContext.h`

CGContextClearRect

Paints a transparent rectangle.

```
void CGContextClearRect (
    CGContextRef c,
    CGRect rect
);
```

Parameters

c
The graphics context in which to paint the rectangle.

rect
The rectangle, in user space coordinates.

Discussion

If the provided context is a window or bitmap context, Quartz effectively clears the rectangle. For other context types, Quartz fills the rectangle in a device-dependent manner. However, you should not use this function in contexts other than window or bitmap contexts.

Availability

Available in iOS 2.0 and later.

Related Sample Code

GLImageProcessing

Declared In

CGContext.h

CGContextClip

Modifies the current clipping path, using the nonzero winding number rule.

```
void CGContextClip (
    CGContextRef c
);
```

Parameters

c
A graphics context that contains a path. If the context does not have a current path, the function does nothing.

Discussion

The function uses the nonzero winding number rule to calculate the intersection of the current path with the current clipping path. Quartz then uses the path resulting from the intersection as the new current clipping path for subsequent painting operations.

If the current path includes any open subpaths, Quartz treats them as if they were closed by calling [CGContextClosePath](#) (page 73).

Unlike the current path, the current clipping path is part of the graphics state. Therefore, to re-enlarge the paintable area by restoring the clipping path to a prior state, you must save the graphics state before you clip and restore the graphics state after you've completed any clipped drawing.

After determining the new clipping path, the function resets the context's current path to an empty path.

Availability

Available in iOS 2.0 and later.

See Also[CGContextEOClip](#) (page 84)**Related Sample Code**[GLImageProcessing](#)[QuartzDemo](#)**Declared In**[CGContext.h](#)**CGContextClipToMask**

Maps a mask into the specified rectangle and intersects it with the current clipping area of the graphics context.

```
void CGContextClipToMask (
    CGContextRef c,
    CGRect rect,
    CGImageRef mask
);
```

Parameters*c*

A graphics context.

rect

The rectangle to map the *mask* parameter to.

mask

An image or an image mask. If *mask* is an image, then it must be in the DeviceGray color space, may not have an alpha component, and may not be masked by an image mask or masking color.

Discussion

If the *mask* parameter is an image mask, then Quartz clips in a manner identical to the behavior seen with the function `CGContextDrawImage`—the mask indicates an area to be left unchanged when drawing. The source samples of the image mask determine which points of the clipping area are changed, acting as an "inverse alpha" value. If the value of a source sample in the image mask is *S*, then the corresponding point in the current clipping area is multiplied by an alpha value of $(1-S)$. For example, if *S* is 1 then the point in the clipping area becomes transparent. If *S* is 0, the point in the clipping area is unchanged.

If the *mask* parameter is an image, then *mask* acts like an alpha mask and is blended with the current clipping area. The source samples of mask determine which points of the clipping area are changed. If the value of the source sample in mask is *S*, then the corresponding point in the current clipping area is multiplied by an alpha of *S*. For example, if *S* is 0, then the point in the clipping area becomes transparent. If *S* is 1, the point in the clipping area is unchanged.

Availability

Available in iOS 2.0 and later.

Related Sample Code[QuartzDemo](#)[Reflection](#)**Declared In**[CGContext.h](#)

CGContextClipToRect

Sets the clipping path to the intersection of the current clipping path with the area defined by the specified rectangle.

```
void CGContextClipToRect (
    CGContextRef c,
    CGRect rect
);
```

Parameters

c

The graphics context for which to set the clipping path.

rect

A `CGRect` value that specifies, in the user space, the location and dimensions of the rectangle to be used in determining the new clipping path.

Discussion

This function sets the specified graphics context's clipping region to the area which intersects both the current clipping path and the specified rectangle.

After determining the new clipping path, the `CGContextClipToRect` function resets the context's current path to an empty path.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextClipToRects](#) (page 72)

Related Sample Code

GLImageProcessing

QuartzDemo

Declared In

`CGContext.h`

CGContextClipToRects

Sets the clipping path to the intersection of the current clipping path with the region defined by an array of rectangles.

```
void CGContextClipToRects (
    CGContextRef c,
    const CGRect rects[],
    size_t count
);
```

Parameters

c

The graphics context for which to set the clipping path.

rects

An array of rectangles. The locations and dimensions of the rectangles are specified in the user space coordinate system.

count

The total number of array entries in the `rects` parameter.

Discussion

This function sets the clipping path to the intersection of the current clipping path and the region within the specified rectangles.

After determining the new clipping path, the function resets the context's current path to an empty path.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextClipToRect](#) (page 72)

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextClosePath

Closes and terminates the current path's subpath.

```
void CGContextClosePath (
    CGContextRef c
);
```

Parameters

c

A graphics context.

Discussion

Appends a line from the current point to the starting point of the current subpath and ends the subpath.

After closing the subpath, your application can begin a new subpath without first calling [CGContextMoveToPoint](#) (page 92). In this case, a new subpath is implicitly created with a starting and current point equal to the previous subpath's starting point.

If the current path is empty or the current subpath is already closed, this function does nothing.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextAddPath](#) (page 64)

Related Sample Code

GLImageProcessing

HeadsUpUI

QuartzDemo

Declared In

CGContext.h

CGContextConcatCTM

Transforms the user coordinate system in a context using a specified matrix.

```
void CGContextConcatCTM (  
    CGContextRef c,  
    CGAffineTransform transform  
);
```

Parameters

c

A graphics context.

transform

The transformation matrix to apply to the specified context's current transformation matrix.

Discussion

When you call the function `CGContextConcatCTM`, it concatenates (that is, it combines) two matrices, by multiplying them together. The order in which matrices are concatenated is important, as the operations are not commutative. When you call `CGContextConcatCTM`, the resulting CTM in the context is: $CTM_{new} = transform * CTM_{context}$.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

`CGContext.h`

CGContextConvertPointToDeviceSpace

Returns a point that is transformed from user space coordinates to device space coordinates.

```
CGPoint CGContextConvertPointToDeviceSpace (  
    CGContextRef c,  
    CGPoint point  
);
```

Parameters

c

A graphics context.

point

The point, in user space coordinates, to transform.

Return Value

The coordinates of the point in device space coordinates.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextConvertPointToUserSpace](#) (page 75)

Declared In

CGContext.h

CGContextConvertPointToUserSpace

Returns a point that is transformed from device space coordinates to user space coordinates.

```
CGPoint CGContextConvertPointToUserSpace (
    CGContextRef c,
    CGPoint point
);
```

Parameters*c*

A graphics context.

point

The point, in device space coordinates, to transform.

Return Value

The coordinates of the point in user space coordinates.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextConvertPointToDeviceSpace](#) (page 74)

Declared In

CGContext.h

CGContextConvertRectToDeviceSpace

Returns a rectangle that is transformed from user space coordinate to device space coordinates.

```
CGRect CGContextConvertRectToDeviceSpace (
    CGContextRef c,
    CGRect rect
);
```

Parameters*c*

A graphics context.

rect

The rectangle, in user space coordinates, to transform.

Return Value

The rectangle in device space coordinates.

Discussion

In general affine transforms do not preserve rectangles. As a result, this function returns the smallest rectangle that contains the transformed corner points of the rectangle.

Availability

Available in iOS 2.0 and later.

See Also[CGContextConvertRectToUserSpace](#) (page 76)**Declared In**

CGContext.h

CGContextConvertRectToUserSpace

Returns a rectangle that is transformed from device space coordinate to user space coordinates.

```
CGRect CGContextConvertRectToUserSpace (
    CGContextRef c,
    CGRect rect
);
```

Parameters*c*

A graphics context.

rect

The rectangle, in device space coordinates, to transform.

Return Value

The rectangle in user space coordinates.

Discussion

In general, affine transforms do not preserve rectangles. As a result, this function returns the smallest rectangle that contains the transformed corner points of the rectangle.

Availability

Available in iOS 2.0 and later.

See Also[CGContextConvertRectToDeviceSpace](#) (page 75)**Declared In**

CGContext.h

CGContextConvertSizeToDeviceSpace

Returns a size that is transformed from user space coordinates to device space coordinates.

```
CGSize CGContextConvertSizeToDeviceSpace (
    CGContextRef c,
    CGSize size
);
```

Parameters*c*

A graphics context.

size

The size, in user space coordinates, to transform.

Return Value

The size in device space coordinates.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextConvertSizeToUserSpace](#) (page 77)

Declared In

CGContext.h

CGContextConvertSizeToUserSpace

Returns a size that is transformed from device space coordinates to user space coordinates

```
CGSize CGContextConvertSizeToUserSpace (
    CGContextRef c,
    CGSize size
);
```

Parameters

c

A graphics context.

size

The size, in device space coordinates, to transform.

Return Value

The size in user space coordinates.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextConvertSizeToDeviceSpace](#) (page 76)

Declared In

CGContext.h

CGContextCopyPath

Returns a Quartz path object built from the current path information in a graphics context.

```
CGPathRef CGContextCopyPath(
    CGContextRef context
);
```

Parameters

context

A graphics context whose current path is not empty.

Return Value

A Quartz path object containing the current path data.

Availability

Available in iOS 4.0 and later.

Declared In

CGContext.h

CGContextDrawImage

Draws an image into a graphics context.

```
void CGContextDrawImage (
    CGContextRef c,
    CGRect rect,
    CGImageRef image
);
```

Parameters*c*

The graphics context in which to draw the image.

rect

The location and dimensions in user space of the bounding box in which to draw the image.

image

The image to draw.

Discussion

Quartz scales the image—disproportionately, if necessary—to fit the bounds specified by the `rect` parameter.

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch

GLPaint

PVRTextureLoader

QuartzDemo

Reflection

Declared In

CGContext.h

CGContextDrawLinearGradient

Paints a gradient fill that varies along the line defined by the provided starting and ending points.

```
void CGContextDrawLinearGradient(
    CGContextRef context,
    CGGradientRef gradient,
    CGPoint startPoint,
    CGPoint endPoint,
    CGGradientDrawingOptions options
);
```

Parameters*context*

A Quartz graphics context.

gradient

A `CGGradient` object.

startPoint

The coordinate that defines the starting point of the gradient.

endPoint

The coordinate that defines the ending point of the gradient.

options

Option flags (`kCGGradientDrawsBeforeStartLocation` (page 200) or `kCGGradientDrawsAfterEndLocation` (page 200)) that control whether the fill is extended beyond the starting or ending point.

Discussion

The color at location 0 in the `CGGradient` object is mapped to the starting point. The color at location 1 in the `CGGradient` object is mapped to the ending point. Colors are linearly interpolated between these two points based on the location values of the gradient. The option flags control whether the gradient is drawn before the start point or after the end point.

Availability

Available in iOS 2.0 and later.

Related Sample Code

GLImageProcessing

PVRTextureLoader

QuartzDemo

Reflection

TheElements

Declared In

`CGContext.h`

CGContextDrawPath

Draws the current path using the provided drawing mode.

```
void CGContextDrawPath (
    CGContextRef c,
    CGPathDrawingMode mode
);
```

Parameters

c

A graphics context that contains a path to paint.

mode

A path drawing mode constant—`kCGPathFill` (page 249), `kCGPathEOFill` (page 249), `kCGPathStroke` (page 249), `kCGPathFillStroke` (page 250), or `kCGPathEOFillStroke` (page 250). For a discussion of these constants, see *CGPath Reference*.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextFillPath](#) (page 85)

[CGContextEOFillPath](#) (page 84)[CGContextStrokePath](#) (page 129)**Related Sample Code**

QuartzDemo

WeatherMap

Declared In

CGContext.h

CGContextDrawPDFPage

Draws a page in the current user space of a PDF context.

```
void CGContextDrawPDFPage (
    CGContextRef c,
    CGPDFPageRef page
);
```

Parameters*c*

The graphics context in which to draw the PDF page.

page

A Quartz PDF page.

Discussion

This function works in conjunction with the opaque type `CGPDFPageRef` to draw individual pages into a PDF context.

For applications running in Mac OS X version 10.3 and later, this function is recommended as a replacement for the older function `CGContextDrawPDFDocument`.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

ZoomingPDFViewer

Declared In

CGContext.h

CGContextDrawRadialGradient

Paints a gradient fill that varies along the area defined by the provided starting and ending circles.


```
void CGContextDrawRadialGradient(
    CGContextRef context,
    CGGradientRef gradient,
    CGPoint startCenter,
    CGFloat startRadius,
    CGPoint endCenter,
    CGFloat endRadius,
    CGGradientDrawingOptions options
);
```

Parameters*context*

A Quartz graphics context.

gradient

A CGGradient object.

startCenter

The coordinate that defines the center of the starting circle.

startRadius

The radius of the starting circle.

endCenter

The coordinate that defines the center of the ending circle.

endRadius

The radius of the ending circle.

options

Option flags ([kCGGradientDrawsBeforeStartLocation](#) (page 200) or [kCGGradientDrawsAfterEndLocation](#) (page 200)) that control whether the gradient is drawn before the starting circle or after the ending circle.

Discussion

The color at location 0 in the CGGradient object is mapped to the circle defined by *startCenter* and *startRadius*. The color at location 1 in the CGGradient object is mapped to the circle defined by *endCenter* and *endRadius*. Colors are linearly interpolated between the starting and ending circles based on the location values of the gradient. The option flags control whether the gradient is drawn before the start point or after the end point.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextDrawShading

Fills the clipping path of a context with the specified shading.

```
void CGContextDrawShading (
    CGContextRef c,
    CGShadingRef shading
);
```

Parameters*c*

The graphics context in which to draw the shading.

shading

A Quartz shading. Quartz retains this object; upon return, you may safely release it.

Discussion

In Mac OS X v10.5 and later, the preferred way to draw gradients is to use a `CGGradient` object. See *CGGradient Reference*.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextDrawLinearGradient](#) (page 78)

[CGContextDrawRadialGradient](#) (page 80)

Declared In

`CGContext.h`

CGContextDrawTiledImage

Repeatedly draws an image, scaled to the provided rectangle, to fill the current clip region.

```
void CGContextDrawTiledImage(
    CGContextRef context,
    CGRect rect,
    CGImageRef image
);
```

Parameters*context*

The graphics context in which to draw the image.

rect

A rectangle that specifies the origin and size of the destination tile. Quartz scales the image—disproportionately, if necessary—to fit the bounds specified by the `rect` parameter.

image

The image to draw.

Discussion

Quartz draws the scaled image starting at the origin of the rectangle in user space, then moves to a new point (horizontally by the width of the tile and/or vertically by the height of the tile), draws the scaled image, moves again, draws again, and so on, until the current clip region is tiled with copies of the image. Unlike patterns, the image is tiled in user space, so transformations applied to the CTM affect the final result.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextEndPage

Ends the current page in a page-based graphics context.

```
void CGContextEndPage (
    CGContextRef c
);
```

Parameters

c
A page-based graphics context.

Discussion

When using a graphics context that supports multiple pages, you should call this function to terminate drawing in the current page.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextBeginPage](#) (page 67)

Declared In

CGContext.h

CGContextEndTransparencyLayer

Ends a transparency layer.

```
void CGContextEndTransparencyLayer (
    CGContextRef context
);
```

Parameters

context
A graphics context.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextBeginTransparencyLayer](#) (page 68)

Declared In

CGContext.h

CGContextEOClip

Modifies the current clipping path, using the even-odd rule.

```
void CGContextEOClip (  
    CGContextRef c  
);
```

Parameters

c

A graphics context containing a path. If the context does not have a current path, the function does nothing.

Discussion

The function uses the even-odd rule to calculate the intersection of the current path with the current clipping path. Quartz then uses the path resulting from the intersection as the new current clipping path for subsequent painting operations.

If the current path includes any open subpaths, Quartz treats them as if they were closed by calling [CGContextClosePath](#) (page 73).

Unlike the current path, the current clipping path is part of the graphics state. Therefore, to re-enlarge the paintable area by restoring the clipping path to a prior state, you must save the graphics state before you clip and restore the graphics state after you've completed any clipped drawing.

After determining the new clipping path, the function resets the context's current path to an empty path.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextClip](#) (page 70)

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextEOFillPath

Paints the area within the current path, using the even-odd fill rule.

```
void CGContextEOFillPath (  
    CGContextRef c  
);
```

Parameters

c

A graphics context that contains a path to fill.

Discussion

Quartz treats each subpath as if it were closed by calling [CGContextClosePath](#) (page 73). The even-odd rule is described in "Filling a Path" in *Quartz 2D Programming Guide*.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextFillPath](#) (page 85)

[CGContextStrokePath](#) (page 129)

[CGContextDrawPath](#) (page 79)

Declared In

CGContext.h

CGContextFillEllipseInRect

Paints the area of the ellipse that fits inside the provided rectangle, using the fill color in the current graphics state.

```
void CGContextFillEllipseInRect (
    CGContextRef context,
    CGRect rect
);
```

Parameters

context

A graphics context.

rect

A rectangle that defines the area for the ellipse to fit in.

Availability

Available in iOS 2.0 and later.

Related Sample Code

GKRocket

QuartzDemo

Declared In

CGContext.h

CGContextFillPath

Paints the area within the current path, using the nonzero winding number rule.

```
void CGContextFillPath (
    CGContextRef c
);
```

Parameters

c

A graphics context that contains a path to fill.

Discussion

Quartz treats each subpath as if it were closed by calling [CGContextClosePath](#) (page 73). The nonzero winding number rule is described in “Filling a Path” in *Quartz 2D Programming Guide*.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextEOFillPath](#) (page 84)

[CGContextStrokePath](#) (page 129)

[CGContextDrawPath](#) (page 79)

Related Sample Code

aurioTouch

GKRocket

GLImageProcessing

HeadsUpUI

QuartzDemo

Declared In

CGContext.h

CGContextFillRect

Paints the area contained within the provided rectangle, using the fill color in the current graphics state.

```
void CGContextFillRect (
    CGContextRef c,
    CGRect rect
);
```

Parameters

c

A graphics context.

rect

A rectangle, in user space coordinates.

Discussion

As a side effect when you call this function, Quartz clears the current path.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextFillRects](#) (page 87)

Related Sample Code

AccelerometerGraph

avTouch

QuartzDemo

SpeakHere

ZoomingPDFViewer

Declared In

CGContext.h

CGContextFillRects

Paints the areas contained within the provided rectangles, using the fill color in the current graphics state.

```
void CGContextFillRects (
    CGContextRef c,
    const CGRect rects[],
    size_t count
);
```

Parameters

c

A graphics context .

rects

An array of rectangles, in user space coordinates.

count

The number rectangles in the *rects* array.

Discussion

As a side effect when you call this function, Quartz clears the current path.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextFillRect](#) (page 86)

Declared In

CGContext.h

CGContextFlush

Forces all pending drawing operations in a window context to be rendered immediately to the destination device.

```
void CGContextFlush (
    CGContextRef c
);
```

Parameters

c

The window context to flush. If you pass a PDF context or a bitmap context, this function does nothing.

Discussion

When you call this function, Quartz immediately flushes the current drawing to the destination device (for example, a screen). Because the system software flushes a context automatically at the appropriate times, calling this function could have an adverse effect on performance. Under normal conditions, you do not need to call this function.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextGetClipBoundingBox

Returns the bounding box of a clipping path.

```
CGRect CGContextGetClipBoundingBox (
    CGContextRef c
);
```

Parameters

c

The graphics context to modify.

Return Value

The bounding box of the clipping path, specified in user space.

Discussion

The bounding box is the smallest rectangle completely enclosing all points in the clipping path, including control points for any Bezier curves in the path.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextGetCTM

Returns the current transformation matrix.

```
CGAffineTransform CGContextGetCTM (
    CGContextRef c
);
```

Parameters

c

A graphics context.

Return Value

The transformation matrix for the current graphics state of the specified context.

Availability

Available in iOS 2.0 and later.

Related Sample Code

PhotoScroller

Declared In

CGContext.h

CGContextGetInterpolationQuality

Returns the current level of interpolation quality for a graphics context.


```
CGInterpolationQuality CGContextGetInterpolationQuality (
    CGContextRef c
);
```

Parameters

c
The graphics context to examine.

Return Value

The current level of interpolation quality.

Discussion

Interpolation quality is a graphics state parameter that provides a hint for the level of quality to use for image interpolation (for example, when scaling the image). Not all contexts support all interpolation quality levels.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetInterpolationQuality](#) (page 110)

Declared In

CGContext.h

CGContextGetPathBoundingBox

Returns the smallest rectangle that contains the current path.

```
CRect CGContextGetPathBoundingBox (
    CGContextRef c
);
```

Parameters

c
The graphics context, containing a path, to examine.

Return Value

A `CRect` value that specifies the dimensions and location, in user space, of the bounding box of the path. If there is no path, the function returns `CRectNull`.

Discussion

The bounding box is the smallest rectangle completely enclosing all points in a path, including control points for Bézier cubic and quadratic curves.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextGetPathCurrentPoint

Returns the current point in a non-empty path.

```
CGPoint CGContextGetPathCurrentPoint (
    CGContextRef c
);
```

Parameters

c

The graphics context containing the path to examine.

Return Value

A `CGPoint` value that specifies the location, in user space, of current point in the context's path. If there is no path, the function returns `CGPointZero`.

Availability

Available in iOS 2.0 and later.

Declared In

`CGContext.h`

CGContextGetTextMatrix

Returns the current text matrix.

```
CGAffineTransform CGContextGetTextMatrix (
    CGContextRef c
);
```

Parameters

c

The graphics context for which to obtain the text matrix.

Return Value

The current text matrix.

Availability

Available in iOS 2.0 and later.

Declared In

`CGContext.h`

CGContextGetTextPosition

Returns the location at which text is drawn.

```
CGPoint CGContextGetTextPosition (
    CGContextRef c
);
```

Parameters

c

The graphics context from which to obtain the current text position.

Return Value

Returns a `CGPoint` value that specifies the x and y values at which text is to be drawn, in user space coordinates.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextGetTypeID

Returns the type identifier for Quartz graphics contexts.

```
CTypeID CGContextGetTypeID (
    void
);
```

Return Value

The identifier for the opaque type [CGContextRef](#) (page 132).

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextGetUserSpaceToDeviceSpaceTransform

Returns an affine transform that maps user space coordinates to device space coordinates.

```
CGAffineTransform CGContextGetUserSpaceToDeviceSpaceTransform (
    CGContextRef c
);
```

Parameters

c

A graphics context.

Return Value

The affine transforms that maps the user space of the graphics context to the device space.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextIsPathEmpty

Indicates whether the current path contains any subpaths.

```
bool CGContextIsPathEmpty (
    CGContextRef c
);
```

Parameters

c
The graphics context containing the path to examine.

Return Value

Returns 1 if the context's path contains no subpaths, otherwise returns 0.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextMoveToPoint

Begins a new subpath at the point you specify.

```
void CGContextMoveToPoint (
    CGContextRef c,
    CGFloat x,
    CGFloat y
);
```

Parameters

c
A graphics context.

x
The x-value, in user space coordinates, for the point.

y
The y-value, in user space coordinates, for the point.

Discussion

This point you specifies becomes the start point of a new subpath. The current point is set to this start point.

Availability

Available in iOS 2.0 and later.

Related Sample Code

GKRocket
GLImageProcessing
HeadsUpUI
QuartzDemo
Teslameter

Declared In

CGContext.h

CGContextPathContainsPoint

Checks to see whether the specified point is contained in the current path.

```
bool CGContextPathContainsPoint (
    CGContextRef context,
    CGPoint point,
    CGPathDrawingMode mode
);
```

Parameters

context

A graphics context.

point

The point to check, specified in user space units.

mode

A path drawing mode—`kCGPathFill`, `kCGPathEOFill`, `kCGPathStroke`, `kCGPathFillStroke`, or `kCGPathEOFillStroke`. See [CGPathDrawingMode](#) for more information on these modes.

Return Value

Returns `true` if *point* is inside the current path of the graphics context; `false` otherwise.

Discussion

A point is contained within the path of a graphics context if the point is inside the painted region when the path is stroked or filled with opaque colors using the specified path drawing mode. A point can be inside a path only if the path is explicitly closed by calling the function [CGContextClosePath](#) (page 73), for paths drawn directly to the current context, or [CGPathCloseSubpath](#) (page 240), for paths first created as `CGPath` objects and then drawn to the current context.

Availability

Available in iOS 2.0 and later.

Declared In

`CGContext.h`

CGContextRelease

Decrements the retain count of a graphics context.

```
void CGContextRelease (
    CGContextRef c
);
```

Parameters

c

The graphics context to release.

Discussion

This function is equivalent to `CFRelease`, except it does not cause an error if *c* is `NULL`.

Availability

Available in iOS 2.0 and later.

Related Sample Code

[aurioTouch](#)

GLImageProcessing
PVRTextureLoader
Reflection
TheElements

Declared In
CGContext.h

CGContextReplacePathWithStrokedPath

Replaces the path in the graphics context with the stroked version of the path.

```
void CGContextReplacePathWithStrokedPath (  
    CGContextRef c  
);
```

Parameters

c
A graphics context.

Discussion

Quartz creates a stroked path using the parameters of the current graphics context. The new path is created so that filling it draws the same pixels as stroking the original path. You can use this path in the same way you use the path of any context. For example, you can clip to the stroked version of a path by calling this function followed by a call to the function [CGContextClip](#) (page 70).

Availability

Available in iOS 2.0 and later.

Declared In
CGContext.h

CGContextRestoreGState

Sets the current graphics state to the state most recently saved.

```
void CGContextRestoreGState (  
    CGContextRef c  
);
```

Parameters

c
The graphics context whose state you want to modify.

Discussion

Quartz removes the graphics state that is at the top of the stack so that the most recently saved state becomes the current graphics state.

Availability

Available in iOS 2.0 and later.

See Also
[CGContextSaveGState](#) (page 96)

Related Sample Code

GLImageProcessing

PVRTextureLoader

QuartzDemo

Teslameter

ZoomingPDFViewer

Declared In

CGContext.h

CGContextRetain

Increments the retain count of a graphics context.

```
CGContextRef CGContextRetain (
    CGContextRef c
);
```

Parameters*c*

The graphics context to retain.

Return Value

The same graphics context you passed in as the `context` parameter.

Discussion

This function is equivalent to `CFRetain`, except it does not cause an error if *c* is `NULL`.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextRotateCTM

Rotates the user coordinate system in a context.

```
void CGContextRotateCTM (
    CGContextRef c,
    CGFloat angle
);
```

Parameters*c*

A graphics context.

angle

The angle, in radians, by which to rotate the coordinate space of the specified context. Positive values rotate counterclockwise and negative values rotate clockwise.)

Discussion

The direction that the context is rotated may appear to be altered by the state of the current transformation matrix prior to executing this function. For example, on iOS, a `UIView` applies a transformation to the graphics context that inverts the Y-axis (by multiplying it by -1). Rotating the user coordinate system on coordinate system that was previously flipped results in a rotation in the opposite direction (that is, positive values appear to rotate the coordinate system in the clockwise direction).

Availability

Available in iOS 2.0 and later.

Related Sample Code

`avTouch`

`SpeakHere`

Declared In

`CGContext.h`

CGContextSaveGState

Pushes a copy of the current graphics state onto the graphics state stack for the context.

```
void CGContextSaveGState (
    CGContextRef c
);
```

Parameters

c

The graphics context whose current graphics state you want to save.

Discussion

Each graphics context maintains a stack of graphics states. Note that not all aspects of the current drawing environment are elements of the graphics state. For example, the current path is not considered part of the graphics state and is therefore not saved when you call the `CGContextSaveGState` function. The graphics state parameters that *are* saved are:

- CTM (current transformation matrix)
- clip region
- image interpolation quality
- line width
- line join
- miter limit
- line cap
- line dash
- flatness
- should anti-alias
- rendering intent
- fill color space
- stroke color space

- fill color
- stroke color
- alpha value
- font
- font size
- character spacing
- text drawing mode
- shadow parameters
- the pattern phase
- the font smoothing parameter
- blend mode

To restore your drawing environment to a previously saved state, you can use [CGContextRestoreGState](#) (page 94).

Availability

Available in iOS 2.0 and later.

Related Sample Code

GLImageProcessing

PVRTextureLoader

QuartzDemo

Teslameter

ZoomingPDFViewer

Declared In

CGContext.h

CGContextScaleCTM

Changes the scale of the user coordinate system in a context.

```
void CGContextScaleCTM (
    CGContextRef c,
    CGFloat sx,
    CGFloat sy
);
```

Parameters

c

A graphics context.

sx

The factor by which to scale the x-axis of the coordinate space of the specified context.

sy

The factor by which to scale the y-axis of the coordinate space of the specified context.

Availability

Available in iOS 2.0 and later.

Related Sample Code

[aurioTouch](#)

[avTouch](#)

[QuartzDemo](#)

[SpeakHere](#)

[ZoomingPDFViewer](#)

Declared In

[CGContext.h](#)

CGContextSelectFont

Sets the font and font size in a graphics context.

```
void CGContextSelectFont (
    CGContextRef c,
    const char *name,
    CGFloat size,
    CGTextEncoding textEncoding
);
```

Parameters

c

The graphics context for which to set the font and font size.

name

A null-terminated string that contains the PostScript name of the font to set.

size

A value that specifies the font size to set, in text space units.

textEncoding

A [CGTextEncoding](#) value that specifies the encoding used for the font. For a description of the available values, see “[CGTextEncoding](#)” (page 141).

Availability

Available in iOS 2.0 and later.

See Also

[CGContextShowText](#) (page 127)

[CGContextShowTextAtPoint](#) (page 127)

Related Sample Code

[QuartzDemo](#)

Declared In

[CGContext.h](#)

CGContextSetAllowsAntialiasing

Sets whether or not to allow anti-aliasing for a graphics context.

```
void CGContextSetAllowsAntialiasing (
    CGContextRef context,
    bool allowsAntialiasing
);
```

Parameters

context

A graphics context.

allowsAntialiasing

A Boolean value that specifies whether or not to allow antialiasing. Pass `true` to allow antialiasing; `false` otherwise. This parameter is not part of the graphics state.

Discussion

Quartz performs antialiasing for a graphics context if both the `allowsAntialiasing` parameter and the graphics state parameter `shouldAntialias` are `true`.

This parameter is not part of the graphics state.

Availability

Available in iOS 2.0 and later.

Related Sample Code

Teslameter

Declared In

`CGContext.h`

CGContextSetAllowsFontSmoothing

Sets whether or not to allow font smoothing for a graphics context.

```
void CGContextSetAllowsFontSmoothing(
    CGContextRef context,
    bool allowsFontSmoothing
);
```

Parameters

context

A graphics context.

allowsFontSmoothing

A Boolean value that specifies whether font smoothing is allowed in the specified context.

Discussion

Font are smoothed if they are antialiased when drawn and if font smoothing is both allowed and enabled. For information on how to enable font smoothing, see the [CGContextSetShouldSmoothFonts](#) (page 118) function.

This parameter is not part of the graphics state.

Availability

Available in iOS 4.0 and later.

Declared In

`CGContext.h`

CGContextSetAllowsFontSubpixelPositioning

Sets whether or not to allow subpixel positioning for a graphics context

```
void CGContextSetAllowsFontSubpixelPositioning(  
    CGContextRef context,  
    bool allowsFontSubpixelPositioning  
);
```

Parameters

context

A graphics context.

allowsFontSubpixelPositioning

A Boolean value that specifies whether subpixel positioning of glyphs is allowed in the specified context.

Discussion

Subpixel positioning is used by the graphics context if it is allowed, enabled, and if the font itself is antialiased when drawn. For information on how to enable subpixel positioning, see the [CGContextSetShouldSubpixelPositionFonts](#) (page 119) function.

This parameter is not part of the graphics state.

Availability

Available in iOS 4.0 and later.

Declared In

CGContext.h

CGContextSetAllowsFontSubpixelQuantization

Sets whether or not to allow subpixel quantization for a graphics context

```
void CGContextSetAllowsFontSubpixelQuantization(  
    CGContextRef context,  
    bool allowsFontSubpixelQuantization  
);
```

Parameters

context

A graphics context.

allowsFontSubpixelQuantization

A Boolean value that specifies whether subpixel quantization of glyphs is allowed in the specified context.

Discussion

Subpixel quantization is used by the graphics context if it is allowed, enabled, and if glyphs would be drawn at subpixel positions. For information on how to enable subpixel quantization, see the [CGContextSetShouldSubpixelQuantizeFonts](#) (page 119) function.

This parameter is not part of the graphics state.

Availability

Available in iOS 4.0 and later.

Declared In

CGContext.h

CGContextSetAlpha

Sets the opacity level for objects drawn in a graphics context.

```
void CGContextSetAlpha (
    CGContextRef c,
    CGFloat alpha
);
```

Parameters*c*

The graphics context for which to set the current graphics state's alpha value parameter.

alpha

A value that specifies the opacity level. Values can range from 0.0 (transparent) to 1.0 (opaque). Values outside this range are clipped to 0.0 or 1.0.

Discussion

This function sets the alpha value parameter for the specified graphics context. To clear the contents of the drawing canvas, use [CGContextClearRect](#) (page 69).

Availability

Available in iOS 2.0 and later.

Related Sample Code

HazardMap

Declared In

CGContext.h

CGContextSetBlendMode

Sets how Quartz composites sample values for a graphics context.

```
void CGContextSetBlendMode (
    CGContextRef context,
    CGBlendMode mode
);
```

Parameters*context*

The graphics context to modify.

mode

A blend mode. See “[CGBlendMode](#)” (page 133) for a list of the constants you can supply.

Availability

Available in iOS 2.0 and later.

Related Sample Code

PVRTextureLoader

QuartzDemo

Declared In

CGContext.h

CGContextSetCharacterSpacing

Sets the current character spacing.

```
void CGContextSetCharacterSpacing (
    CGContextRef c,
    CGFloat spacing
);
```

Parameters*c*

The graphics context for which to set the character spacing.

spacing

A value that represents the amount of additional space to place between glyphs, in text space coordinates.

Discussion

Quartz adds the additional space to the advance between the origin of one character and the origin of the next character. For information about the text coordinate system, see [CGContextSetTextMatrix](#) (page 123).

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextSetCMYKFillColor

Sets the current fill color to a value in the DeviceCMYK color space.

```
void CGContextSetCMYKFillColor (
    CGContextRef c,
    CGFloat cyan,
    CGFloat magenta,
    CGFloat yellow,
    CGFloat black,
    CGFloat alpha
);
```

Parameters*c*

The graphics context for which to set the current fill color.

cyan

The cyan intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

magenta

The magenta intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

yellow

The yellow intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

black

The black intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

alpha

A value that specifies the opacity level. Values can range from 0.0 (transparent) to 1.0 (opaque). Values outside this range are clipped to 0.0 or 1.0.

Discussion

Quartz provides convenience functions for each of the device color spaces that allow you to set the fill or stroke color space and the fill or stroke color with one function call.

When you call this function, two things happen:

- Quartz sets the current fill color space to DeviceCMYK.
- Quartz sets the current fill color to the value specified by the `cyan`, `magenta`, `yellow`, `black`, and `alpha` parameters.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetCMYKStrokeColor](#) (page 103)

Declared In

`CGContext.h`

CGContextSetCMYKStrokeColor

Sets the current stroke color to a value in the DeviceCMYK color space.

```
void CGContextSetCMYKStrokeColor (
    CGContextRef c,
    CGFloat cyan,
    CGFloat magenta,
    CGFloat yellow,
    CGFloat black,
    CGFloat alpha
);
```

Parameters

c

The graphics context for which to set the current stroke color.

cyan

The cyan intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

magenta

The magenta intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

yellow

The yellow intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

black

The black intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

alpha

A value that specifies the opacity level. Values can range from 0.0 (transparent) to 1.0 (opaque). Values outside this range are clipped to 0.0 or 1.0.

Discussion

When you call this function, two things happen:

- Quartz sets the current stroke color space to DeviceCMYK.
- Quartz sets the current stroke color to the value specified by the *cyan*, *magenta*, *yellow*, *black*, and *alpha* parameters.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetCMYKFillColor](#) (page 102)

Declared In

CGContext.h

CGContextSetFillColor

Sets the current fill color.

```
void CGContextSetFillColor (
    CGContextRef c,
    const CGFloat components[]
);
```

Parameters

c

The graphics context for which to set the current fill color.

components

An array of intensity values describing the color to set. The number of array elements must equal the number of components in the current fill color space, plus an additional component for the alpha value.

Discussion

The current fill color space must not be a pattern color space. For information on setting the fill color when using a pattern color space, see [CGContextSetFillPattern](#) (page 106). Note that the preferred API to use is now [CGContextSetFillColorWithColor](#) (page 106).

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch

Declared In

CGContext.h

CGContextSetFillColorSpace

Sets the fill color space in a graphics context.

```
void CGContextSetFillColorSpace (
    CGContextRef c,
    CGColorSpaceRef colorspace
);
```

Parameters

c

The graphics context for which to set the fill color space.

colorspace

The new fill color space. Quartz retains this object; upon return, you may safely release it.

Discussion

As a side effect of this function, Quartz assigns an appropriate initial value to the fill color, based on the specified color space. To change this value, call [CGContextSetFillColor](#) (page 104). Note that the preferred API to use is now [CGContextSetFillColorWithColor](#) (page 106).

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch

avTouch

QuartzDemo

SpeakHere

Declared In

CGContext.h

CGContextSetFillColorWithColor

Sets the current fill color in a graphics context, using a Quartz color.

```
void CGContextSetFillColorWithColor (
    CGContextRef c,
    CGColorRef color
);
```

Parameters

c
The graphics context for which to set the fill color.

color
The new fill color.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetFillColor](#) (page 104)

Related Sample Code

AccelerometerGraph

avTouch

QuartzDemo

SpeakHere

WeatherMap

Declared In

CGContext.h

CGContextSetFillPattern

Sets the fill pattern in the specified graphics context.

```
void CGContextSetFillPattern (
    CGContextRef c,
    CGPatternRef pattern,
    const CGFloat components[]
);
```

Parameters

c
The graphics context to modify.

pattern
A fill pattern. Quartz retains this object; upon return, you may safely release it.

components
If the pattern is an uncolored (or a masking) pattern, pass an array of intensity values that specify the color to use when the pattern is painted. The number of array elements must equal the number of components in the base space of the fill pattern color space, plus an additional component for the alpha value.

If the pattern is a colored pattern, pass an alpha value.

Discussion

The current fill color space must be a pattern color space. Otherwise, the result of calling this function is undefined. If you want to set a fill color, not a pattern, use [CGContextSetFillColorWithColor](#) (page 106).

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextSetFlatness

Sets the accuracy of curved paths in a graphics context.

```
void CGContextSetFlatness (
    CGContextRef c,
    CGFloat flatness
);
```

Parameters

c

The graphics context to modify.

flatness

The largest permissible distance, measured in device pixels, between a point on the true curve and a point on the approximated curve.

Discussion

This function controls how accurately curved paths are rendered. Setting the flatness value to less than 1.0 renders highly accurate curves, but lengthens rendering times.

In most cases, you should not change the flatness value. Customizing the flatness value for the capabilities of a particular output device impairs the ability of your application to render to other devices.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextSetFont

Sets the platform font in a graphics context.

```
void CGContextSetFont (
    CGContextRef c,
    CGFontRef font
);
```

Parameters

c
The graphics context for which to set the font.

font
A Quartz font.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextSetFontSize

Sets the current font size.

```
void CGContextSetFontSize (
    CGContextRef c,
    CGFloat size
);
```

Parameters

c
A graphics context.

size
A font size, expressed in text space units.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextSetGrayFillColor

Sets the current fill color to a value in the DeviceGray color space.

```
void CGContextSetGrayFillColor (
    CGContextRef c,
    CGFloat gray,
    CGFloat alpha
);
```

Parameters*c*

The graphics context for which to set the current fill color.

gray

A value that specifies the desired gray level. The DeviceGray color space permits the specification of a value ranging from 0.0 (absolute black) to 1.0 (absolute white). Values outside this range are clamped to 0.0 or 1.0.

alpha

A value that specifies the opacity level. Values can range from 0.0 (transparent) to 1.0 (opaque). Values outside this range are clipped to 0.0 or 1.0.

Discussion

When you call this function, two things happen:

- Quartz sets the current fill color space to DeviceGray.
- Quartz sets the current fill color to the value you specify in the *gray* and *alpha* parameters.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetGrayStrokeColor](#) (page 109)

Related Sample Code

HeadsUpUI

QuartzDemo

Declared In

CGContext.h

CGContextSetGrayStrokeColor

Sets the current stroke color to a value in the DeviceGray color space.

```
void CGContextSetGrayStrokeColor (
    CGContextRef c,
    CGFloat gray,
    CGFloat alpha
);
```

Parameters*c*

The graphics context for which to set the current stroke color.

gray

A value that specifies the desired gray level. The DeviceGray color space permits the specification of a value ranging from 0.0 (absolute black) to 1.0 (absolute white). Values outside this range are clamped to 0.0 or 1.0.

alpha

A value that specifies the opacity level. Values can range from 0.0 (transparent) to 1.0 (opaque). Values outside this range are clipped to 0.0 or 1.0.

Discussion

When you call this function, two things happen:

- Quartz sets the current stroke color space to DeviceGray. The DeviceGray color space is a single-dimension space in which color values are specified solely by the intensity of a gray value (from absolute black to absolute white).
- Quartz sets the current stroke color to the value you specify in the `gray` and `alpha` parameters.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetGrayFillColor](#) (page 108)

Related Sample Code

Teslameter

Declared In

CGContext.h

CGContextSetInterpolationQuality

Sets the level of interpolation quality for a graphics context.

```
void CGContextSetInterpolationQuality (
    CGContextRef c,
    CGInterpolationQuality quality
);
```

Parameters

c

The graphics context to modify.

quality

A `CGInterpolationQuality` constant that specifies the required level of interpolation quality. For possible values, see “[CGInterpolationQuality](#)” (page 137).

Discussion

Interpolation quality is merely a hint to the context—not all contexts support all interpolation quality levels.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextGetInterpolationQuality](#) (page 88)

Declared In

CGContext.h

CGContextSetLineCap

Sets the style for the endpoints of lines drawn in a graphics context.

```
void CGContextSetLineCap (
    CGContextRef c,
    CGLineCap cap
);
```

Parameters*c*

The graphics context to modify.

cap

A line cap style constant—[kCGLineCapButt](#) (page 139) (the default), [kCGLineCapRound](#) (page 139), or [kCGLineCapSquare](#) (page 139). See “[CGLineCap](#)” (page 138).

Availability

Available in iOS 2.0 and later.

Related Sample Code

Breadcrumb

GLImageProcessing

QuartzDemo

Declared In

CGContext.h

CGContextSetLineDash

Sets the pattern for dashed lines in a graphics context.

```
void CGContextSetLineDash (
    CGContextRef c,
    CGFloat phase,
    const CGFloat lengths[],
    size_t count
);
```

Parameters*c*

The graphics context to modify.

phase

A value that specifies how far into the dash pattern the line starts, in units of the user space. For example, passing a value of 3 means the line is drawn with the dash pattern starting at three units from its beginning. Passing a value of 0 draws a line starting with the beginning of a dash pattern.

lengths

An array of values that specify the lengths of the painted segments and unpainted segments, respectively, of the dash pattern—or `NULL` for no dash pattern.

For example, passing an array with the values `[2,3]` sets a dash pattern that alternates between a 2-user-space-unit-long painted segment and a 3-user-space-unit-long unpainted segment. Passing the values `[1,3,4,2]` sets the pattern to a 1-unit painted segment, a 3-unit unpainted segment, a 4-unit painted segment, and a 2-unit unpainted segment.

count

If the `lengths` parameter specifies an array, pass the number of elements in the array. Otherwise, pass 0.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

`CGContext.h`

CGContextSetLineJoin

Sets the style for the joins of connected lines in a graphics context.

```
void CGContextSetLineJoin (
    CGContextRef c,
    CGLineJoin join
);
```

Parameters

c

The graphics context to modify.

join

A line join value—`kCGLineJoinMiter` (page 139) (the default), `kCGLineJoinRound` (page 140), or `kCGLineJoinBevel` (page 140). See “`CGLineJoin`” (page 139).

Availability

Available in iOS 2.0 and later.

Related Sample Code

Breadcrumb

QuartzDemo

Declared In

`CGContext.h`

CGContextSetLineWidth

Sets the line width for a graphics context.


```
void CGContextSetLineWidth (
    CGContextRef c,
    CGFloat width
);
```

Parameters*c*

The graphics context to modify.

width

The new line width to use, in user space units. The value must be greater than 0.

Discussion

The default line width is 1 unit. When stroked, the line straddles the path, with half of the total width on either side.

Availability

Available in iOS 2.0 and later.

Related Sample Code

Breadcrumb

GLImageProcessing

PhotoScroller

QuartzDemo

Teslameter

Declared In

CGContext.h

CGContextSetMiterLimit

Sets the miter limit for the joins of connected lines in a graphics context.

```
void CGContextSetMiterLimit (
    CGContextRef c,
    CGFloat limit
);
```

Parameters*c*

The graphics context to modify.

limit

The miter limit to use.

Discussion

If the current line join style is set to `kCGLineJoinMiter` (see [CGContextSetLineJoin](#) (page 112)), Quartz uses the miter limit to determine whether the lines should be joined with a bevel instead of a miter. Quartz divides the length of the miter by the line width. If the result is greater than the miter limit, Quartz converts the style to a bevel.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextSetPatternPhase

Sets the pattern phase of a context.

```
void CGContextSetPatternPhase (
    CGContextRef c,
    CGSize phase
);
```

Parameters

c

The graphics context to modify.

phase

A pattern phase, specified in user space.

Discussion

The pattern phase is a translation that Quartz applies prior to drawing a pattern in the context. The pattern phase is part of the graphics state of a context, and the default pattern phase is (0, 0). Setting the pattern phase has the effect of temporarily changing the pattern matrix of any pattern you draw. For example, setting the context's pattern phase to (2, 3) has the effect of moving the start of pattern cell tiling to the point (2, 3) in default user space.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextSetRenderingIntent

Sets the rendering intent in the current graphics state.

```
void CGContextSetRenderingIntent (
    CGContextRef c,
    CGColorRenderingIntent intent
);
```

Parameters

c

The graphics context to modify.

intent

A rendering intent constant—[kCGRenderingIntentDefault](#) (page 48), [kCGRenderingIntentAbsoluteColorimetric](#) (page 48), [kCGRenderingIntentRelativeColorimetric](#) (page 48), [kCGRenderingIntentPerceptual](#) (page 49), or [kCGRenderingIntentSaturation](#) (page 49). For a discussion of these constants, see *CGColorSpace Reference*.

Discussion

The rendering intent specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context. If you do not explicitly set the rendering intent, Quartz uses perceptual rendering intent for drawing sampled images and relative colorimetric rendering intent for all other drawing.

Availability

Available in iOS 2.0 and later.

Declared In
CGContext.h

CGContextSetRGBFillColor

Sets the current fill color to a value in the DeviceRGB color space.

```
void CGContextSetRGBFillColor (
    CGContextRef c,
    CGFloat red,
    CGFloat green,
    CGFloat blue,
    CGFloat alpha
);
```

Parameters

c

The graphics context for which to set the current fill color.

red

The red intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from 0.0 (zero intensity) to 1.0 (full intensity).

green

The green intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from 0.0 (zero intensity) to 1.0 (full intensity).

blue

The blue intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from 0.0 (zero intensity) to 1.0 (full intensity).

alpha

A value that specifies the opacity level. Values can range from 0.0 (transparent) to 1.0 (opaque). Values outside this range are clipped to 0.0 or 1.0.

Discussion

When you call this function, two things happen:

- Quartz sets the current fill color space to DeviceRGB.
- Quartz sets the current fill color to the value specified by the *red*, *green*, *blue*, and *alpha* parameters.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetRGBStrokeColor](#) (page 116)

Related Sample Code

GLImageProcessing

PVRTextureLoader

QuartzDemo

ZoomingPDFViewer

Declared In
CGContext.h

CGContextSetRGBStrokeColor

Sets the current stroke color to a value in the DeviceRGB color space.

```
void CGContextSetRGBStrokeColor (
    CGContextRef c,
    CGFloat red,
    CGFloat green,
    CGFloat blue,
    CGFloat alpha
);
```

Parameters

c

The graphics context for which to set the current stroke color.

red

The red intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from 0.0 (zero intensity) to 1.0 (full intensity).

green

The green intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from 0.0 (zero intensity) to 1.0 (full intensity).

blue

The blue intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from 0.0 (zero intensity) to 1.0 (full intensity).

alpha

A value that specifies the opacity level. Values can range from 0.0 (transparent) to 1.0 (opaque). Values outside this range are clipped to 0.0 or 1.0.

Discussion

When you call this function, two things happen:

- Quartz sets the current stroke color space to DeviceRGB.
- Quartz sets the current stroke color to the value specified by the *red*, *green*, *blue*, and *alpha* parameters.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetRGBFillColor](#) (page 115)

Related Sample Code

Breadcrumb

GLImageProcessing

QuartzDemo

Teslameter

Declared In

CGContext.h

CGContextSetShadow

Enables shadowing in a graphics context.

```
void CGContextSetShadow (
    CGContextRef context,
    CGSize offset,
    CGFloat blur
);
```

Parameters*context*

A graphics context.

*offset*Specifies a translation of the context's coordinate system, to establish an offset for the shadow ($\{0, 0\}$ specifies a light source immediately above the screen).*blur*

A non-negative number specifying the amount of blur.

Discussion

Shadow parameters are part of the graphics state in a context. After shadowing is set, all objects drawn are shadowed using a black color with 1/3 alpha (i.e., $\text{RGBA} = \{0, 0, 0, 1.0/3.0\}$) in the DeviceRGB color space.

To turn off shadowing:

- Use the standard save/restore mechanism for the graphics state.
- Use [CGContextSetShadowWithColor](#) (page 117) to set the shadow color to a fully transparent color (or pass `NULL` as the color).

Availability

Available in iOS 2.0 and later.

Declared In`CGContext.h`**CGContextSetShadowWithColor**

Enables shadowing with color a graphics context.

```
void CGContextSetShadowWithColor (
    CGContextRef context,
    CGSize offset,
    CGFloat blur,
    CGColorRef color
);
```

Parameters*context*

The graphics context to modify.

offset

Specifies a translation in base-space.

blur

A non-negative number specifying the amount of blur.

color

Specifies the color of the shadow, which may contain a non-opaque alpha value. If `NULL`, then shadowing is disabled.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetShadow](#) (page 116)

Declared In

`CGContext.h`

CGContextSetShouldAntialias

Sets anti-aliasing on or off for a graphics context.

```
void CGContextSetShouldAntialias (
    CGContextRef c,
    bool shouldAntialias
);
```

Parameters

c

The graphics context to modify.

shouldAntialias

A Boolean value that specifies whether anti-aliasing should be turned on. Anti-aliasing is turned on by default when a window or bitmap context is created. It is turned off for other types of contexts.

Discussion

Anti-aliasing is a graphics state parameter.

Availability

Available in iOS 2.0 and later.

Declared In

`CGContext.h`

CGContextSetShouldSmoothFonts

Enables or disables font smoothing in a graphics context.

```
void CGContextSetShouldSmoothFonts (
    CGContextRef c,
    bool shouldSmoothFonts
);
```

Parameters

c

The graphics context to modify.

shouldSmoothFonts

A Boolean value that specifies whether to enable font smoothing.

Discussion

There are cases, such as rendering to a bitmap, when font smoothing is not appropriate and should be disabled. Note that some contexts (such as PostScript contexts) do not support font smoothing.

This parameter is part of the graphics state.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextSetShouldSubpixelPositionFonts

Enables or disables subpixel positioning in a graphics context.

```
void CGContextSetShouldSubpixelPositionFonts(  
    CGContextRef context,  
    bool shouldSubpixelPositionFonts  
);
```

Parameters

context

A graphics context.

shouldSubpixelPositionFonts

A Boolean value that specifies whether to enable subpixel positioning.

Discussion

When enabled, the graphics context may position glyphs on nonintegral pixel boundaries. When disabled, the position of glyphs are always forced to integral pixel boundaries.

This parameter is part of the graphics state.

Availability

Available in iOS 4.0 and later.

Declared In

CGContext.h

CGContextSetShouldSubpixelQuantizeFonts

Enables or disables subpixel quantization in a graphics context.

```
void CGContextSetShouldSubpixelQuantizeFonts(  
    CGContextRef context,  
    bool shouldSubpixelQuantizeFonts  
);
```

Parameters

context

A graphics context.

shouldSubpixelQuantizeFonts

A Boolean value that specifies whether to enable subpixel quantization.

Discussion

When enabled, the graphics context may quantize the subpixel positions of glyphs.

This parameter is part of the graphics state.

Availability

Available in iOS 4.0 and later.

Declared In

CGContext.h

CGContextSetStrokeColor

Sets the current stroke color.

```
void CGContextSetStrokeColor (
    CGContextRef c,
    const CGFloat components[]
);
```

Parameters

c

The graphics context for which to set the current stroke color.

components

An array of intensity values describing the color to set. The number of array elements must equal the number of components in the current stroke color space, plus an additional component for the alpha value.

Discussion

The current stroke color space must not be a pattern color space. For information on setting the stroke color when using a pattern color space, see [CGContextSetStrokePattern](#) (page 121). Note that the preferred API is now [CGContextSetStrokeColorWithColor](#) (page 121).

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextSetStrokeColorSpace

Sets the stroke color space in a graphics context.

```
void CGContextSetStrokeColorSpace (
    CGContextRef c,
    CGColorSpaceRef colorspace
);
```

Parameters

c

The graphics context for the new stroke color space.

colorspace

The new stroke color space. Quartz retains this object; upon return, you may safely release it.

Discussion

As a side effect when you call this function, Quartz assigns an appropriate initial value to the stroke color, based on the color space you specify. To change this value, call [CGContextSetStrokeColor](#) (page 120). Note that the preferred API is now [CGContextSetStrokeColorWithColor](#) (page 121).

Availability

Available in iOS 2.0 and later.

Related Sample Code

avTouch

QuartzDemo

SpeakHere

Declared In

CGContext.h

CGContextSetStrokeColorWithColor

Sets the current stroke color in a context, using a Quartz color.

```
void CGContextSetStrokeColorWithColor (
    CGContextRef c,
    CGColorRef color
);
```

Parameters

c

The graphics context to modify.

color

The new stroke color.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextSetStrokeColor](#) (page 120)

Related Sample Code

AccelerometerGraph

QuartzDemo

WeatherMap

Declared In

CGContext.h

CGContextSetStrokePattern

Sets the stroke pattern in the specified graphics context.

```
void CGContextSetStrokePattern (
    CGContextRef c,
    CGPatternRef pattern,
    const CGFloat components[]
);
```

Parameters*c*

The graphics context to modify.

pattern

A pattern for stroking. Quartz retains this object; upon return, you may safely release it.

components

If the specified pattern is an uncolored (or masking) pattern, pass an array of intensity values that specify the color to use when the pattern is painted. The number of array elements must equal the number of components in the base space of the stroke pattern color space, plus an additional component for the alpha value.

If the specified pattern is a colored pattern, pass an alpha value.

Discussion

The current stroke color space must be a pattern color space. Otherwise, the result of calling this function is undefined. If you want to set a stroke color, not a stroke pattern, use [CGContextSetStrokeColorWithColor](#) (page 121).

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextSetTextDrawingMode

Sets the current text drawing mode.

```
void CGContextSetTextDrawingMode (
    CGContextRef c,
    CGTextDrawingMode mode
);
```

Parameters*c*

A graphics context.

mode

A text drawing mode (such as [kCGTextFill](#) (page 140) or [kCGTextStroke](#) (page 140)) that specifies how Quartz renders individual glyphs in a graphics context. See “[CGTextDrawingMode](#)” (page 140) for a complete list.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextSetTextMatrix

Sets the current text matrix.

```
void CGContextSetTextMatrix (
    CGContextRef c,
    CGAffineTransform t
);
```

Parameters*c*

A graphics context.

transform

The text matrix to set.

Discussion

The text matrix specifies the transform from text space to user space. To produce the final text rendering matrix that is used to actually draw the text on the page, Quartz concatenates the text matrix with the current transformation matrix and other parameters from the graphics state.

Note that the text matrix is *not* a part of the graphics state—saving or restoring the graphics state has no effect on the text matrix. The text matrix is an attribute of the graphics context, not of the current font.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextSetTextPosition

Sets the location at which text is drawn.

```
void CGContextSetTextPosition (
    CGContextRef c,
    CGFloat x,
    CGFloat y
);
```

Parameters*c*

A graphics context.

x

A value for the x-coordinate at which to draw the text, in user space coordinates.

y

A value for the y-coordinate at which to draw the text, in user space coordinates.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextShowGlyphs

Displays an array of glyphs at the current text position.

```
void CGContextShowGlyphs (
    CGContextRef c,
    const CGGlyph g[],
    size_t count
);
```

Parameters*c*

The graphics context in which to display the glyphs.

glyphs

An array of glyphs to display.

count

The total number of glyphs passed in the *g* parameter.

Discussion

This function displays an array of glyphs at the current text position, a point specified by the current text matrix.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextShowGlyphsAtPoint](#) (page 124)

[CGContextShowText](#) (page 127)

[CGContextShowTextAtPoint](#) (page 127)

[CGContextShowGlyphsWithAdvances](#) (page 126)

Declared In

CGContext.h

CGContextShowGlyphsAtPoint

Displays an array of glyphs at a position you specify.

```
void CGContextShowGlyphsAtPoint (
    CGContextRef c,
    CGFloat x,
    CGFloat y,
    const CGGlyph glyphs[],
    size_t count
);
```

Parameters*c*

The graphics context in which to display the glyphs.

x

A value for the x-coordinate of the user space at which to display the glyphs.

y

A value for the y-coordinate of the user space at which to display the glyphs.

glyphs

An array of glyphs to display.

*count*The total number of glyphs passed in the *glyphs* parameter.**Discussion**

This function displays an array of glyphs at the specified position in the user space.

Availability

Available in iOS 2.0 and later.

See Also[CGContextShowGlyphs](#) (page 124)[CGContextShowText](#) (page 127)[CGContextShowTextAtPoint](#) (page 127)[CGContextShowGlyphsWithAdvances](#) (page 126)**Related Sample Code**

QuartzDemo

Declared In

CGContext.h

CGContextShowGlyphsAtPositions

Draws glyphs at the provided position.

```
void CGContextShowGlyphsAtPositions(
    CGContextRef context,
    const CGGlyph glyphs[],
    const CGPoint positions[],
    size_t count
);
```

Parameters*c*

The graphics context in which to display the glyphs.

glyphs

An array of Quartz glyphs.

positions

The positions for the glyphs. Each item in this array matches with the glyph at the corresponding index in the *glyphs* array. The position of each glyph is specified in text space, and, as a consequence, is transformed through the text matrix to user space.

count

The number of items in the *glyphs* array.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextShowGlyphsWithAdvances

Draws an array of glyphs with varying offsets.

```
void CGContextShowGlyphsWithAdvances (
    CGContextRef c,
    const CGGlyph glyphs[],
    const CGSize advances[],
    size_t count
);
```

Parameters

c

The graphics context in which to display the glyphs.

glyphs

An array of Quartz glyphs.

advances

An array of offset values associated with each glyph in the array. Each value specifies the offset from the previous glyph's origin to the origin of the corresponding glyph. Offsets are specified in user space.

count

The number of glyphs in the specified array.

Discussion

This function draws an array of glyphs at the current point specified by the text matrix.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextShowGlyphs](#) (page 124)

[CGContextShowText](#) (page 127)

[CGContextShowTextAtPoint](#) (page 127)

[CGContextShowGlyphsAtPoint](#) (page 124)

Declared In

CGContext.h

CGContextShowText

Displays a character array at the current text position, a point specified by the current text matrix.

```
void CGContextShowText (
    CGContextRef c,
    const char *string,
    size_t length
);
```

Parameters

c

A graphics context.

string

An array of characters to draw.

length

The length of the array specified in the *bytes* parameter.

Discussion

Quartz uses font data provided by the system to map each byte of the array through the encoding vector of the current font to obtain the glyph to display. Note that the font must have been set using [CGContextSelectFont](#) (page 98). Don't use `CGContextShowText` in conjunction with [CGContextSetFont](#) (page 107).

Availability

Available in iOS 2.0 and later.

See Also

[CGContextShowTextAtPoint](#) (page 127)

[CGContextShowGlyphs](#) (page 124)

[CGContextShowGlyphsAtPoint](#) (page 124)

[CGContextShowGlyphsWithAdvances](#) (page 126)

Declared In

`CGContext.h`

CGContextShowTextAtPoint

Displays a character string at a position you specify.

```
void CGContextShowTextAtPoint (
    CGContextRef c,
    CGFloat x,
    CGFloat y,
    const char *string,
    size_t length
);
```

Parameters

c

A graphics context .

x

A value for the x-coordinate (in user space) at which to display the text.

y

A value for the y-coordinate (in user space) at which to display the text.

string

An array of characters to draw.

length

The length of the array specified in the `string` parameter.

Discussion

Quartz uses font data provided by the system to map each byte of the array through the encoding vector of the current font to obtain the glyph to display. Note that the font must have been set using [CGContextSelectFont](#) (page 98). Don't use `CGContextShowTextAtPoint` in conjunction with `CGContextSetFont` (page 107).

Availability

Available in iOS 2.0 and later.

See Also

[CGContextShowText](#) (page 127)

[CGContextShowGlyphs](#) (page 124)

[CGContextShowGlyphsAtPoint](#) (page 124)

[CGContextShowGlyphsWithAdvances](#) (page 126)

Related Sample Code

QuartzDemo

Declared In

`CGContext.h`

CGContextStrokeEllipseInRect

Strokes an ellipse that fits inside the specified rectangle.

```
void CGContextStrokeEllipseInRect (
    CGContextRef context,
    CGRect rect
);
```

Parameters*context*

A graphics context.

rect

A rectangle that defines the area for the ellipse to fit in.

Availability

Available in iOS 2.0 and later.

Related Sample Code

GLImageProcessing

QuartzDemo

Declared In

`CGContext.h`

CGContextStrokeLineSegments

Strokes a sequence of line segments.

```
void CGContextStrokeLineSegments (
    CGContextRef c,
    const CGPoint points[],
    size_t count
);
```

Parameters

c

A graphics context.

points

An array of points, organized as pairs—the starting point of a line segment followed by the ending point of a line segment. For example, the first point in the array specifies the starting position of the first line, the second point specifies the ending position of the first line, the third point specifies the starting position of the second line, and so forth.

count

The number of points in the *points* array.

Discussion

This function is equivalent to the following code:

```
CGContextBeginPath (context);
for (k = 0; k < count; k += 2) {
    CGContextMoveToPoint(context, s[k].x, s[k].y);
    CGContextAddLineToPoint(context, s[k+1].x, s[k+1].y);
}
CGContextStrokePath(context);
```

Availability

Available in iOS 2.0 and later.

Related Sample Code

AccelerometerGraph

QuartzDemo

Declared In

CGContext.h

CGContextStrokePath

Paints a line along the current path.

```
void CGContextStrokePath (
    CGContextRef c
);
```

Parameters

c

A graphics context.

Discussion

Quartz uses the line width and stroke color of the graphics state to paint the path. As a side effect when you call this function, Quartz clears the current path.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextDrawPath](#) (page 79)

[CGContextFillPath](#) (page 85)

[CGContextEOFillPath](#) (page 84)

Related Sample Code

AccelerometerGraph

Breadcrumb

GLImageProcessing

QuartzDemo

Teslameter

Declared In

CGContext.h

CGContextStrokeRect

Paints a rectangular path.

```
void CGContextStrokeRect (
    CGContextRef c,
    CGRect rect
);
```

Parameters

c

A graphics context .

rect

A rectangle, specified in user space coordinates.

Discussion

Quartz uses the line width and stroke color of the graphics state to paint the path.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextStrokeRectWithWidth](#) (page 131)

Related Sample Code

avTouch

PhotoScroller

QuartzDemo

SpeakHere

Declared In

CGContext.h

CGContextStrokeRectWithWidth

Paints a rectangular path, using the specified line width.

```
void CGContextStrokeRectWithWidth (
    CGContextRef c,
    CGRect rect,
    CGFloat width
);
```

Parameters*c*

A graphics context.

rect

A rectangle, in user space coordinates.

width

A value, in user space units, that is greater than zero. This value does not affect the line width values in the current graphics state.

Discussion

Aside from the line width value, Quartz uses the current attributes of the graphics state (such as stroke color) to paint the line. The line straddles the path, with half of the total width on either side.

As a side effect when you call this function, Quartz clears the current path.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextStrokeRect](#) (page 130)

Related Sample Code

QuartzDemo

Declared In

CGContext.h

CGContextSynchronize

Marks a window context for update.

```
void CGContextSynchronize (
    CGContextRef c
);
```

Parameters*c*

The window context to synchronize. If you pass a PDF context or a bitmap context, this function does nothing.

Discussion

When you call this function, all drawing operations since the last update are flushed at the next regular opportunity. Under normal conditions, you do not need to call this function.

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

CGContextTranslateCTM

Changes the origin of the user coordinate system in a context.

```
void CGContextTranslateCTM (
    CGContextRef c,
    CGFloat tx,
    CGFloat ty
);
```

Parameters

c

A graphics context.

tx

The amount to displace the x-axis of the coordinate space, in units of the user space, of the specified context.

ty

The amount to displace the y-axis of the coordinate space, in units of the user space, of the specified context.

Availability

Available in iOS 2.0 and later.

Related Sample Code

avTouch

QuartzDemo

SpeakHere

TheElements

ZoomingPDFViewer

Declared In

CGContext.h

Data Types

CGContextRef

An opaque type that represents a Quartz 2D drawing environment.

```
typedef struct CGContext * CGContextRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGContext.h

Constants

CGBlendMode

Compositing operations for images.

```
enum CGBlendMode {
    kCGBlendModeNormal,
    kCGBlendModeMultiply,
    kCGBlendModeScreen,
    kCGBlendModeOverlay,
    kCGBlendModeDarken,
    kCGBlendModeLighten,
    kCGBlendModeColorDodge,
    kCGBlendModeColorBurn,
    kCGBlendModeSoftLight,
    kCGBlendModeHardLight,
    kCGBlendModeDifference,
    kCGBlendModeExclusion,
    kCGBlendModeHue,
    kCGBlendModeSaturation,
    kCGBlendModeColor,
    kCGBlendModeLuminosity,
    kCGBlendModeClear,
    kCGBlendModeCopy,
    kCGBlendModeSourceIn,
    kCGBlendModeSourceOut,
    kCGBlendModeSourceAtop,
    kCGBlendModeDestinationOver,
    kCGBlendModeDestinationIn,
    kCGBlendModeDestinationOut,
    kCGBlendModeDestinationAtop,
    kCGBlendModeXOR,
    kCGBlendModePlusDarker,
    kCGBlendModePlusLighter
};
typedef enum CGBlendMode CGBlendMode;
```

Constants

kCGBlendModeNormal

Paints the source image samples over the background image samples.

Available in iOS 2.0 and later.

Declared in CGContext.h.

kCGBlendModeMultiply

Multiplies the source image samples with the background image samples. This results in colors that are at least as dark as either of the two contributing sample colors.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeScreen

Multiplies the inverse of the source image samples with the inverse of the background image samples. This results in colors that are at least as light as either of the two contributing sample colors.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeOverlay

Either multiplies or screens the source image samples with the background image samples, depending on the background color. The result is to overlay the existing image samples while preserving the highlights and shadows of the background. The background color mixes with the source image to reflect the lightness or darkness of the background.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeDarken

Creates the composite image samples by choosing the darker samples (either from the source image or the background). The result is that the background image samples are replaced by any source image samples that are darker. Otherwise, the background image samples are left unchanged.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeLighten

Creates the composite image samples by choosing the lighter samples (either from the source image or the background). The result is that the background image samples are replaced by any source image samples that are lighter. Otherwise, the background image samples are left unchanged.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeColorDodge

Brightens the background image samples to reflect the source image samples. Source image sample values that specify black do not produce a change.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeColorBurn

Darkens the background image samples to reflect the source image samples. Source image sample values that specify white do not produce a change.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModeSoftLight`

Either darkens or lightens colors, depending on the source image sample color. If the source image sample color is lighter than 50% gray, the background is lightened, similar to dodging. If the source image sample color is darker than 50% gray, the background is darkened, similar to burning. If the source image sample color is equal to 50% gray, the background is not changed. Image samples that are equal to pure black or pure white produce darker or lighter areas, but do not result in pure black or white. The overall effect is similar to what you'd achieve by shining a diffuse spotlight on the source image. Use this to add highlights to a scene.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModeHardLight`

Either multiplies or screens colors, depending on the source image sample color. If the source image sample color is lighter than 50% gray, the background is lightened, similar to screening. If the source image sample color is darker than 50% gray, the background is darkened, similar to multiplying. If the source image sample color is equal to 50% gray, the source image is not changed. Image samples that are equal to pure black or pure white result in pure black or white. The overall effect is similar to what you'd achieve by shining a harsh spotlight on the source image. Use this to add highlights to a scene.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModeDifference`

Subtracts either the source image sample color from the background image sample color, or the reverse, depending on which sample has the greater brightness value. Source image sample values that are black produce no change; white inverts the background color values.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModeExclusion`

Produces an effect similar to that produced by `kCGBlendModeDifference`, but with lower contrast. Source image sample values that are black don't produce a change; white inverts the background color values.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModeHue`

Uses the luminance and saturation values of the background with the hue of the source image.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModeSaturation`

Uses the luminance and hue values of the background with the saturation of the source image. Areas of the background that have no saturation (that is, pure gray areas) don't produce a change.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeColor

Uses the luminance values of the background with the hue and saturation values of the source image. This mode preserves the gray levels in the image. You can use this mode to color monochrome images or to tint color images.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeLuminosity

Uses the hue and saturation of the background with the luminance of the source image. This mode creates an effect that is inverse to the effect created by `kCGBlendModeColor`.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeClear

$R = 0$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeCopy

$R = S$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeSourceIn

$R = S * D_a$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeSourceOut

$R = S * (1 - D_a)$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeSourceAtop

$R = S * D_a + D * (1 - S_a)$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeDestinationOver

$R = S * (1 - D_a) + D$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeDestinationIn

$R = D * S_a$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGBlendModeDestinationOut

$R = D * (1 - S_a)$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModeDestinationAtop`

$$R = S*(1 - Da) + D*Sa$$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModeXOR`

$$R = S*(1 - Da) + D*(1 - Sa)$$
. This XOR mode is only nominally related to the classical bitmap XOR operation, which is not supported by Quartz 2D.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModePlusDarker`

$$R = \text{MAX}(0, (1 - D) + (1 - S))$$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGBlendModePlusLighter`

$$R = \text{MIN}(1, S + D)$$

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

Discussion

The blend mode constants introduced in Mac OS X v10.5 represent the Porter-Duff blend modes. The symbols in the equations for these blend modes are:

- R is the premultiplied result
- S is the source color, and includes alpha
- D is the destination color, and includes alpha
- Ra, Sa, and Da are the alpha components of R, S, and D

You can find more information on blend modes, including examples of images produced using them, and many mathematical descriptions of the modes, in *PDF Reference, Fourth Edition, Version 1.5*, Adobe Systems, Inc. If you are a former QuickDraw developer, it may be helpful for you to think of blend modes as an alternative to transfer modes

For examples of using blend modes see "Setting Blend Modes" and "Using Blend Modes With Images" in *Quartz 2D Programming Guide*.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGContext.h`

CGInterpolationQuality

Levels of interpolation quality for rendering an image.

```
enum CGInterpolationQuality {
    kCGInterpolationDefault = 0,
    kCGInterpolationNone = 1,
    kCGInterpolationLow = 2,
    kCGInterpolationMedium = 4,
    kCGInterpolationHigh = 3
};
typedef enum CGInterpolationQuality CGInterpolationQuality;
```

Constants**kCGInterpolationDefault**

The default level of quality.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.**kCGInterpolationNone**

No interpolation.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.**kCGInterpolationLow**

A low level of interpolation quality. This setting may speed up image rendering.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.**kCGInterpolationMedium**

A medium level of interpolation quality. This setting is slower than the low setting but faster than the high setting.

Available in iOS 4.0 and later.

Declared in `CGContext.h`.**kCGInterpolationHigh**

A high level of interpolation quality. This setting may slow down image rendering.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.**Discussion**

You use the function `CGContextSetInterpolationQuality` (page 110) to set the interpolation quality in a graphics context.

Declared In`CGContext.h`**CGLineCap**

Styles for rendering the endpoint of a stroked line.

```
enum CGLineCap {
    kCGLineCapButt,
    kCGLineCapRound,
    kCGLineCapSquare
};
typedef enum CGLineCap CGLineCap;
```

Constants**kCGLineCapButt**

A line with a squared-off end. Quartz draws the line to extend only to the exact endpoint of the path. This is the default.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGLineCapRound

A line with a rounded end. Quartz draws the line to extend beyond the endpoint of the path. The line ends with a semicircular arc with a radius of 1/2 the line's width, centered on the endpoint.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

kCGLineCapSquare

A line with a squared-off end. Quartz extends the line beyond the endpoint of the path for a distance equal to half the line width.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

Discussion

A line cap specifies the method used by [CGContextStrokePath](#) (page 129) to draw the endpoint of the line. To change the line cap style in a graphics context, you use the function [CGContextSetLineCap](#) (page 111).

Declared In

`CGContext.h`

CGLineJoin

Junction types for stroked lines.

```
enum CGLineJoin {
    kCGLineJoinMiter,
    kCGLineJoinRound,
    kCGLineJoinBevel
};
typedef enum CGLineJoin CGLineJoin;
```

Constants**kCGLineJoinMiter**

A join with a sharp (angled) corner. Quartz draws the outer sides of the lines beyond the endpoint of the path, until they meet. If the length of the miter divided by the line width is greater than the miter limit, a bevel join is used instead. This is the default. To set the miter limit, see [CGContextSetMiterLimit](#) (page 113).

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGLineJoinRound`

A join with a rounded end. Quartz draws the line to extend beyond the endpoint of the path. The line ends with a semicircular arc with a radius of 1/2 the line's width, centered on the endpoint.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGLineJoinBevel`

A join with a squared-off end. Quartz draws the line to extend beyond the endpoint of the path, for a distance of 1/2 the line's width.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

Discussion

A line join specifies how `CGContextStrokePath` (page 129) draws the junction between connected line segments. To set the line join style in a graphics context, you use the function `CGContextSetLineJoin` (page 112).

Declared In

`CGContext.h`

CGTextDrawingMode

Modes for rendering text.

```
enum CGTextDrawingMode {
    kCGTextFill,
    kCGTextStroke,
    kCGTextFillStroke,
    kCGTextInvisible,
    kCGTextFillClip,
    kCGTextStrokeClip,
    kCGTextFillStrokeClip,
    kCGTextClip
};
typedef enum CGTextDrawingMode CGTextDrawingMode;
```

Constants

`kCGTextFill`

Perform a fill operation on the text.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGTextStroke`

Perform a stroke operation on the text.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGTextFillStroke`

Perform fill, then stroke operations on the text.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGTextInvisible`

Do not draw the text, but do update the text position.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGTextFillClip`

Perform a fill operation, then intersect the text with the current clipping path.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGTextStrokeClip`

Perform a stroke operation, then intersect the text with the current clipping path.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGTextFillStrokeClip`

Perform fill then stroke operations, then intersect the text with the current clipping path.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGTextClip`

Specifies to intersect the text with the current clipping path. This mode does not paint the text.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

Discussion

You provide a text drawing mode constant to the function `CGContextSetTextDrawingMode` (page 122) to set the current text drawing mode for a graphics context. Text drawing modes determine how Quartz renders individual glyphs onscreen. For example, you can set a text drawing mode to draw text filled in or outlined (stroked) or both. You can also create special effects with the text clipping drawing modes, such as clipping an image to a glyph shape.

Declared In

`CGContext.h`

CGTextEncoding

Text encodings for fonts.

```
enum CGTextEncoding {
    kCGEncodingFontSpecific,
    kCGEncodingMacRoman
};
typedef enum CGTextEncoding CGTextEncoding;
```

Constants

`kCGEncodingFontSpecific`

The built-in encoding of the font.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGEncodingMacRoman`

The MacRoman encoding. MacRoman is an ASCII variant originally created for use in the Mac OS, in which characters 127 and lower are ASCII, and characters 128 and higher are non-English characters and symbols.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

Discussion

For more information on setting the font in a graphics context, see [CGContextSelectFont](#) (page 98).

Declared In

`CGContext.h`

CGDataConsumer Reference

Derived From:	CType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGDataConsumer.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGDataConsumerRef` opaque type abstracts the data-writing task and eliminates the need for applications to manage data through a raw memory buffer. You can use data consumer objects to write image or PDF data and all, except for [CGDataConsumerCreateWithCFData](#) (page 144), are available in Mac OS X v10.0 or later.

If your application runs in Mac OS X v10.4 or later, you should use `CGImageDestination` objects rather than data consumers. See *CGImageDestination Reference*.

Functions by Task

Creating Data Consumers

[CGDataConsumerCreate](#) (page 144)

Creates a data consumer that uses callback functions to write data.

[CGDataConsumerCreateWithURL](#) (page 145)

Creates a data consumer that writes data to a location specified by a URL.

[CGDataConsumerCreateWithCFData](#) (page 144)

Creates a data consumer that writes to a CFData object.

Getting the CType ID

[CGDataConsumerGetTypeID](#) (page 145)

Returns the Core Foundation type identifier for Quartz data consumers.

Retaining and Releasing Data Consumers

[CGDataConsumerRelease](#) (page 145)

Decrements the retain count of a data consumer.

[CGDataConsumerRetain](#) (page 146)

Increments the retain count of a data consumer.

Functions

CGDataConsumerCreate

Creates a data consumer that uses callback functions to write data.

```
CGDataConsumerRef CGDataConsumerCreate (  
    void *info,  
    const CGDataConsumerCallbacks *callbacks  
);
```

Parameters

info

A pointer to data of any type or NULL. When Quartz calls the functions specified in the `callbacks` parameter, it passes this pointer as the `info` parameter.

callbacks

A pointer to a `CGDataConsumerCallbacks` structure that specifies the callback functions you implement to copy data sent to the consumer and to handle the consumer's basic memory management. For a complete description, see [CGDataConsumerCallbacks](#) (page 148).

Return Value

A new data consumer object. You are responsible for releasing this object using [CGDataConsumerRelease](#) (page 145).

Availability

Declared In

`CGDataConsumer.h`

CGDataConsumerCreateWithCFData

Creates a data consumer that writes to a CFData object.

```
CGDataConsumerRef CGDataConsumerCreateWithCFData (  
    CFMutableDataRef data  
);
```

Parameters

data

The CFData object to write to.

Return Value

A new data consumer object. You are responsible for releasing this object using [CGDataConsumerRelease](#) (page 145).

Discussion

You can use this function when you need to represent Quartz data as a CFData type. For example, you might create a CFData object that you then copy to the pasteboard.

Availability**Declared In**

CGDataConsumer.h

CGDataConsumerCreateWithURL

Creates a data consumer that writes data to a location specified by a URL.

```
CGDataConsumerRef CGDataConsumerCreateWithURL (  
    CFURLRef url  
);
```

Parameters

url

A CFURL object that specifies the data destination.

Return Value

A new data consumer object. You are responsible for releasing this object using [CGDataConsumerRelease](#) (page 145).

Availability**Declared In**

CGDataConsumer.h

CGDataConsumerGetTypeID

Returns the Core Foundation type identifier for Quartz data consumers.

```
CTypeID CGDataConsumerGetTypeID (  
    void  
);
```

Return Value

The Core Foundation identifier for the opaque type [CGDataConsumerRef](#) (page 148).

Availability**Declared In**

CGDataConsumer.h

CGDataConsumerRelease

Decrements the retain count of a data consumer.

```
void CGDataConsumerRelease (
    CGDataConsumerRef consumer
);
```

Parameters

consumer

The data consumer to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `consumer` parameter is `NULL`.

Availability**Declared In**

`CGDataConsumer.h`

CGDataConsumerRetain

Increments the retain count of a data consumer.

```
CGDataConsumerRef CGDataConsumerRetain (
    CGDataConsumerRef consumer
);
```

Parameters

consumer

The data consumer to retain.

Return Value

The same data consumer you passed in as the `consumer` parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `consumer` parameter is `NULL`.

Availability**Declared In**

`CGDataConsumer.h`

Callbacks

CGDataConsumerPutBytesCallback

Copies data from a Quartz-supplied buffer into a data consumer.

```
size_t (*CGDataConsumerPutBytesCallback) (
    void *info,
    const void *buffer,
    size_t count
);
```

If you name your function `MyConsumerPutBytes`, you would declare it like this:

```
size_t MyConsumerPutBytes (  
    void *info,  
    const void *buffer,  
    size_t count  
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the pointer supplied to [CGDataConsumerCreate](#) (page 144).

buffer

The Quartz-supplied buffer from which you copy the specified number of bytes.

count

The number of bytes to copy.

Return Value

The number of bytes copied. If no more data can be written to the consumer, you should return 0.

Discussion

When Quartz is ready to send data to the consumer, your function is called. It should copy the specified number of bytes from `buffer` into some resource under your control—for example, a file.

For information on how to associate your callback function with a data consumer, see [CGDataConsumerCreate](#) (page 144) and [CGDataConsumerCallbacks](#) (page 148).

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataConsumer.h`

CGDataConsumerReleaseInfoCallback

Releases any private data or resources associated with the data consumer.

```
void (*CGDataConsumerReleaseInfoCallback) (  
    void *info  
);
```

If you name your function `MyConsumerReleaseInfo`, you would declare it like this:

```
void MyConsumerReleaseInfo (  
    void *info  
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to [CGDataConsumerCreate](#) (page 144).

Discussion

When Quartz frees a data consumer that has an associated release function, the release function is called.

For information on how to associate your callback function with a data consumer, see [CGDataConsumerCreate](#) (page 144) and [CGDataConsumerCallbacks](#) (page 148).

Availability

Available in iOS 2.0 and later.

Declared In

CGDataConsumer.h

Data Types

CGDataConsumerCallbacks

A structure that contains pointers to callback functions that manage the copying of data for a data consumer.

```
struct CGDataConsumerCallbacks {
    CGDataConsumerPutBytesCallback putBytes;
    CGDataConsumerReleaseInfoCallback releaseConsumer;
};
typedef struct CGDataConsumerCallbacks CGDataConsumerCallbacks;
```

Fields

putBytes

A pointer to a function that copies data to the data consumer. For more information, see [CGDataConsumerPutBytesCallback](#) (page 146).

releaseConsumer

A pointer to a function that handles clean-up for the data consumer, or NULL. For more information, see [CGDataConsumerReleaseInfoCallback](#) (page 147)

Discussion

The functions specified by the `CGDataConsumerCallbacks` structure are responsible for copying data that Quartz sends to your consumer and for handling the consumer's basic memory management. You supply a `CGDataConsumerCallbacks` structure to the function [CGDataConsumerCreate](#) (page 144) to create a data consumer.

Availability

Available in iOS 2.0 and later.

Declared In

CGDataConsumer.h

CGDataConsumerRef

An opaque type that handles the storage of data supplied by Quartz functions.

```
typedef struct CGDataConsumer *CGDataConsumerRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGDataConsumer.h

CGDataProvider Reference

Derived From:	<i>CType Reference</i>
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGDataProvider.h

Overview

The CGDataProvider header file declares a data type that supplies Quartz functions with data. Data provider objects abstract the data-access task and eliminate the need for applications to manage data through a raw memory buffer.

For information on how to use CGDataProvider functions, see *Quartz 2D Programming Guide* Programming Guide.

See also *CGDataConsumer Reference*.

Functions

CGDataProviderCopyData

Returns a copy of the provider's data.

```
CFDataRef CGDataProviderCopyData(
    CGDataProviderRef provider
);
```

Parameters

provider

The data provider whose data you want to copy.

Return Value

A new data object containing a copy of the provider's data. You are responsible for releasing this object.

Availability

Available in iOS 2.0 and later.

Related Sample Code

GLImageProcessing

Declared In

CGDataProvider.h

CGDataProviderCreateDirect

Creates a Quartz direct-access data provider.

```
CGDataProviderRef CGDataProviderCreateDirect (
    void *info,
    off_t size,
    const CGDataProviderDirectCallbacks *callbacks
);
```

Parameters

info

A pointer to data of any type or NULL. When Quartz calls the functions specified in the `callbacks` parameter, it sends each of the functions this pointer.

size

The number of bytes of data to provide.

callbacks

A pointer to a `CGDataProviderDirectCallbacks` structure that specifies the callback functions you implement to handle the data provider's basic memory management.

Return Value

A new data provider. You are responsible for releasing this object using [CGDataProviderRelease](#) (page 155).

Discussion

You use this function to create a direct-access data provider that uses callback functions to read data from your program in a single block.

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderCreateSequential

Creates a Quartz sequential-access data provider.

```
CGDataProviderRef CGDataProviderCreateSequential (
    void *info,
    const CGDataProviderSequentialCallbacks *callbacks
);
```

Parameters

info

A pointer to data of any type or NULL. When Quartz calls the functions specified in the `callbacks` parameter, it sends each of the functions this pointer.

callbacks

A pointer to a `CGDataProviderSequentialCallbacks` structure that specifies the callback functions you implement to handle the data provider's basic memory management.

Return Value

A new data provider. You are responsible for releasing this object using [CGDataProviderRelease](#) (page 155).

Discussion

You use this function to create a sequential-access data provider that uses callback functions to read data from your program in a single block.

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

CGDataProviderCreateWithCFData

Creates a Quartz data provider that reads from a CFData object.

```
CGDataProviderRef CGDataProviderCreateWithCFData (
    CFDataRef data
);
```

Parameters

data

The CFData object to read from.

Return Value

A new data provider. You are responsible for releasing this object using [CGDataProviderRelease](#) (page 155).

Discussion

You can use this function when you need to represent Quartz data as a CFData type. For example, you might create a CFData object when reading data from the pasteboard.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGDataProvider.h

CGDataProviderCreateWithData

Creates a Quartz direct-access data provider that uses data your program supplies.

```
CGDataProviderRef CGDataProviderCreateWithData (
    void *info,
    const void *data,
    size_t size,
    CGDataProviderReleaseDataCallback releaseData
);
```

Parameters

info

A pointer to data of any type, or NULL. When Quartz calls the function specified in the `releaseData` parameter, Quartz sends it this pointer as its first argument.

data

A pointer to the array of data that the provider contains.

size

A value that specifies the number of bytes that the data provider contains.

releaseData

A pointer to a release callback for the data provider, or `NULL`. Your release function is called when Quartz frees the data provider. For more information, see [CGDataProviderReleaseDataCallback](#) (page 161).

Return Value

A new data provider. You are responsible for releasing this object using [CGDataProviderRelease](#) (page 155).

Discussion

You use this function to create a direct-access data provider that uses callback functions to read data from your program an entire block at one time.

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderCreateWithFilename

Creates a Quartz direct-access data provider that uses a file to supply data.

```
CGDataProviderRef CGDataProviderCreateWithFilename(
    const char *filename
);
```

Parameters*filename*

The full or relative pathname to use for the data provider. When you supply Quartz data via the provider, it reads the data from the specified file.

Return Value

A new data provider or `NULL` if the file could not be opened. You are responsible for releasing this object using [CGDataProviderRelease](#) (page 155).

Discussion

You use this function to create a direct-access data provider that supplies data from a file. When you supply Quartz with a direct-access data provider, Quartz obtains data from your program in a single block.

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderCreateWithURL

Creates a Quartz direct-access data provider that uses a URL to supply data.

```
CGDataProviderRef CGDataProviderCreateWithURL (
    CFURLRef url
);
```

Parameters

url

A CFURL object to use for the data provider. When you supply Quartz data via the provider, it reads the data from the URL address.

Return Value

A new data provider or NULL if the data from the URL could not be accessed. You are responsible for releasing this object using [CGDataProviderRelease](#) (page 155).

Discussion

You use this function to create a direct-access data provider that supplies data from a URL. When you supply Quartz with a direct-access data provider, Quartz obtains data from your program in a single entire block.

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

CGDataProviderGetTypeID

Returns the Core Foundation type identifier for Quartz data providers.

```
CFTypeID CGDataProviderGetTypeID (
    void
);
```

Return Value

The identifier for the opaque type [CGDataProviderRef](#) (page 164).

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

CGDataProviderRelease

Decrements the retain count of a data provider.

```
void CGDataProviderRelease (
    CGDataProviderRef provider
);
```

Parameters

provider

The data provider to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `provider` parameter is NULL.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGDataProvider.h

CGDataProviderRetain

Increments the retain count of a data provider.

```
CGDataProviderRef CGDataProviderRetain (  
    CGDataProviderRef provider  
);
```

Parameters

provider

The data provider to retain.

Return Value

The same data provider you passed in as the *provider* parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the *provider* parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

Callbacks by Task

Sequential-Access Data Provider Callbacks

[CGDataProviderGetBytesCallback](#) (page 159)

A callback function that copies from a provider data stream into a Quartz-supplied buffer.

[CGDataProviderReleaseInfoCallback](#) (page 162)

A callback function that releases any private data or resources associated with the data provider.

[CGDataProviderRewindCallback](#) (page 162)

A callback function that moves the current position in the data stream back to the beginning.

[CGDataProviderSkipBytesCallback](#) (page 163)

A callback function that advances the current position in the data stream supplied by the provider.

[CGDataProviderSkipForwardCallback](#) (page 164)

A callback function that advances the current position in the data stream supplied by the provider.

Direct-Access Data Provider Callbacks

[CGDataProviderGetBytePointerCallback](#) (page 157)

A callback function that returns a generic pointer to the provider data.

[CGDataProviderGetBytesAtOffsetCallback](#) (page 158)

A callback function that copies data from the provider into a Quartz buffer.

[CGDataProviderReleaseBytePointerCallback](#) (page 160)

A callback function that releases the pointer Quartz obtained by calling [CGDataProviderGetBytePointerCallback](#) (page 157).

[CGDataProviderReleaseDataCallback](#) (page 161)

A callback function that releases data you supply to the function [CGDataProviderCreateWithData](#) (page 153).

[CGDataProviderGetBytesAtPositionCallback](#) (page 159)

A callback function that copies data from the provider into a Quartz buffer.

Callbacks

CGDataProviderGetBytePointerCallback

A callback function that returns a generic pointer to the provider data.

```
const void * (*CGDataProviderGetBytePointerCallback) (
    void *info
);
```

If you name your function `MyProviderGetBytePointer`, you would declare it like this:

```
void *MyProviderGetBytePointer (
    void *info
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to [CGDataProviderCreateDirectAccess](#).

Return Value

A generic pointer to your provider data. By supplying this pointer, you are giving Quartz read-only access to both the pointer and the underlying provider data. You must not move or modify the provider data until Quartz calls your [CGDataProviderReleaseBytePointerCallback](#) (page 160) function.

Discussion

When Quartz needs direct access to your provider data, this function is called.

For information on how to associate your function with a direct-access data provider, see [CGDataProviderCreateDirectAccess](#) and [CGDataProviderDirectAccessCallbacks](#) (page 165).

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

CGDataProviderGetBytesAtOffsetCallback

A callback function that copies data from the provider into a Quartz buffer.

```
typedef size_t (*CGDataProviderGetBytesAtOffsetCallback) (  
    void *info,  
    void *buffer,  
    size_t offset,  
    size_t count  
);
```

If you name your function `MyProviderGetBytesWithOffset`, you would declare it like this:

```
size_t MyProviderGetBytesWithOffset (  
    void *info,  
    void *buffer,  
    size_t offset,  
    size_t count  
);
```

Parameters*info*

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreateDirectAccess`.

buffer

The Quartz-supplied buffer into which you copy the specified number of bytes.

offset

Specifies the relative location in the data provider at which to begin copying data.

count

The number of bytes to copy.

Return Value

The number of bytes copied. If no more data can be written to the buffer, you should return 0.

Discussion

When Quartz is ready to receive data from the provider, your function is called.

For information on how to associate your function with a direct-access data provider, see `CGDataProviderCreateDirectAccess` and [CGDataProviderDirectAccessCallbacks](#) (page 165).

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

CGDataProviderGetBytesAtPositionCallback

A callback function that copies data from the provider into a Quartz buffer.

```
typedef size_t (*CGDataProviderGetBytesAtPositionCallback) (  
    void *info,  
    void *buffer,  
    off_t position,  
    size_t count  
);
```

If you name your function `MyProviderGetBytesAtPosition`, you would declare it like this:

```
size_t MyProviderGetBytesAtPosition (  
    void *info,  
    void *buffer,  
    off_t position,  
    size_t count  
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to [CGDataProviderCreateDirect](#) (page 152).

buffer

The Quartz-supplied buffer into which you copy the specified number of bytes.

position

Specifies the relative location in the data provider at which to begin copying data.

count

The number of bytes to copy.

Return Value

The number of bytes copied. If no more data can be written to the buffer, you should return 0.

Discussion

When Quartz is ready to receive data from the provider, your function is called.

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderGetBytesCallback

A callback function that copies from a provider data stream into a Quartz-supplied buffer.

```
size_t (*CGDataProviderGetBytesCallback) (  
    void *info,  
    void *buffer,  
    size_t count  
);
```

If you name your function `MyProviderGetBytes`, you would declare it like this:

```
size_t MyProviderGetBytes (
    void *info,
    void *buffer,
    size_t count
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate`.

buffer

The Quartz-supplied buffer into which you copy the specified number of bytes.

count

The number of bytes to copy.

Return Value

The number of bytes copied. If no more data can be written to the buffer, you should return 0.

Discussion

When Quartz is ready to receive data from the provider data stream, your function is called. It should copy the specified number of bytes into *buffer*.

For information on how to associate your callback function with a data provider, see `CGDataProviderCreate` and [CGDataProviderCallbacks](#) (page 165).

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderReleaseBytePointerCallback

A callback function that releases the pointer Quartz obtained by calling [CGDataProviderGetBytePointerCallback](#) (page 157).

```
typedef void (*CGDataProviderReleaseBytePointerCallback) (
    void *info,
    const void *pointer
);
```

If you name your function `MyProviderReleaseBytePointer`, you would declare it like this:

```
void MyProviderReleaseBytePointer (
    void *info,
    const void *pointer
);
```


Parameters*info*

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreateDirectAccess`.

pointer

A pointer to your provider data. This is the same pointer you returned in [CGDataProviderGetBytePointerCallback](#) (page 157).

Discussion

When Quartz no longer needs direct access to your provider data, your function is called. You may safely modify, move, or release your provider data at this time.

For information on how to associate your function with a direct-access data provider, see `CGDataProviderCreateDirectAccess` and [CGDataProviderDirectAccessCallbacks](#) (page 165).

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderReleaseDataCallback

A callback function that releases data you supply to the function [CGDataProviderCreateWithData](#) (page 153).

```
typedef void (*CGDataProviderReleaseDataCallback) (
    void *info,
    const void *data,
    size_t size
);
```

If you name your function `MyProviderReleaseData`, you would declare it like this:

```
void MyProviderReleaseData (
    void *info,
    const void *data,
    size_t size
);
```

Parameters*info*

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to [CGDataProviderCreateWithData](#) (page 153).

data

A pointer to your provider data.

size

The size of the data.

Discussion

When Quartz no longer needs direct access to your provider data, your function is called. You may safely modify, move, or release your provider data at this time.

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

CGDataProviderReleaseInfoCallback

A callback function that releases any private data or resources associated with the data provider.

```
void (*CGDataProviderReleaseInfoCallback) (  
    void *info  
);
```

If you name your function `MyProviderReleaseInfo`, you would declare it like this:

```
void MyProviderReleaseInfo (  
    void *info  
);
```

Parameters

info

A generic pointer to private information shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate`.

Discussion

When Quartz frees a data provider that has an associated release function, the release function is called.

For information on how to associate your callback function with a data provider, see `CGDataProviderCreate` and [CGDataProviderCallbacks](#) (page 165).

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

CGDataProviderRewindCallback

A callback function that moves the current position in the data stream back to the beginning.

```
void (*CGDataProviderRewindCallback) (  
    void *info  
);
```

If you name your function `MyProviderRewind`, you would declare it like this:

```
void MyProviderRewind (  
    void *info  
);
```

```
void *info  
);
```

Parameters*info*

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate`.

Discussion

When Quartz needs to read from the beginning of the provider's data stream, your function is called.

For information on how to associate your callback function with a data provider, see `CGDataProviderCreate` and [CGDataProviderCallbacks](#) (page 165).

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderSkipBytesCallback

A callback function that advances the current position in the data stream supplied by the provider.

```
void (*CGDataProviderSkipBytesCallback) (  
    void *info,  
    size_t count  
);
```

If you name your function `MyProviderSkipBytes`, you would declare it like this:

```
void MyProviderSkipBytes (  
    void *info,  
    size_t count  
);
```

Parameters*info*

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate`.

count

The number of bytes to skip.

Discussion

When Quartz needs to advance forward in the provider's data stream, your function is called.

For information on how to associate your callback function with a data provider, see `CGDataProviderCreate` and [CGDataProviderCallbacks](#) (page 165).

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

CGDataProviderSkipForwardCallback

A callback function that advances the current position in the data stream supplied by the provider.

```
off_t (*CGDataProviderSkipForwardCallback) (
    void *info,
    off_t count
);
```

If you name your function `MyProviderSkipForwardBytes`, you would declare it like this:

```
off_t MyProviderSkipForwardBytes (
    void *info,
    off_t count
);
```

Parameters*info*

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate`.

count

The number of bytes to skip.

Return Value

The number of bytes that were actually skipped.

Discussion

When Quartz needs to advance forward in the provider's data stream, your function is called.

Availability

Available in iOS 2.0 and later.

Declared In

CGDataProvider.h

Data Types

CGDataProviderRef

Defines an opaque type that supplies Quartz with data.

```
typedef struct CGDataProvider *CGDataProviderRef;
```

Discussion

Some Quartz routines supply blocks of data to your program. Rather than reading through a raw memory buffer, data provider objects of type `CGDataProviderRef` allow you to supply Quartz functions with data.

In Mac OS X version 10.2 and later, `CGDataProviderRef` is derived from `CTypeRef` and inherits the properties that all Core Foundation types have in common. For more information, see *CType Reference*.

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderCallbacks

Defines a structure containing pointers to client-defined callback functions that manage the sending of data for a sequential-access data provider.

```
struct CGDataProviderCallbacks {
    CGDataProviderGetBytesCallback getBytes;
    CGDataProviderSkipBytesCallback skipBytes;
    CGDataProviderRewindCallback rewind;
    CGDataProviderReleaseInfoCallback releaseProvider;
};
typedef struct CGDataProviderCallbacks CGDataProviderCallbacks;
```

Fields

`getBytes`

A pointer to a function that copies data from the provider. For more information, see [CGDataProviderGetBytesCallback](#) (page 159).

`skipBytes`

A pointer to a function that Quartz calls to advance the stream of data supplied by the provider. For more information, see [CGDataProviderSkipBytesCallback](#) (page 163).

`rewind`

A pointer to a function Quartz calls to return the provider to the beginning of the data stream. For more information, see [CGDataProviderRewindCallback](#) (page 162).

`releaseProvider`

A pointer to a function that handles clean-up for the data provider, or `NULL`. For more information, see [CGDataProviderReleaseInfoCallback](#) (page 162).

Discussion

The functions specified by the `CGDataProviderCallbacks` structure are responsible for sequentially copying data to a memory buffer for Quartz to use. The functions are also responsible for handling the data provider's basic memory management. You supply a `CGDataProviderCallbacks` structure to the function `CGDataProviderCreate` to create a sequential-access data provider.

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderDirectAccessCallbacks

Defines pointers to client-defined callback functions that manage the sending of data for a direct-access data provider.

```

struct CGDataProviderDirectAccessCallbacks {
    CGDataProviderGetBytePointerCallback getBytePointer;
    CGDataProviderReleaseBytePointerCallback releaseBytePointer;
    CGDataProviderGetBytesAtOffsetCallback getBytes;
    CGDataProviderReleaseInfoCallback releaseProvider;
};
typedef struct CGDataProviderDirectAccessCallbacks
CGDataProviderDirectAccessCallbacks;

```

Fields**getBytePointer**

A pointer to a function that returns a pointer to the provider's data. For more information, see [CGDataProviderGetBytePointerCallback](#) (page 157).

releaseBytePointer

A pointer to a function that Quartz calls to release a pointer to the provider's data. For more information, see [CGDataProviderReleaseBytePointerCallback](#) (page 160).

getBytes

A pointer to a function that copies data from the provider. For more information, see [CGDataProviderGetBytesAtOffsetCallback](#) (page 158).

releaseProvider

A pointer to a function that handles clean-up for the data provider, or NULL. For more information, see [CGDataProviderReleaseInfoCallback](#) (page 162).

Discussion

You supply a `CGDataProviderDirectAccessCallbacks` structure to the function `CGDataProviderCreateDirectAccess` to create a data provider for direct access. The functions specified by the `CGDataProviderDirectAccessCallbacks` structure are responsible for copying data a block at a time to a memory buffer for Quartz to use. The functions are also responsible for handling the data provider's basic memory management. For the callback to work, one of the `getBytePointer` and `getBytes` parameters must be non-NULL. If both are non-NULL, then `getBytePointer` is used to access the data.

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderDirectCallbacks

Defines pointers to client-defined callback functions that manage the sending of data for a direct-access data provider.

```

struct CGDataProviderDirectCallbacks {
    unsigned int version;
    CGDataProviderGetBytePointerCallback getBytePointer;
    CGDataProviderReleaseBytePointerCallback releaseBytePointer;
    CGDataProviderGetBytesAtPositionCallback getBytesAtPosition;
    CGDataProviderReleaseInfoCallback releaseInfo;
};
typedef struct CGDataProviderDirectCallbacks CGDataProviderDirectCallbacks;

```

Fields**version**

The version of this structure. It should be set to 0.

getBytesPointer

A pointer to a function that returns a pointer to the provider's data. For more information, see [CGDataProviderGetBytesPointerCallback](#) (page 157).

releaseBytesPointer

A pointer to a function that Quartz calls to release a pointer to the provider's data. For more information, see [CGDataProviderReleaseBytesPointerCallback](#) (page 160).

getBytesAtPosition

A pointer to a function that copies data from the provider.

releaseInfo

A pointer to a function that handles clean-up for the data provider, or NULL. For more information, see [CGDataProviderReleaseInfoCallback](#) (page 162).

Discussion

You supply a `CGDataProviderDirectCallbacks` structure to the function [CGDataProviderCreateDirect](#) (page 152) to create a data provider for direct access. The functions specified by the `CGDataProviderDirectCallbacks` structure are responsible for copying data a block at a time to a memory buffer for Quartz to use. The functions are also responsible for handling the data provider's basic memory management. For the callback to work, one of the `getBytesPointer` and `getBytesAtPosition` parameters must be non-NULL. If both are non-NULL, then `getBytesPointer` is used to access the data.

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderSequentialCallbacks

Defines a structure containing pointers to client-defined callback functions that manage the sending of data for a sequential-access data provider.

```
struct CGDataProviderSequentialCallbacks {
    unsigned int version;
    CGDataProviderGetBytesCallback getBytes;
    CGDataProviderSkipForwardCallback skipForward;
    CGDataProviderRewindCallback rewind;
    CGDataProviderReleaseInfoCallback releaseInfo;
};
typedef struct CGDataProviderSequentialCallbacks CGDataProviderSequentialCallbacks;
```

Fields**version**

The version of this structure. It should be set to 0.

getBytes

A pointer to a function that copies data from the provider. For more information, see [CGDataProviderGetBytesCallback](#) (page 159).

skipForward

A pointer to a function that Quartz calls to advance the stream of data supplied by the provider.

rewind

A pointer to a function Quartz calls to return the provider to the beginning of the data stream. For more information, see [CGDataProviderRewindCallback](#) (page 162).

`releaseInfo`

A pointer to a function that handles clean-up for the data provider, or `NULL`. For more information, see [CGDataProviderReleaseInfoCallback](#) (page 162).

Discussion

The functions specified by the `CGDataProviderSequentialCallbacks` structure are responsible for sequentially copying data to a memory buffer for Quartz to use. The functions are also responsible for handling the data provider's basic memory management. You supply a `CGDataProviderCallbacks` structure to the function [CGDataProviderCreateSequential](#) (page 152) to create a sequential-access data provider.

Availability

Available in iOS 2.0 and later.

Declared In

`CGDataProvider.h`

CGFont Reference

Derived From:	<i>CType Reference</i>
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGFont.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGFontRef` opaque type encapsulates font information. A font is a set of shapes or glyphs associated with a character set. A glyph can represent a single character (such as 'b'), more than one character (such as the "fi" ligature), or a special character such as a space. Quartz retrieves the glyphs for the font from ATS (Apple Type Services) and paints the glyphs based on the relevant parameters of the current graphics state.

Quartz provides a limited, low-level interface for drawing text. For information on text-drawing functions, see *CGContext Reference*. For full Unicode and text-layout support, use the services provided by Core Text or ATSUI).

Functions by Task

Retaining and Releasing a CGFont Object

[CGFontRelease](#) (page 184)

Decrements the retain count of a Quartz font.

[CGFontRetain](#) (page 184)

Increments the retain count of a Quartz font.

Creating a CGFont Object

[CGFontCreateWithDataProvider](#) (page 176)

Creates a font object from data supplied from a data provider.

[CGFontCreateWithFontName](#) (page 177)

Creates a font object corresponding to the font specified by a PostScript or full name.

[CGFontCreateCopyWithVariations](#) (page 175)

Creates a copy of a font using a variation specification dictionary.

Working With PostScript Fonts

[CGFontCopyPostScriptName](#) (page 172)

Obtains the PostScript name of a font.

[CGFontCanCreatePostScriptSubset](#) (page 171)

Determines whether Quartz can create a subset of the font in PostScript format.

[CGFontCreatePostScriptSubset](#) (page 176)

Creates a subset of the font in the specified PostScript format.

[CGFontCreatePostScriptEncoding](#) (page 175)

Creates a PostScript encoding of a font.

Working With Font Tables

[CGFontCopyTableTags](#) (page 173)

Returns an array of tags that correspond to the font tables for a font.

[CGFontCopyTableForTag](#) (page 173)

Returns the font table that corresponds to the provided tag.

Getting Font Information

[CGFontGetTypeID](#) (page 183)

Returns the Core Foundation type identifier for Quartz fonts.

[CGFontCopyVariationAxes](#) (page 174)

Returns an array of the variation axis dictionaries for a font.

[CGFontCopyVariations](#) (page 174)

Returns the variation specification dictionary for a font.

[CGFontCopyFullName](#) (page 171)

Returns the full name associated with a font object.

[CGFontGetAscent](#) (page 177)

Returns the ascent of a font.

[CGFontGetDescent](#) (page 178)

Returns the descent of a font.

[CGFontGetLeading](#) (page 181)

Returns the leading of a font.

[CGFontGetCapHeight](#) (page 178)

Returns the cap height of a font.

[CGFontGetXHeight](#) (page 183)

Returns the x-height of a font.

[CGFontGetFontBBox](#) (page 179)

Returns the bounding box of a font.

[CGFontGetItalicAngle](#) (page 181)

Returns the italic angle of a font.

[CGFontGetStemV](#) (page 182)

Returns the thickness of the dominant vertical stems of glyphs in a font.

[CGFontGetGlyphBBoxes](#) (page 180)

Get the bounding box of each glyph in an array.

[CGFontGetGlyphWithGlyphName](#) (page 181)

Returns the glyph for the glyph name associated with the specified font object.

[CGFontCopyGlyphNameForGlyph](#) (page 172)

Returns the glyph name of the specified glyph in the specified font.

[CGFontGetNumberOfGlyphs](#) (page 182)

Returns the number of glyphs in a font.

[CGFontGetGlyphAdvances](#) (page 179)

Gets the bound box of each glyph in the provided array.

[CGFontGetUnitsPerEm](#) (page 183)

Returns the number of glyph space units per em for the provided font.

Functions

CGFontCanCreatePostScriptSubset

Determines whether Quartz can create a subset of the font in PostScript format.

```
bool CGFontCanCreatePostScriptSubset (
    CGFontRef font,
    CGFontPostScriptFormat format
);
```

Parameters

font

A font object.

Return Value

Returns `true` if a subset in the PostScript format can be created for the font; `false` otherwise.

Discussion

For more information on PostScript format, see *Adobe Type 1 Font Format*, which is available from <http://partners.adobe.com/>.

Availability

Available in iOS 2.0 and later.

Declared In

`CGFont.h`

CGFontCopyFullName

Returns the full name associated with a font object.

```
CFStringRef CGFontCopyFullName (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value

The full name associated with the font.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCopyGlyphNameForGlyph

Returns the glyph name of the specified glyph in the specified font.

```
CFStringRef CGFontCopyGlyphNameForGlyph (
    CGFontRef font
    CGGlyph glyph
);
```

Parameters

font

A font object.

glyph

The glyph whose name is desired.

Return Value

The name of the specified glyph, or `NULL` if the glyph isn't associated with the font object.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCopyPostScriptName

Obtains the PostScript name of a font.

```
CFStringRef CGFontCopyPostScriptName (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value

The PostScript name of the font.

Discussion**Availability**

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCopyTableForTag

Returns the font table that corresponds to the provided tag.

```
CFDataRef CGFontCopyTableForTag(  
    CGFontRef font,  
    uint32_t tag  
);
```

Parameters

font

A font object.

tag

The tag for the table you want to obtain.

Return Value

The font table that corresponds to the tag, or NULL if no such table exists.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCopyTableTags

Returns an array of tags that correspond to the font tables for a font.

```
CFArrayRef CGFontCopyTableTags(  
    CGFontRef font  
);
```

Parameters

font

A CGFont object.

Return Value

An array of font table tags.

Discussion

Each entry in the returned array is a four-byte value that represents a single TrueType or OpenType font table tag. To obtain a tag at index *k* in a manner that is appropriate for 32-bit and 64-bit architectures, you need to use code similar to the following:

```
tag = (uint32_t)(uintptr_t)CFArrayGetValue(table, k);
```

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCopyVariationAxes

Returns an array of the variation axis dictionaries for a font.

```
CFArrayRef CGFontCopyVariationAxes (
    CGFontRef font
);
```

Parameters

font

A CGFont object.

Return Value

An array of the variation axis dictionaries. Returns `NULL` if the font doesn't support variations.

Discussion

A variation axis is a range included in a font by the font designer that allows a font to produce different type styles. Each variation axis dictionary contains key-value pairs that specify the variation axis name and the minimum, maximum, and default values for that variation axis.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCopyVariations

Returns the variation specification dictionary for a font.

```
CFDictionaryRef CGFontCopyVariations (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value

The variation specification dictionary for the font. Returns `NULL` if the font doesn't support variations.

Discussion

The variation specification dictionary contains keys that correspond to the variation axis names of the font. Each key is a variation axis name. The value for each key is the value specified for that particular variation axis represented as a `CFNumber` object.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCreateCopyWithVariations

Creates a copy of a font using a variation specification dictionary.

```
CGFontRef CGFontCreateCopyWithVariations (
    CGFontRef font,
    CFDictionaryRef variations
);
```

Parameters*font*

The Quartz font to copy.

variations

A variation specification dictionary that contains keys corresponding to the variation axis names of the font. Each key in the dictionary is a variation axis name. The value for each key is the value specified for that particular variation axis represented as a CFNumber object. If a variation axis name is not specified in *variations*, then the current value from *font* is used.

Return Value

The font object.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCreatePostScriptEncoding

Creates a PostScript encoding of a font.

```
CFDataRef CGFontCreatePostScriptEncoding (
    CGFontRef font,
    const CGGlyph encoding[256]
);
```

Parameters*font*

A CGFont object.

encoding

The encoding to use.

Return Value

A PostScript encoding of the font that contains glyphs in the specified encoding.

Discussion

For more information on PostScript format, see *Adobe Type 1 Font Format*, which is available from <http://partners.adobe.com/>.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCreatePostScriptSubset

Creates a subset of the font in the specified PostScript format.

```
CFDataRef CGFontCreatePostScriptSubset (
    CGFontRef font,
    CFStringRef subsetName,
    CGFontPostScriptFormat format,
    const CGGlyph glyphs[],
    size_t count,
    const CGGlyph encoding[256]
);
```

Parameters*font*

A font object.

subsetName

The name of the subset.

format

The PostScript format of the font.

glyphs

An array that contains the glyphs in the subset.

count

The number of glyphs specified by the *glyphs* array.

encoding

The default encoding for the subset. You can pass `NULL` if you do not want to specify an encoding.

Return Value

A subset of the font created from the supplied parameters.

Discussion

For more information on PostScript format, see *Adobe Type 1 Font Format*, which is available from <http://partners.adobe.com/>.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontCreateWithDataProvider

Creates a font object from data supplied from a data provider.


```
CGFontRef CGFontCreateWithDataProvider (
    CGDataProviderRef provider
);
```

Parameters

provider

A data provider.

Return Value

The font object or `NULL` if the font can't be created. You are responsible for releasing this object using [CGFontRelease](#) (page 184).

Discussion

Before drawing text in a Quartz context, you must set the font in the current graphics state by calling the function [CGContextSetFontSize](#) (page 108).

Availability

Available in iOS 2.0 and later.

Declared In

`CGFont.h`

CGFontCreateWithFontName

Creates a font object corresponding to the font specified by a PostScript or full name.

```
CGFontRef CGFontCreateWithFontName (
    CFStringRef name
);
```

Parameters

name

The PostScript or full name of a font.

Return Value

The font object or `NULL` if the font can't be created. You are responsible for releasing this object using [CGFontRelease](#) (page 184).

Discussion

Before drawing text in a Quartz context, you must set the font in the current graphics state by calling the function [CGContextSetFont](#) (page 107).

Availability

Available in iOS 2.0 and later.

Related Sample Code

`QuartzDemo`

Declared In

`CGFont.h`

CGFontGetAscent

Returns the ascent of a font.

```
int CGFontGetAscent (  
    CGFontRef font  
);
```

Parameters

font

A font object.

Return Value

The ascent of the font.

Discussion

The ascent is the maximum distance above the baseline of glyphs in a font. The value is specified in glyph space units.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetCapHeight

Returns the cap height of a font.

```
int CGFontGetCapHeight (  
    CGFontRef font  
);
```

Parameters

font

A font object.

Return Value

The cap height of the font.

Discussion

The cap height is the distance above the baseline of the top of flat capital letters of glyphs in a font. The value is specified in glyph space units.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetDescent

Returns the descent of a font.

```
int CGFontGetDescent (  
    CGFontRef font  
);
```

Parameters

font

A font object.

Return Value

The descent of the font .

Discussion

The descent is the maximum distance below the baseline of glyphs in a font. The value is specified in glyph space units.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetFontBBox

Returns the bounding box of a font.

```
CRect CGFontGetFontBBox (  
    CGFontRef font  
);
```

Parameters

font

A font object.

Return Value

The bounding box of the font.

Discussion

The font bounding box is the union of all of the bounding boxes for all the glyphs in a font. The value is specified in glyph space units.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetGlyphAdvances

Gets the bound box of each glyph in the provided array.

```
bool CGFontGetGlyphAdvances (
    CGFontRef font,
    const CGGlyph glyphs[],
    size_t count,
    int advances[]
);
```

Parameters*font*

The font object associated with the provided glyphs.

glyphs

An array of glyphs.

count

The number of glyphs in the array.

advances

On output, an array of advances for the provided glyphs.

Return Value

TRUE unless the advances can't be provided for some reason.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetGlyphBBoxes

Get the bounding box of each glyph in an array.

```
bool CGFontGetGlyphBBoxes (
    CGFontRef font,
    const CGGlyph glyphs[],
    size_t count,
    CGRect bboxes[]
);
```

Parameters*font*

A font object.

glyphs

A array of glyphs.

count

The number of items in the glyphs array.

bboxes

On return, the bounding boxes for each glyph.

Return Value

false if bounding boxes can't be retrieved for any reason; true otherwise.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetGlyphWithGlyphName

Returns the glyph for the glyph name associated with the specified font object.

```
CGGlyph CGFontGetGlyphWithGlyphName (  
    CGFontRef font  
    CFStringRef name  
);
```

Parameters*font*

A font object.

name

The name of the desired glyph.

Return Value

The named glyph, or 0 if the named glyph isn't associated with the font.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetItalicAngle

Returns the italic angle of a font.

```
CGFloat CGFontGetItalicAngle (  
    CGFontRef font  
);
```

Parameters*font*

A font object.

Return Value

The italic angle of the font, measured in degrees counter-clockwise from the vertical.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetLeading

Returns the leading of a font.

```
int CGFontGetLeading (
    CGFontRef font
);
```

Parameters*font*

A font object.

Return Value

The leading of the font.

Discussion

The leading is the spacing between consecutive lines of text in a font. The value is specified in glyph space units.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetNumberOfGlyphs

Returns the number of glyphs in a font.

```
size_t CGFontGetNumberOfGlyphs (
    CGFontRef font
);
```

Parameters*font*

A CGFont object.

Return Value

The number of glyphs in the provided font.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetStemV

Returns the thickness of the dominant vertical stems of glyphs in a font.

```
CGFloat CGFontGetItalicAngle (
    CGFontRef font
);
```

Parameters*font*

A font object.

Return Value

The thickness of the dominant vertical stems of glyphs in a font.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetTypeID

Returns the Core Foundation type identifier for Quartz fonts.

```
CTypeID CGFontGetTypeID (  
    void  
);
```

Return Value

The Core Foundation identifier for the opaque type [CGFontRef](#) (page 185).

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetUnitsPerEm

Returns the number of glyph space units per em for the provided font.

```
int CGFontGetUnitsPerEm (  
    CGFontRef font  
);
```

Parameters

font

A `CGFont` object.

Return Value

The number of glyph space units per em for the provided font.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontGetXHeight

Returns the x-height of a font.

```
int CGFontGetXHeight (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value

The x-height of the font.

Discussion

The x-height is the distance above the baseline of the top of flat, non-ascending lowercase letters (such as x) of glyphs in a font. The value is specified in glyph space units.

Availability

Available in iOS 2.0 and later.

Declared In

CGFont.h

CGFontRelease

Decrements the retain count of a Quartz font.

```
void CGFontRelease (
    CGFontRef font
);
```

Parameters

font

The Quartz font to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `font` parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

CGFont.h

CGFontRetain

Increments the retain count of a Quartz font.


```
CGFontRef CGFontRetain (  
    CGFontRef font  
);
```

Parameters

font

The Quartz font to retain.

Return Value

The same font you specified in the `font` parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `font` parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Declared In

`CGFont.h`

Data Types

CGFontRef

An opaque type that encapsulates font information.

```
typedef struct CGFont *CGFontRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

`CGFont.h`

CGFontIndex

An index into a font table.

```
typedef unsigned short CGFontIndex;
```

Discussion

This integer type provides an additional way to specify a glyph identifier. `CGFontIndex` is equivalent to [CGGlyph](#) (page 186), and you can use constants of either type interchangeably.

Availability**Declared In**

`CGFont.h`

CGGlyph

An index into the internal glyph table of a font.

```
typedef unsigned short CGGlyph;
```

Discussion

When drawing text, you typically specify a sequence of characters. However, Quartz also allows you to use `CGGlyph` values to specify glyphs. In either case, Quartz renders the text using font data provided by the Apple Type Services (ATS) framework.

You provide `CGGlyph` values to the functions `CGContextShowGlyphs` (page 124) and `CGContextShowGlyphsAtPoint` (page 124). These functions display an array of glyphs at the current text position or at a position you specify, respectively.

Availability

Available in iOS 2.0 and later.

Declared In

`CGFont.h`

Constants

CGFontPostScriptFormat

Possible formats for a PostScript font subset.

```
enum CGFontPostScriptFormat {
    kCGFontPostScriptFormatType1 = 1,
    kCGFontPostScriptFormatType3 = 3,
    kCGFontPostScriptFormatType42 = 42
};
typedef enum CGFontPostScriptFormat CGFontPostScriptFormat;
```

Constants

`kCGFontPostScriptFormatType1`

This is documented in *Adobe Type 1 Font Format*, which is available from <http://partners.adobe.com/>.

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

`kCGFontPostScriptFormatType3`

This is documented in *PostScript Language Reference, 3rd edition*, which is available from <http://partners.adobe.com/>.

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

`kCGFontPostScriptFormatType42`

This is documented in *Adobe Technical Note 5012, The Type 42 Font Format Specification*, which is available from <http://partners.adobe.com/>.

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

Font Table Index Values

Possible values for an index into a font table.

```
enum {
    kCGFontIndexMax = ((1 << 16) - 2),
    kCGFontIndexInvalid = ((1 << 16) - 1),
    kCGGlyphMax = kCGFontIndexMax
};
```

Constants

`kCGFontIndexMax`

The maximum allowed value for [CGFontIndex](#) (page 185).

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

`kCGFontIndexInvalid`

An invalid font index (a value which never represents a valid glyph).

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

`kCGGlyphMax`

The same as [kCGFontIndexMax](#) (page 187).

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

Discussion

See [CGFontIndex](#) (page 185).

Obsolete Font Table Index Values

Deprecated values for an index into a font table. (**Deprecated.** Use [kCGGlyphMax](#) (page 187) instead.)

```
enum {
    CGGlyphMin = 0,
    CGGlyphMax = kCGGlyphMax
};
```

Constants

`CGGlyphMin`

Minimum font index value.

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

`CGGlyphMax`

Maximum font index value.

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

Font Variation Axis Keys

Keys used for a font variation axis dictionary.

```
const CFStringRef kCGFontVariationAxisName
const CFStringRef kCGFontVariationAxisMinValue
const CFStringRef kCGFontVariationAxisMaxValue
const CFStringRef kCGFontVariationAxisDefaultValue
```

Constants

`kCGFontVariationAxisName`

The key used to obtain the variation axis name from a variation axis dictionary. The value obtained with this key is a `CFStringRef` that specifies the name of the variation axis.

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

`kCGFontVariationAxisMinValue`

The key used to obtain the minimum variation axis value from a variation axis dictionary. The value obtained with this key is a `CFNumberRef` that specifies the minimum value of the variation axis.

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

`kCGFontVariationAxisMaxValue`

The key used to obtain the maximum variation axis value from a variation axis dictionary. The value obtained with this key is a `CFNumberRef` that specifies the maximum value of the variation axis.

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

`kCGFontVariationAxisDefaultValue`

The key used to obtain the default variation axis value from a variation axis dictionary. The value obtained with this key is a `CFNumberRef` that specifies the default value of the variation axis.

Available in iOS 2.0 and later.

Declared in `CGFont.h`.

CGFunction Reference

Derived From:	CType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGFunction.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGFunctionRef` opaque type provides a general facility for defining and using callback functions. These functions can take an arbitrary number of floating-point input values and pass back an arbitrary number of floating-point output values.

Quartz uses CGFunction objects to implement shadings. *CGShading Reference* describes the parameters and semantics required for the callbacks used by CGFunction objects.

Functions by Task

Creating a CGFunction Object

[CGFunctionCreate](#) (page 190)
Creates a Quartz function.

Retaining and Releasing CGFunction Objects

[CGFunctionRelease](#) (page 191)
Decrements the retain count of a function object.

[CGFunctionRetain](#) (page 191)
Increments the retain count of a function object.

Getting the CType ID

[CGFunctionGetTypeID](#) (page 191)
Returns the type identifier for Quartz function objects.

Functions

CGFunctionCreate

Creates a Quartz function.

```
CGFunctionRef CGFunctionCreate (
    void *info,
    size_t domainDimension,
    const CGFloat *domain,
    size_t rangeDimension,
    const CGFloat *range,
    const CGFunctionCallbacks *callbacks
);
```

Parameters

info

A pointer to user-defined storage for data that you want to pass to your callbacks. You need to make sure that the data persists for as long as it's needed, which can be beyond the scope in which the Quartz function is used.

domainDimension

The number of inputs.

domain

An array of ($2 \times \text{domainDimension}$) floats used to specify the valid intervals of input values. For each k from 0 to $(\text{domainDimension} - 1)$, $\text{domain}[2 \times k]$ must be less than or equal to $\text{domain}[2 \times k + 1]$, and the k th input value will be clipped to lie in the interval $\text{domain}[2 \times k] \leq \text{input}[k] \leq \text{domain}[2 \times k + 1]$. If this parameter is NULL, then the input values are not clipped.

rangeDimension

The number of outputs.

range

An array of ($2 \times \text{rangeDimension}$) floats that specifies the valid intervals of output values. For each k from 0 to $(\text{rangeDimension} - 1)$, $\text{range}[2 \times k]$ must be less than or equal to $\text{range}[2 \times k + 1]$, and the k th output value will be clipped to lie in the interval $\text{range}[2 \times k] \leq \text{output}[k] \leq \text{range}[2 \times k + 1]$. If this parameter is NULL, then the output values are not clipped.

callbacks

A pointer to a callback function table. This table should contain pointers to the callbacks you provide to implement the semantics of this Quartz function. Quartz makes a copy of your table, so, for example, you could safely pass in a pointer to a structure on the stack.

Return Value

The new Quartz function. You are responsible for releasing this object using [CGFunctionRelease](#) (page 191).

Availability

Declared In

`CGFunction.h`

CGFunctionGetTypeID

Returns the type identifier for Quartz function objects.

```
CTypeID CGFunctionGetTypeID (  
    void  
);
```

Return Value

The identifier for the opaque type [CGFunctionRef](#) (page 193).

Availability

Available in iOS 2.0 and later.

Declared In

`CGFunction.h`

CGFunctionRelease

Decrements the retain count of a function object.

```
void CGFunctionRelease (  
    CGFunctionRef function  
);
```

Parameters

function

The function object to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `function` parameter is `NULL`.

Availability

Declared In

`CGFunction.h`

CGFunctionRetain

Increments the retain count of a function object.

```
CGFunctionRef CGFunctionRetain (  
    CGFunctionRef function  
);
```

Parameters

function

The same function object you passed in as the `function` parameter.

Return Value

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `function` parameter is `NULL`.

Availability**Declared In**

CGFunction.h

Callbacks

CGFunctionEvaluateCallback

Performs custom operations on the supplied input data to produce output data.

```
typedef void (*CGFunctionEvaluateCallback) (
    void *info,
    const float *inData,
    float *outData
);
```

If you name your function `MyCGFunctionEvaluate`, you would declare it like this:

```
void MyCGFunctionEvaluate (
    void *info,
    const float *inData,
    float *outData
);
```

Parameters*info*

The `info` parameter passed to [CGFunctionCreate](#) (page 190).

inData

An array of floats. The size of the array is that specified by the `domainDimension` parameter passed to the [CGFunctionCreate](#) (page 190) function.

outData

An array of floats. The size of the array is that specified by the `rangeDimension` parameter passed to the [CGFunctionCreate](#) (page 190) function.

Discussion

The callback you write is responsible for implementing the calculation of output values from the supplied input values. For example, if you want to implement a simple "squaring" function of one input argument to one output argument, your evaluation function might be:

```
void evaluateSquare(void *info, const float *inData, float *outData)
{
    outData[0] = inData[0] * inData[0];
}
```

Availability

Available in iOS 2.0 and later.

Declared In

CGFunction.h

CGFunctionReleaseInfoCallback

Performs custom clean-up tasks when Quartz deallocates a CGFunction object.

```
typedef void (*CGFunctionReleaseInfoCallback) (  
    void *info  
);
```

If you name your function `MyCGFunctionReleaseInfo`, you would declare it like this:

```
void MyCGFunctionReleaseInfo (  
    void *info  
);
```

Parameters

info

The `info` parameter passed to [CGFunctionCreate](#) (page 190).

Availability

Available in iOS 2.0 and later.

Declared In

`CGFunction.h`

Data Types

CGFunctionRef

An opaque type that represents a callback function.

```
typedef struct CGFunction *CGFunctionRef;
```

Availability

Declared In

`CGFunction.h`

CGFunctionCallbacks

A structure that contains callbacks needed by a CGFunction object.

```
struct CGFunctionCallbacks  
{  
    unsigned int version;  
    CGFunctionEvaluateCallback evaluate;  
    CGFunctionReleaseInfoCallback releaseInfo  
};
```

```
typedef struct CGFunctionCallbacks CGFunctionCallbacks;
```

Fields

`version`

The structure version number. For this structure, the version should be 0.

`evaluate`

The callback that evaluates the function.

`releaseInfo`

If non-NULL, the callback used to release the `info` parameter passed to [CGFunctionCreate](#) (page 190).

Availability

Available in iOS 2.0 and later.

Declared In

`CGFunction.h`

CGGradient Reference

Derived From:	CType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGGradient.h
Companion guide	Quartz 2D Programming Guide

Overview

A gradient defines a smooth transition between colors across an area. The `CGGradientRef` opaque type, and the functions that operate on it, make creating and using radial and axial gradient fills an easy task. A `CGGradient` object has a color space, two or more colors, and a location for each color. The color space cannot be a pattern or indexed color space, otherwise it can be any Quartz color space ([CGColorSpaceRef](#) (page 46)).

Colors can be provided as component values (such as red, green, blue) or as Quartz color objects ([CGColorRef](#) (page 30)). In Quartz, component can vary from 0.0 to 1.0, designating the proportion of the component present in the color.

A location is a normalized value. When it comes time to paint the gradient, Quartz maps the normalized location values to the points in coordinate space that you provide.

If you want more precise control over gradients, or if your application runs in versions of Mac OS X that are earlier than v10.5, see *CGShading Reference*.

Functions by Task

Creating a CGGradient Object

[CGGradientCreateWithColorComponents](#) (page 196)

Creates a `CGGradient` object from a color space and the provided color components and locations.

[CGGradientCreateWithColors](#) (page 197)

Creates a `CGGradient` object from a color space and the provided color objects and locations.

Retaining and Releasing a CGGradient Object

[CGGradientRelease](#) (page 198)

Decrements the retain count of a CGGradient object.

[CGGradientRetain](#) (page 199)

Increments the retain count of a CGGradient object.

Getting the Type ID for a CGGradient Object

[CGGradientGetTypeID](#) (page 198)

Returns the Core Foundation type identifier for CGGradient objects.

Functions

CGGradientCreateWithColorComponents

Creates a CGGradient object from a color space and the provided color components and locations.

```
CGGradientRef CGGradientCreateWithColorComponents(
    CGColorSpaceRef space,
    const CGFloat components[],
    const CGFloat locations[],
    size_t count
);
```

Parameters

space

The color space to use for the gradient. You cannot use a pattern or indexed color space.

components

The color components for each color that defines the gradient. The components should be in the color space specified by *space*. If you are unsure of the number of components, you can call the function [CGColorSpaceGetNumberOfComponents](#) (page 44).

The number of items in this array should be the product of *count* and the number of components in the color space. For example, if the color space is an RGBA color space and you want to use two colors in the gradient (one for a starting location and another for an ending location), then you need to provide 8 values in *components*—red, green, blue, and alpha values for the first color, followed by red, green, blue, and alpha values for the second color.

locations

The location for each color provided in *components*. Each location must be a `CGFloat` value in the range of 0 to 1, inclusive. If 0 and 1 are not in the *locations* array, Quartz uses the colors provided that are closest to 0 and 1 for those locations.

If *locations* is `NULL`, the first color in *colors* is assigned to location 0, the last color in *colors* is assigned to location 1, and intervening colors are assigned locations that are at equal intervals in between.

count

The number of locations provided in the *locations* parameters.

Return Value

A CGGradient object.

Availability

Available in iOS 2.0 and later.

See Also

[CGContextDrawLinearGradient](#) (page 78)

[CGContextDrawRadialGradient](#) (page 80)

Related Sample Code

GLImageProcessing

PVRTextureLoader

QuartzDemo

Reflection

TheElements

Declared In

CGGradient.h

CGGradientCreateWithColors

Creates a CGGradient object from a color space and the provided color objects and locations.

```
CGGradientRef CGGradientCreateWithColors(
    CGColorSpaceRef space,
    CFArrayRef colors,
    const CGFloat locations[]
);
```

Parameters

space

The color space to use for the gradient. You cannot use a pattern or indexed color space.

colors

A non-empty array of CGColor objects that should be in the color space specified by *space*. If *space* is not NULL, each color will be converted (if necessary) to that color space and the gradient will drawn in that color space. Otherwise, each color will be converted to and drawn in the GenericRGB color space.

locations

The location for each color provided in *colors*; each location must be a CGFloat value in the range of 0 to 1, inclusive. If 0 and 1 are not in the *locations* array, Quartz uses the colors provided that are closest to 0 and 1 for those locations.

If *locations* is NULL, the first color in *colors* is assigned to location 0, the last color in *colors* is assigned to location 1, and intervening colors are assigned locations that are at equal intervals in between.

The *locations* array should contain the same number of items as the *colors* array.

Return Value

A CGGradient object.

Availability

Available in iOS 2.0 and later.

See Also[CGContextDrawLinearGradient](#) (page 78)[CGContextDrawRadialGradient](#) (page 80)**Declared In**

CGGradient.h

CGGradientGetTypeID

Returns the Core Foundation type identifier for CGGradient objects.

```
CTypeID CGGradientGetTypeID (  
    void  
);
```

Return Value

The Core Foundation identifier for the opaque type CGGradientRef.

Availability**Declared In**

CGGradient.h

CGGradientRelease

Decrements the retain count of a CGGradient object.

```
void CGGradientRelease (  
    CGGradientRef gradient  
);
```

Parameters*gradient*

The gradient object to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the *gradient* parameter is `NULL`.

Availability**Related Sample Code**

GLImageProcessing

PVRTextureLoader

QuartzDemo

Reflection

TheElements

Declared In

CGGradient.h

CGGradientRetain

Increments the retain count of a CGGradient object.

```
CGGradientRef CGGradientRetain(  
    CGGradientRef gradient  
);
```

Parameters

gradient

The gradient object to retain.

Return Value

The same gradient object that you passed in as the *gradient* parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the *gradient* parameter is `NULL`.

Availability

Declared In

`CGGradient.h`

Data Types

CGGradientRef

An opaque type that represents a Quartz gradient.

```
typedef struct CGGradient *CGGradientRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

`CGGradient.h`

Constants

Gradient Drawing Options

Drawing locations for gradients.

```
enum {  
    kCGGradientDrawsBeforeStartLocation = (1 << 0),  
    kCGGradientDrawsAfterEndLocation = (1 << 1)  
};  
typedef enum CGGradientDrawingOptions CGGradientDrawingOptions;
```

Constants

`kCGGradientDrawsBeforeStartLocation`

The fill should extend beyond the starting location. The color that extends beyond the starting point is the solid color defined by the `CGGradient` object to be at location 0.

Available in iOS 2.0 and later.

Declared in `CGGradient.h`.

`kCGGradientDrawsAfterEndLocation`

The fill should extend beyond the ending location. The color that extends beyond the ending point is the solid color defined by the `CGGradient` object to be at location 1.

Available in iOS 2.0 and later.

Declared in `CGGradient.h`.

Declared In

`CGGradient.h`

CGImage Reference

Derived From:	<i>CType Reference</i>
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGImage.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGImageRef` opaque type represents bitmap images and bitmap image masks, based on sample data that you supply. A bitmap (or sampled) image is a rectangular array of pixels, with each pixel representing a single sample or data point in a source image.

Functions by Task

Creating Bitmap Images

[CGImageCreate](#) (page 203)

Creates a bitmap image from data supplied by a data provider.

[CGImageCreateCopy](#) (page 204)

Creates a copy of a bitmap image.

[CGImageCreateCopyWithColorSpace](#) (page 205)

Create a copy of a bitmap image, replacing its colorspace.

[CGImageCreateWithJPEGDataProvider](#) (page 206)

Creates a bitmap image using JPEG-encoded data supplied by a data provider.

[CGImageCreateWithPNGDataProvider](#) (page 208)

Creates a Quartz bitmap image using PNG-encoded data supplied by a data provider.

[CGImageCreateWithImageInRect](#) (page 205)

Creates a bitmap image using the data contained within a subregion of an existing bitmap image.

[CGImageCreateWithMask](#) (page 207)

Creates a bitmap image from an existing image and an image mask.

[CGImageCreateWithMaskingColors](#) (page 207)

Creates a bitmap image by masking an existing bitmap image with the provided color values.

Creating an Image Mask

[CGImageMaskCreate](#) (page 215)

Creates a bitmap image mask from data supplied by a data provider.

Retaining and Releasing Images

[CGImageRetain](#) (page 217)

Increments the retain count of a bitmap image.

[CGImageRelease](#) (page 217)

Decrements the retain count of a bitmap image.

Getting the CType ID

[CGImageGetTypeID](#) (page 214)

Returns the type identifier for Quartz bitmap images.

Getting Information About an Image

[CGImageGetAlphaInfo](#) (page 209)

Returns the alpha channel information for a bitmap image.

[CGImageGetBitmapInfo](#) (page 209)

Returns the bitmap information for a bitmap image.

[CGImageGetBitsPerComponent](#) (page 210)

Returns the number of bits allocated for a single color component of a bitmap image.

[CGImageGetBitsPerPixel](#) (page 210)

Returns the number of bits allocated for a single pixel in a bitmap image.

[CGImageGetBytesPerRow](#) (page 211)

Returns the number of bytes allocated for a single row of a bitmap image.

[CGImageGetColorSpace](#) (page 211)

Return the color space for a bitmap image.

[CGImageGetDataProvider](#) (page 212)

Returns the data provider for a bitmap image.

[CGImageGetDecode](#) (page 212)

Returns the decode array for a bitmap image.

[CGImageGetHeight](#) (page 212)

Returns the height of a bitmap image.

[CGImageGetShouldInterpolate](#) (page 213)

Returns the interpolation setting for a bitmap image.

[CGImageGetRenderingIntent](#) (page 213)

Returns the rendering intent setting for a bitmap image.

[CGImageGetWidth](#) (page 214)

Returns the width of a bitmap image.

[CGImageIsMask](#) (page 215)

Returns whether a bitmap image is an image mask.

Functions

CGImageCreate

Creates a bitmap image from data supplied by a data provider.

```
CGImageRef CGImageCreate (
    size_t width,
    size_t height,
    size_t bitsPerComponent,
    size_t bitsPerPixel,
    size_t bytesPerRow,
    CGColorSpaceRef colorspace,
    CGBitmapInfo bitmapInfo,
    CGDataProviderRef provider,
    const CGFloat decode[],
    bool shouldInterpolate,
    CGColorRenderingIntent intent
);
```

Parameters

width

The width, in pixels, of the required image.

height

The height, in pixels, of the required image

bitsPerComponent

The number of bits for each component in a source pixel. For example, if the source image uses the RGBA-32 format, you would specify 8 bits per component.

bitsPerPixel

The total number of bits in a source pixel. This value must be at least `bitsPerComponent` times the number of components per pixel.

bytesPerRow

The number of bytes of memory for each horizontal row of the bitmap.

colorspace

The color space for the image. Quartz retains the color space you pass in; on return, you may safely release it.

bitmapInfo

A `CGBitmapInfo` constant that specifies whether the bitmap should contain an alpha channel and its relative location in a pixel, along with whether the components are floating-point or integer values.

provider

The source of data for the bitmap. For information about supported data formats, see the discussion below. Quartz retains this object; on return, you may safely release it.

decode

The decode array for the image. If you do not want to allow remapping of the image's color values, pass `NULL` for the decode array. For each color component in the image's color space (including the alpha component), a decode array provides a pair of values denoting the upper and lower limits of a range. For example, the decode array for a source image in the RGB color space would contain six entries total, consisting of one pair each for red, green, and blue. When the image is rendered, Quartz uses a linear transform to map the original component value into a relative number within your designated range that is appropriate for the destination color space.

shouldInterpolate

A Boolean value that specifies whether interpolation should occur. The interpolation setting specifies whether Quartz should apply a pixel-smoothing algorithm to the image. Without interpolation, the image may appear jagged or pixelated when drawn on an output device with higher resolution than the image data.

intent

A rendering intent constant that specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context. The rendering intent determines the exact method used to map colors from one color space to another. For descriptions of the defined rendering-intent constants, see [Color Rendering Intents](#) (page 48).

Return Value

A new Quartz bitmap image. You are responsible for releasing this object by calling [CGImageRelease](#) (page 217).

Discussion

The data provider should provide raw data that matches the format specified by the other input parameters. To use encoded data (for example, from a file specified by a URL-based data provider), see [CGImageCreateWithJPEGDataProvider](#) (page 206) and [CGImageCreateWithPNGDataProvider](#) (page 208). In Mac OS X version 10.3 and later, you can also use the QuickTime function `GraphicsImportCreateCGImage` to decode an image file in any supported format and create a `CGImage`, in a single operation.

For information on supported pixel formats, see *Quartz 2D Programming Guide*.

Availability**Declared In**

`CGImage.h`

CGImageCreateCopy

Creates a copy of a bitmap image.

```
CGImageRef CGImageCreateCopy (
    CGImageRef image
);
```

Parameters*image*

The image to copy.

Return Value

An copy of the image specified by the `image` parameter.

Availability**Declared In**

CGImage.h

CGImageCreateCopyWithColorSpace

Create a copy of a bitmap image, replacing its colorspace.

```
CGImageRef CGImageCreateCopyWithColorSpace (
    CGImageRef image,
    CGColorSpaceRef colorspace
);
```

Parameters*image*

The graphics image to copy.

colorspace

The destination color space. The number of components in this color space must be the same as the number in the specified image.

Return Value

A new Quartz image that is a copy of the image passed as the *image* parameter but with its color space replaced by that specified by the *colorspace* parameter. Returns `NULL` if *image* is an image mask, or if the number of components of *colorspace* is not the same as the number of components of the colorspace of *image*. You are responsible for releasing this object using [CGImageRelease](#) (page 217).

Availability**Declared In**

CGImage.h

CGImageCreateWithImageInRect

Creates a bitmap image using the data contained within a subregion of an existing bitmap image.

```
CGImageRef CGImageCreateWithImageInRect (
    CGImageRef image,
    CGRect rect
);
```

Parameters*image*

The image to extract the subimage from.

rect

A rectangle whose coordinates specify the area to create an image from.

Return Value

A `CGImage` object that specifies a subimage of the image. If the *rect* parameter defines an area that is not in the image, returns `NULL`.

Discussion

Quartz performs these tasks to create the subimage:

- Adjusts the area specified by the `rect` parameter to integral bounds by calling the function `CGRectIntegral`.
- Intersects the result with a rectangle whose origin is (0, 0) and size is equal to the size of the image specified by the `image` parameter.
- References the pixels within the resulting rectangle, treating the first pixel within the rectangle as the origin of the subimage.

If `W` and `H` are the width and height of image, respectively, then the point (0, 0) corresponds to the first pixel of the image data. The point (`W-1`, 0) is the last pixel of the first row of the image data while (0, `H-1`) is the first pixel of the last row of the image data and (`W-1`, `H-1`) is the last pixel of the last row of the image data.

The resulting image retains a reference to the original image, which means you may release the original image after calling this function.

Availability

Declared In

`CGImage.h`

CGImageCreateWithJPEGDataProvider

Creates a bitmap image using JPEG-encoded data supplied by a data provider.

```
CGImageRef CGImageCreateWithJPEGDataProvider (
    CGDataProviderRef source,
    const CGFloat decode[],
    bool shouldInterpolate,
    CGColorRenderingIntent intent
);
```

Parameters

source

A data provider supplying JPEG-encoded data.

decode

The decode array for the image. Typically a decode array is unnecessary, and you should pass `NULL`.

shouldInterpolate

A Boolean value that specifies whether interpolation should occur. The interpolation setting specifies whether Quartz should apply a pixel-smoothing algorithm to the image.

intent

A `CGColorRenderingIntent` constant that specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context.

Return Value

A new Quartz bitmap image. You are responsible for releasing this object by calling [CGImageRelease](#) (page 217).

Availability

Declared In

`CGImage.h`

CGImageCreateWithMask

Creates a bitmap image from an existing image and an image mask.

```
CGImageRef CGImageCreateWithMask (
    CGImageRef image,
    CGImageRef mask
);
```

Parameters

image

The image to apply the `mask` parameter to. This image must not be an image mask and may not have an image mask or masking color associated with it.

mask

A mask. If the mask is an image, it must be in the DeviceGray color space, must not have an alpha component, and may not itself be masked by an image mask or a masking color. If the mask is not the same size as the image specified by the `image` parameter, then Quartz scales the mask to fit the image.

Return Value

An image created by masking `image` with `mask`. You are responsible for releasing this object by calling [CGImageRelease](#) (page 217).

Discussion

The resulting image depends on whether the `mask` parameter is an image mask or an image. If the `mask` parameter is an image mask, then the source samples of the image mask act as an inverse alpha value. That is, if the value of a source sample in the image mask is *S*, then the corresponding region in `image` is blended with the destination using an alpha value of (1-*S*). For example, if *S* is 1, then the region is not painted, while if *S* is 0, the region is fully painted.

If the `mask` parameter is an image, then it serves as an alpha mask for blending the image onto the destination. The source samples of `mask` act as an alpha value. If the value of the source sample in `mask` is *S*, then the corresponding region in `image` is blended with the destination with an alpha of *S*. For example, if *S* is 0, then the region is not painted, while if *S* is 1, the region is fully painted.

Availability

Related Sample Code

TheElements

Declared In

CGImage.h

CGImageCreateWithMaskingColors

Creates a bitmap image by masking an existing bitmap image with the provided color values.

```
CGImageRef CGImageCreateWithMaskingColors (
    CGImageRef image,
    const CGFloat components[]
);
```

Parameters

image

The image to mask. This parameter may not be an image mask, may not already have an image mask or masking color associated with it, and cannot have an alpha component.

components

An array of color components that specify a color or range of colors to mask the image with. The array must contain 2N values { min[1], max[1], ... min[N], max[N] } where N is the number of components in color space of *image*. Each value in *components* must be a valid image sample value. If *image* has integer pixel components, then each value must be in the range [0 .. 2**bitsPerComponent - 1] (where *bitsPerComponent* is the number of bits/component of *image*). If *image* has floating-point pixel components, then each value may be any floating-point number which is a valid color component.

Return Value

An image created by masking *image* with the colors specified in the *components* array. You are responsible for releasing this object by calling [CGImageRelease](#) (page 217).

Discussion

Any image sample with color value {c[1], ... c[N]} where min[i] <= c[i] <= max[i] for 1 <= i <= N is masked out (that is, not painted). This means that anything underneath the unpainted samples, such as the current fill color, shows through.

Availability

Declared In

CGImage.h

CGImageCreateWithPNGDataProvider

Creates a Quartz bitmap image using PNG-encoded data supplied by a data provider.

```
CGImageRef CGImageCreateWithPNGDataProvider (
    CGDataProviderRef source,
    const CGFloat decode[],
    bool shouldInterpolate,
    CGColorRenderingIntent intent
);
```

Parameters

source

A data provider supplying PNG-encoded data.

decode

The decode array for the image. Typically a decode array is unnecessary, and you should pass `NULL`.

shouldInterpolate

A Boolean value that specifies whether interpolation should occur. The interpolation setting specifies whether Quartz should apply a pixel-smoothing algorithm to the image.

intent

A `CGColorRenderingIntent` constant that specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context.

Return Value

A new Quartz bitmap image. You are responsible for releasing this object by calling [CGImageRelease](#) (page 217).

Availability**Declared In**

CGImage.h

CGImageGetAlphaInfo

Returns the alpha channel information for a bitmap image.

```
CGImageAlphaInfo CGImageGetAlphaInfo (
    CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

A `CGImageAlphaInfo` constant that specifies (1) whether the bitmap contains an alpha channel, (2) where the alpha bits are located in the image data, and (3) whether the alpha value is premultiplied. For possible values, see [“Constants”](#) (page 218). The function returns `kCGImageAlphaNone` if the `image` parameter refers to an image mask.

Discussion

The alpha value is what determines the opacity of a pixel when it is drawn.

Availability**Declared In**

CGImage.h

CGImageGetBitmapInfo

Returns the bitmap information for a bitmap image.

```
CGBitmapInfo CGImageGetBitmapInfo (
    CGImageRef image
);
```

Parameters

image

An image.

Return Value

The bitmap information associated with an image.

Discussion

This function returns a constant that specifies:

- The type of bitmap data—floating point or integer. You use the constant `kCGBitmapFloatComponents` to extract this information.

- Whether an alpha channel is in the data, and if so, how the alpha data is stored. You use the constant `kCGBitmapAlphaInfoMask` to extract the alpha information. Alpha information is specified as one of the constants listed in [“Alpha Information for Images”](#) (page 218).

You can extract the alpha information

Availability**Related Sample Code**

GLImageProcessing

Declared In

CGImage.h

CGImageGetBitsPerComponent

Returns the number of bits allocated for a single color component of a bitmap image.

```
size_t CGImageGetBitsPerComponent (
    CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

The number of bits used in memory for each color component of the specified bitmap image (or image mask). Possible values are 1, 2, 4, or 8. For example, for a 16-bit RGB(A) colorspace, the function would return a value of 4 bits per color component.

Availability**Declared In**

CGImage.h

CGImageGetBitsPerPixel

Returns the number of bits allocated for a single pixel in a bitmap image.

```
size_t CGImageGetBitsPerPixel (
    CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

The number of bits used in memory for each pixel of the specified bitmap image (or image mask).

Availability**Related Sample Code**

GLImageProcessing

Declared In

CGImage.h

CGImageGetBytesPerRow

Returns the number of bytes allocated for a single row of a bitmap image.

```
size_t CGImageGetBytesPerRow (  
    CGImageRef image  
);
```

Parameters*image*

The image to examine.

Return Value

The number of bytes used in memory for each row of the specified bitmap image (or image mask).

Availability**Related Sample Code**

GLImageProcessing

Declared In

CGImage.h

CGImageGetColorSpace

Return the color space for a bitmap image.

```
CGColorSpaceRef CGImageGetColorSpace (  
    CGImageRef image  
);
```

Parameters*image*

The image to examine.

Return Value

The source color space for the specified bitmap image, or `NULL` if the image is an image mask. You are responsible for retaining and releasing the color space as necessary.

Availability**Related Sample Code**

aurioTouch

GLImageProcessing

GLPaint

GLSprite

MusicCube

Declared In

CGImage.h

CGImageGetDataProvider

Returns the data provider for a bitmap image.

```
CGDataProviderRef CGImageGetDataProvider (
    CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

The data provider for the specified bitmap image (or image mask). You are responsible for retaining and releasing the data provider as necessary.

Availability

Related Sample Code

GLImageProcessing

Declared In

CGImage.h

CGImageGetDecode

Returns the decode array for a bitmap image.

```
const CGFloat * CGImageGetDecode (
    CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

The decode array for a bitmap image (or image mask). See the discussion for a description of possible return values.

Discussion

For a bitmap image or image mask, for each color component in the source color space, the decode array contains a pair of values denoting the upper and lower limits of a range. When the image is rendered, Quartz uses a linear transform to map the original component value into a relative number, within the designated range, that is appropriate for the destination color space. If remapping of the image's color values is not allowed, the returned value will be `NULL`.

Availability

Declared In

CGImage.h

CGImageGetHeight

Returns the height of a bitmap image.

```
size_t CGImageGetHeight (
    CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

The height in pixels of the bitmap image (or image mask).

Availability**Related Sample Code**

aurioTouch

GLImageProcessing

GLPaint

oalTouch

PVRTextureLoader

Declared In

CGImage.h

CGImageGetRenderingIntent

Returns the rendering intent setting for a bitmap image.

```
CGColorRenderingIntent CGImageGetRenderingIntent (
    CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

Returns the `CGColorRenderingIntent` constant that specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context in which the image is drawn. If the image is an image mask, this function returns `kCGRenderingIntentDefault`.

Availability**Declared In**

CGImage.h

CGImageGetShouldInterpolate

Returns the interpolation setting for a bitmap image.

```
bool CGImageGetShouldInterpolate (  
    CGImageRef image  
);
```

Parameters

image

The image to examine.

Return Value

Returns 1 if interpolation is enabled for the specified bitmap image (or image mask), otherwise, returns 0.

Discussion

The interpolation setting specifies whether Quartz should apply an edge-smoothing algorithm to the associated image.

Availability**Declared In**

CGImage.h

CGImageGetTypeID

Returns the type identifier for Quartz bitmap images.

```
CTypeID CGImageGetTypeID (  
    void  
);
```

Return Value

The identifier for the opaque type [CGImageRef](#) (page 218).

Availability**Declared In**

CGImage.h

CGImageGetWidth

Returns the width of a bitmap image.

```
size_t CGImageGetWidth (  
    CGImageRef image  
);
```

Parameters

image

The image to examine.

Return Value

The width, in pixels, of the specified bitmap image (or image mask).

Availability**Related Sample Code**

aurioTouch

GLImageProcessing
 GLPaint
 oalTouch
 PVRTextureLoader

Declared In
 CGImage.h

CGImageIsMask

Returns whether a bitmap image is an image mask.

```
bool CGImageIsMask (
    CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

A Boolean value that indicates whether the image passed in the *image* parameter is an image mask (*true* indicates that the image is an image mask).

Availability

Declared In
 CGImage.h

CGImageMaskCreate

Creates a bitmap image mask from data supplied by a data provider.

```
CGImageRef CGImageMaskCreate (
    size_t width,
    size_t height,
    size_t bitsPerComponent,
    size_t bitsPerPixel,
    size_t bytesPerRow,
    CGDataProviderRef provider,
    const CGFloat decode[],
    bool shouldInterpolate
);
```

Parameters

width

The width, in pixels, of the required image mask.

height

The height, in pixels, of the required image mask.

bitsPerComponent

The number of significant masking bits in a source pixel. For example, if the source image is an 8-bit mask, you specify 8 bits per component. Image masks must be 1, 2, 4, or 8 bits per component.

bitsPerPixel

The total number of bits in a source pixel.

bytesPerRow

The number of bytes to use for each horizontal row of the image mask.

provider

The data source for the image mask.

decode

Typically a decode array is unnecessary, and you should pass `NULL`.

shouldInterpolate

A Boolean value that specifies whether interpolation should occur. The interpolation setting specifies whether Quartz should apply an edge-smoothing algorithm to the image mask.

Return Value

A Quartz bitmap image mask. You are responsible for releasing this object by calling [CGImageRelease](#) (page 217).

Discussion

A Quartz bitmap image mask is used the same way an artist uses a silkscreen, or a sign painter uses a stencil. The bitmap represents a mask through which a color is transferred. The bitmap itself does not have a color. It gets its color from the fill color currently set in the graphics state.

When you draw into a context with a bitmap image mask, Quartz uses the mask to determine where and how the current fill color is applied to the image rectangle. Each sample value in the mask specifies how much of the current fill color is masked out at a specific location. Effectively, the sample value specifies the opacity of the mask. Larger values represent greater opacity and hence less color applied to the page.

Image masks must be 1, 2, 4, or 8 bits per component. For a 1-bit mask, a sample value of 1 specifies sections of the mask that are masked out; these sections block the current fill color. A sample value of 0 specifies sections of the mask that are not masked out; these sections show the current fill color of the graphics state when the mask is painted. You can think of the sample values as an inverse alpha. That is, a value of 1 is transparent and 0 is opaque.

For image masks that are 2, 4, or 8 bits per component, each component is mapped to a range of 0 to 1 by scaling using this formula:

$$1/(2^{\text{bits per component}} - 1)$$

For example, a 4-bit mask has values that range from 0 to 15. These values are scaled by 1/15 so that each component ranges from 0 to 1. Component values that rescale to 0 or 1 behave the same way as they behave for 1-bit image masks. Values that scale to between 0 and 1 act as an inverse alpha. That is, the fill color is painted as if it has an alpha value of $(1 - \text{MaskSampleValue})$. For example, if the sample value of an 8-bit mask scales to 0.8, the current fill color is painted as if it has an alpha value of 0.2, that is $(1 - 0.8)$.

Availability

Related Sample Code

QuartzDemo

Declared In

CGImage.h

CGImageRelease

Decrements the retain count of a bitmap image.

```
void CGImageRelease (  
    CGImageRef image  
);
```

Parameters

image

The image to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `image` parameter is `NULL`.

Availability

Related Sample Code

GLImageProcessing

oalTouch

QuartzDemo

Reflection

TheElements

Declared In

CGImage.h

CGImageRetain

Increments the retain count of a bitmap image.

```
CGImageRef CGImageRetain (  
    CGImageRef image  
);
```

Parameters

image

The image to retain.

Return Value

The same image you passed in as the `image` parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `image` parameter is `NULL`.

Availability

Related Sample Code

oalTouch

QuartzDemo

Declared In

CGImage.h

Data Types

CGImageRef

An opaque type that encapsulates bitmap image information.

```
typedef struct CGImage *CGImageRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGImage.h

Constants

Alpha Information for Images

Storage options for alpha component data.

```
enum CGImageAlphaInfo {  
    kCGImageAlphaNone,  
    kCGImageAlphaPremultipliedLast,  
    kCGImageAlphaPremultipliedFirst,  
    kCGImageAlphaLast,  
    kCGImageAlphaFirst,  
    kCGImageAlphaNoneSkipLast,  
    kCGImageAlphaNoneSkipFirst  
};  
typedef enum CGImageAlphaInfo CGImageAlphaInfo;
```

Constants

kCGImageAlphaFirst

The alpha component is stored in the most significant bits of each pixel. For example, non-premultiplied ARGB.

Available in iOS 2.0 and later.

Declared in CGImage.h.

kCGImageAlphaLast

The alpha component is stored in the least significant bits of each pixel. For example, non-premultiplied RGBA.

Available in iOS 2.0 and later.

Declared in CGImage.h.

kCGImageAlphaNone

There is no alpha channel. If the total size of the pixel is greater than the space required for the number of color components in the color space, the least significant bits are ignored. This value is equivalent to `kCGImageAlphaNoneSkipLast`.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGImageAlphaNoneSkipFirst

There is no alpha channel. If the total size of the pixel is greater than the space required for the number of color components in the color space, the most significant bits are ignored.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGImageAlphaOnly

There is no color data, only an alpha channel.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGImageAlphaNoneSkipLast

There is no alpha channel. If the total size of the pixel is greater than the space required for the number of color components in the color space, the least significant bits are ignored. This value is equivalent to `kCGImageAlphaNone`.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGImageAlphaPremultipliedFirst

The alpha component is stored in the most significant bits of each pixel and the color components have already been multiplied by this alpha value. For example, premultiplied ARGB.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGImageAlphaPremultipliedLast

The alpha component is stored in the least significant bits of each pixel and the color components have already been multiplied by this alpha value. For example, premultiplied RGBA.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

Discussion

A `CGImageAlphaInfo` constant specifies (1) whether a bitmap contains an alpha channel, (2) where the alpha bits are located in the image data, and (3) whether the alpha value is premultiplied. You can obtain a `CGImageAlphaInfo` constant for an image by calling the function `CGImageGetAlphaInfo` (page 209). (You provide a `CGBitmapInfo` constant to the function `CGImageCreate` (page 203), part of which is a `CGImageAlphaInfo` constant.)

Quartz accomplishes alpha blending by combining the color components of the source image with the color components of the destination image using the linear interpolation formula, where “source” is one color component of one pixel of the new paint and “destination” is one color component of the background image.

Quartz supports premultiplied alpha only for images. You should not premultiply any other color values specified in Quartz.

Declared In

`CGImage.h`

Image Bitmap Information

Component information for a bitmap image.

```
enum {
    kCGBitmapAlphaInfoMask = 0x1F,
    kCGBitmapFloatComponents = (1 << 8),

    kCGBitmapByteOrderMask = 0x7000,
    kCGBitmapByteOrderDefault = (0 << 12),
    kCGBitmapByteOrder16Little = (1 << 12),
    kCGBitmapByteOrder32Little = (2 << 12),
    kCGBitmapByteOrder16Big = (3 << 12),
    kCGBitmapByteOrder32Big = (4 << 12)
};
typedef uint32_t CGBitmapInfo;
```

Constants

kCGBitmapAlphaInfoMask

The alpha information mask. Use this to extract alpha information that specifies whether a bitmap contains an alpha channel and how the alpha channel is generated.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGBitmapFloatComponents

The components of a bitmap are floating-point values.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGBitmapByteOrderMask

The byte ordering of pixel formats.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGBitmapByteOrderDefault

The default byte order.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGBitmapByteOrder16Little

16-bit, little endian format.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGBitmapByteOrder32Little

32-bit, little endian format.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

kCGBitmapByteOrder16Big

16-bit, big endian format.

Available in iOS 2.0 and later.

Declared in `CGImage.h`.

`kCGBitmapByteOrder32Big`
 32-bit, big endian format.
 Available in iOS 2.0 and later.
 Declared in `CGImage.h`.

Discussion

Applications that store pixel data in memory using ARGB format must take care in how they read data. If the code is not written correctly, it's possible to misread the data which leads to colors or alpha that appear wrong. The Quartz byte order constants specify the byte ordering of pixel formats. To specify byte ordering to Quartz use a bitwise OR operator to combine the appropriate constant with the `bitmapInfo` parameter.

Host Endian Bitmap Formats

Bit-depth constants for image bitmaps in host-endian byte order.

```
#ifdef __BIG_ENDIAN__
#define kCGBitmapByteOrder16Host kCGBitmapByteOrder16Big
#define kCGBitmapByteOrder32Host kCGBitmapByteOrder32Big
#else
#define kCGBitmapByteOrder16Host kCGBitmapByteOrder16Little
#define kCGBitmapByteOrder32Host kCGBitmapByteOrder32Little
#endif
```

Constants

`kCGBitmapByteOrder16Host`
 16-bit, host endian format.
 Available in iOS 2.0 and later.
 Declared in `CGImage.h`.

`kCGBitmapByteOrder32Host`
 32-bit, host endian format.
 Available in iOS 2.0 and later.
 Declared in `CGImage.h`.

CGLayer Reference

Derived From:	CType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGLayer.h

Overview

CGLayer objects are useful for offscreen drawing and can be used in much the same way that a bitmap context can be used. In fact, a CGLayer object is a much better representation than a bitmap context.

Using CGLayer objects can improve performance, particularly when you need to capture a piece of drawing that you stamp repeatedly (using the same scale factor and orientation). Quartz can cache CGLayer objects to the video card, making drawing a CGLayer to a destination much faster than rendering the equivalent image constructed from a bitmap context.

A CGLayer object is created relative to a graphics context. Although layer uses this graphics context as a reference for initialization, you are not restricted to drawing the layer to this graphics context. You can draw the layer to other graphics contexts, although any limitations of the original context are imposed. For example, if you create a CGLayer object using a bitmap context, the layer is rendered as a bitmap when drawn to any other graphics context.

You can use a CGLayer when you want to apply a shadow to a group of objects (such as a group of circles) rather than to individual objects.

Use these layers in your code whenever you can, especially when:

- You need to reuse a filled or stroked shape.
- You are building a scene and at least some of it can be reused. Put the reusable drawing in its own CGLayer.

Any CG object that you draw repeatedly—including CGPath, CGShading, and CGPDFPage—benefit from improved performance if you draw it to a CGLayer object.

Functions by Task

Creating Layer Objects

[CGLayerCreateWithContext](#) (page 225)

Creates a CGLayer object that is associated with a graphics context.

Drawing Layer Content

[CGContextDrawLayerInRect](#) (page 225)

Draws the contents of a CGLayer object into the specified rectangle.

[CGContextDrawLayerAtPoint](#) (page 224)

Draws the contents of a CGLayer object at the specified point.

Retaining and Releasing Layers

[CGLayerRelease](#) (page 227)

Decrements the retain count of a CGLayer object.

[CGLayerRetain](#) (page 228)

Increments the retain count of a CGLayer object.

Getting the CType ID for a Layer

[CGLayerGetTypeID](#) (page 227)

Returns the unique type identifier used for CGLayer objects.

Getting Layer Information

[CGLayerGetSize](#) (page 226)

Returns the width and height of a CGLayer object.

[CGLayerGetContext](#) (page 226)

Returns the graphics context associated with a CGLayer object.

Functions

CGContextDrawLayerAtPoint

Draws the contents of a CGLayer object at the specified point.

```
void CGContextDrawLayerAtPoint (
    CGContextRef context,
    CGPoint point,
    CGLayerRef layer
);
```

Parameters

context

The graphics context associated with the layer.

point

The location, in current user space coordinates, to use as the origin for the drawing.

layer

The layer whose contents you want to draw.

Discussion

Calling the function `CGContextDrawLayerAtPoint` is equivalent to calling the function `CGContextDrawLayerInRect` with a rectangle that has its origin at `point` and its size equal to the size of the layer.

Availability

Declared In

`CGLayer.h`

CGContextDrawLayerInRect

Draws the contents of a CGLayer object into the specified rectangle.

```
void CGContextDrawLayerInRect (
    CGContextRef context,
    CGRect rect,
    CGLayerRef layer
);
```

Parameters

context

The graphics context associated with the layer.

rect

The rectangle, in current user space coordinates, to draw to.

layer

The layer whose contents you want to draw.

Discussion

The contents are scaled, if necessary, to fit into the rectangle.

Availability

Declared In

`CGLayer.h`

CGLayerCreateWithContext

Creates a CGLayer object that is associated with a graphics context.

```
CGLayerRef CGLayerCreateWithContext (
    CGContextRef context,
    CGSize size,
    CFDictionaryRef auxiliaryInfo
);
```

Parameters

context

The graphics context you want to create the layer relative to. The layer uses this graphics context as a reference for initialization.

size

The size, in default user space units, of the layer relative to the graphics context.

auxiliaryInfo

Reserved for future use. Pass `NULL`.

Return Value

A `CGLayer` object. You are responsible for releasing this object using the function `CGLayerRelease` (page 227) when you no longer need the layer.

Discussion

After you create a `CGLayer` object, you should reuse it whenever you can to facilitate the Quartz caching strategy. Quartz caches any objects that are reused, including `CGLayer` objects. Objects that are reused frequently remain in the cache. In contrast, objects that are used once in a while may be moved in and out of the cache according to their frequency of use. If you don't reuse `CGLayer` objects, Quartz won't cache them. This means that you lose an opportunity to improve the performance of your application.

Availability

Available in iOS 2.0 and later.

Declared In

`CGLayer.h`

CGLayerGetContext

Returns the graphics context associated with a `CGLayer` object.

```
CGContextRef CGLayerGetContext (
    CGLayerRef layer
);
```

Parameters

layer

The layer whose graphics context you want to obtain.

Return Value

The graphics context associated with the layer.

Discussion

The context that's returned is the context for the layer itself, not the context that you specified when you created the layer.

Availability

Available in iOS 2.0 and later.

Declared In

`CGLayer.h`

CGLayerGetSize

Returns the width and height of a `CGLayer` object.

```
CGSize CGLayerGetSize (
    CGLayerRef layer
);
```

Parameters

layer

The layer whose width and height you want to obtain.

Return Value

The width and height of the layer, in default user space coordinates.

Availability

Available in iOS 2.0 and later.

Declared In

CGLayer.h

CGLayerGetTypeID

Returns the unique type identifier used for CGLayer objects.

```
CTypeID CGLayerGetTypeID (
    void
);
```

Return Value

The type identifier for CGLayer objects.

Discussion

A type identifier is an integer that identifies the opaque type to which a Core Foundation object belongs. You use type IDs in various contexts, such as when you are operating on heterogeneous collections.

Availability

Available in iOS 2.0 and later.

Declared In

CGLayer.h

CGLayerRelease

Decrements the retain count of a CGLayer object.

```
void CGLayerRelease (
    CGLayerRef layer
);
```

Parameters

layer

The layer to release.

Discussion

This function is equivalent to calling `CFRelease (layer)` except that it does not crash (as `CFRetain` does) if the `layer` parameter is `null`.

Availability

Available in iOS 2.0 and later.

Declared In

CGLayer.h

CGLayerRetain

Increments the retain count of a CGLayer object.

```
CGLayerRef CGLayerRetain (  
    CGLayerRef layer  
);
```

Parameters

layer

The layer to retain.

Return Value

The same layer you passed in as the *layer* parameter.

Discussion

This function is equivalent to calling `CFRetain (layer)` except that it does not crash (as `CFRetain` does) if the *layer* parameter is `null`.

Availability

Available in iOS 2.0 and later.

Declared In

CGLayer.h

Data Types

CGLayerRef

An opaque type used for offscreen drawing.

```
typedef struct CGLayer *CGLayerRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGLayer.h

CGPath Reference

Derived From:	<i>CType Reference</i>
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPath.h
Companion guide	Quartz 2D Programming Guide

Overview

A **graphics path** is a mathematical description of a series of shapes or lines. `CGPathRef` defines an opaque type that represents an immutable graphics path. `CGMutablePathRef` defines an opaque type that represents a mutable graphics path. Neither `CGPathRef` nor `CGMutablePathRef` define functions to draw a path. To draw a Quartz path to a graphics context, you add the path to the graphics context by calling `CGContextAddPath` (page 64) and then call one of the context's drawing functions—see *CGContext Reference*.

Each figure in the graphics path is constructed with a connected set of lines and Bézier curves, called a **subpath**. A subpath has an ordered set of **path elements** that represent single steps in the construction of the subpath. (For example, a line segment from one corner of a rectangle to another corner is a path element. Every subpath includes a **starting point**, which is the first point in the subpath. The path also maintains a **current point**, which is the last point in the last subpath.

To append a new subpath onto a mutable path, your application typically calls `CGPathMoveToPoint` (page 246) to set the subpath's starting point and initial current point, followed by a series of `CGPathAdd*` calls to add line segments and curves to the subpath. As segments or curves are added to the subpath, the subpath's current point is updated to point to the end of the last segment or curve to be added. The lines and curves of a subpath are always connected, but they are not required to form a closed set of lines. Your application explicitly closes a subpath by calling `CGPathCloseSubpath` (page 240). Closing the subpath adds a line segment that terminates at the subpath's starting point, and also changes how those lines are rendered—for more information see “Paths” in *Quartz 2D Programming Guide*.

Functions by Task

Creating and Managing Paths

`CGPathCreateMutable` (page 241)

Creates a mutable graphics path.

`CGPathCreateMutableCopy` (page 242)

Creates a mutable copy of an existing graphics path.

[CGPathCreateCopy](#) (page 241)
Creates an immutable copy of a graphics path.

[CGPathRelease](#) (page 246)
Decrements the retain count of a graphics path.

[CGPathRetain](#) (page 247)
Increments the retain count of a graphics path.

Modifying Quartz Paths

[CGPathAddArc](#) (page 231)
Appends an arc to a mutable graphics path, possibly preceded by a straight line segment.

[CGPathAddArcToPoint](#) (page 232)
Appends an arc to a mutable graphics path, possibly preceded by a straight line segment.

[CGPathAddCurveToPoint](#) (page 233)
Appends a cubic Bézier curve to a mutable graphics path.

[CGPathAddLines](#) (page 235)
Appends an array of new line segments to a mutable graphics path.

[CGPathAddLineToPoint](#) (page 235)
Appends a line segment to a mutable graphics path.

[CGPathAddPath](#) (page 236)
Appends a path to onto a mutable graphics path.

[CGPathAddQuadCurveToPoint](#) (page 237)
Appends a quadratic Bézier curve to a mutable graphics path.

[CGPathAddRect](#) (page 238)
Appends a rectangle to a mutable graphics path.

[CGPathAddRects](#) (page 238)
Appends an array of rectangles to a mutable graphics path.

[CGPathApply](#) (page 239)
For each element in a graphics path, calls a custom applier function.

[CGPathMoveToPoint](#) (page 246)
Starts a new subpath at a specified location in a mutable graphics path.

[CGPathCloseSubpath](#) (page 240)
Closes and completes a subpath in a mutable graphics path.

[CGPathAddEllipseInRect](#) (page 234)
Adds to a path an ellipse that fits inside a rectangle.

Getting Information about Quartz Paths

[CGPathEqualToPath](#) (page 242)
Indicates whether two graphics paths are equivalent.

[CGPathGetBoundingBox](#) (page 243)
Returns the bounding box containing all points in a graphics path.

[CGPathGetPathBoundingBox](#) (page 244)

Returns the bounding box of a graphics path.

[CGPathGetCurrentPoint](#) (page 243)

Returns the current point in a graphics path.

[CGPathGetTypeID](#) (page 244)

Returns the Core Foundation type identifier for Quartz graphics paths.

[CGPathIsEmpty](#) (page 245)

Indicates whether or not a graphics path is empty.

[CGPathIsRect](#) (page 245)

Indicates whether or not a graphics path represents a rectangle.

[CGPathContainsPoint](#) (page 240)

Checks whether a point is contained in a graphics path.

Functions

CGPathAddArc

Appends an arc to a mutable graphics path, possibly preceded by a straight line segment.

```
void CGPathAddArc (
    CGMutablePathRef path,
    const CGAffineTransform *m,
    CGFloat x,
    CGFloat y,
    CGFloat radius,
    CGFloat startAngle,
    CGFloat endAngle,
    bool clockwise
);
```

Parameters

path

The mutable graphics path to change.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the arc before it is added to the path.

x

The x-coordinate of the center point of the arc.

y

The y-coordinate of the center point of the arc.

r

The radius of the arc.

startAngle

The angle (in radians) from the horizontal that determines the starting point of the arc.

endAngle

The angle (in radians) from the horizontal that determines the ending point of the arc.

clockwise

A Boolean value that specifies whether or not to draw the arc in the clockwise direction, before applying the transformation matrix.

Discussion

An arc is a segment of a circle with radius *r* centered at a point (*x*, *y*). When you call this function, you provide the center point, radius, and two angles in radians. Quartz uses this information to determine the end points of the arc, and then approximates the new arc using a sequence of cubic Bézier curves. The *clockwise* parameter determines the direction in which the arc is created. The actual direction may change depending on the coordinate system transformation applied to the path.

A transformation may be applied to the Bézier curves before they are added to the path. If no transform is needed, the second argument should be *NULL*.

If the specified path already contains a subpath, Quartz implicitly adds a line connecting the subpath's current point to the beginning of the arc. If the path is empty, Quartz creates a new subpath with a starting point set to the starting point of the arc.

The ending point of the arc becomes the new current point of the path.

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathAddArcToPoint

Appends an arc to a mutable graphics path, possibly preceded by a straight line segment.

```
void CGPathAddArcToPoint (
    CGMutablePathRef path,
    const CGAffineTransform *m,
    CGFloat x1,
    CGFloat y1,
    CGFloat x2,
    CGFloat y2,
    CGFloat radius
);
```

Parameters

path

The mutable path to change. The path must not be empty.

m

A pointer to an affine transformation matrix, or *NULL* if no transformation is needed. If specified, Quartz applies the transformation to the arc before it is added to the path.

x1

The x-coordinate of the user space for the end point of the first tangent line. The first tangent line is drawn from the current point to (*x1*, *y1*).

y1

The y-coordinate of the user space for the end point of the first tangent line. The first tangent line is drawn from the current point to (*x1*, *y1*).

x2

The x-coordinate of the user space for the end point of the second tangent line. The second tangent line is drawn from (x1,y1) to (x2,y2).

y2

The y-coordinate of the user space for the end point of the second tangent line. The second tangent line is drawn from (x1,y1) to (x2,y2).

radius

The radius of the arc, in user space coordinates.

Discussion

This function uses a sequence of cubic Bézier curves to create an arc that is tangent to the line from the current point to (x1,y1) and to the line from (x1,y1) to (x2,y2). The start and end points of the arc are located on the first and second tangent lines, respectively. The start and end points of the arc are also the “tangent points” of the lines.

If the current point and the first tangent point of the arc (the starting point) are not equal, Quartz appends a straight line segment from the current point to the first tangent point.

The ending point of the arc becomes the new current point of the path.

For another way to draw an arc in a path, see [CGPathAddArc](#) (page 231).

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch

oalTouch

WeatherMap

Declared In

CGPath.h

CGPathAddCurveToPoint

Appends a cubic Bézier curve to a mutable graphics path.

```
void CGPathAddCurveToPoint (
    CGMutablePathRef path,
    const CGAffineTransform *m,
    CGFloat cp1x,
    CGFloat cp1y,
    CGFloat cp2x,
    CGFloat cp2y,
    CGFloat x,
    CGFloat y
);
```

Parameters*path*

The mutable path to change. The path must not be empty.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the curve before it is added to the path.

cx1

The x-coordinate of the first control point.

cy1

The y-coordinate of the first control point.

cx2

The x-coordinate of the second control point.

cy2

The y-coordinate of the second control point.

x

The x-coordinate of the end point of the curve.

y

The y-coordinate of the end point of the curve.

Discussion

Appends a cubic Bézier curve from the current point in a path to the specified location using two control points, after an optional transformation. Before returning, this function updates the current point to the specified location (*x*, *y*).

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathAddEllipseInRect

Adds to a path an ellipse that fits inside a rectangle.

```
void CGPathAddEllipseInRect (
    CGMutablePathRef path,
    const CGAffineTransform *m,
    CGRect rect
);
```

Parameters*path*

The path to modify.

m

An affine transform to apply to the ellipse, or `NULL` if you don't want to transform the ellipse.

rect

A rectangle to enclose the ellipse.

Discussion

The ellipse is approximated by a sequence of Bézier curves. Its center is the midpoint of the rectangle defined by the `rect` parameter. If the rectangle is square, then the ellipse is circular with a radius equal to one-half the width (or height) of the rectangle. If the `rect` parameter specifies a rectangular shape, then the major and minor axes of the ellipse are defined by the width and height of the rectangle.

The ellipse forms a complete subpath of the path—that is, the ellipse drawing starts with a move-to operation and ends with a close-subpath operation, with all moves oriented in the clockwise direction. If you supply an affine transform, then the constructed Bézier curves that define the ellipse are transformed before they are added to the path.

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathAddLines

Appends an array of new line segments to a mutable graphics path.

```
void CGPathAddLines (
    CGMutablePathRef path,
    const CGAffineTransform *m,
    const CGPoint points[],
    size_t count
);
```

Parameters

path

The mutable path to change.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the lines before adding them to the path.

points

An array of points that specifies the line segments to add.

count

The number of elements in the array.

Discussion

This is a convenience function that adds a sequence of connected line segments to a path, using the following operation:

```
CGPathMoveToPoint (path, m, points[0].x, points[0].y);
for (k = 1; k < count; k++) {
    CGPathAddLineToPoint (path, m, points[k].x, points[k].y);
}
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathAddLineToPoint

Appends a line segment to a mutable graphics path.

```
void CGContextAddLineToPoint (
    CGContextRef path,
    const CGAffineTransform *m,
    CGFloat x,
    CGFloat y
);
```

Parameters*path*

The mutable path to change. The path must not be empty.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the line before it is added to the path.

x

The x-coordinate of the end point of the line.

y

The y-coordinate of the end point of the line.

Discussion

Before returning, this function updates the current point to the specified location (*x*, *y*).

Availability

Available in iOS 2.0 and later.

Related Sample Code

[aurioTouch](#)

[Breadcrumb](#)

[MoveMe](#)

[oalTouch](#)

[WeatherMap](#)

Declared In

`CGPath.h`

CGPathAddPath

Appends a path to onto a mutable graphics path.

```
void CGContextAddPath (
    CGContextRef path1,
    const CGAffineTransform *m,
    CGContextRef path2
);
```

Parameters*path1*

The mutable path to change.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to *path2* before it is added to *path1*.

path2

The path to add.

Discussion

If the source path is non-empty, then its path elements are appended in order onto the mutable path. After the call completes, the start point and current point of the path are those of the last subpath in *path2*.

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathAddQuadCurveToPoint

Appends a quadratic Bézier curve to a mutable graphics path.

```
void CGPathAddQuadCurveToPoint (
    CGMutablePathRef path,
    const CGAffineTransform *m,
    CGFloat cpx,
    CGFloat cpy,
    CGFloat x,
    CGFloat y
);
```

Parameters*path*

The mutable path to change. The path must not be empty.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the curve before adding it to the path.

cpx

The x-coordinate of the control point.

cpy

The y-coordinate of the control point.

x

The x-coordinate of the end point of the curve.

y

The y-coordinate of the end point of the curve.

Discussion

Before returning, this function updates the current point to the specified location (*x*, *y*).

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathAddRect

Appends a rectangle to a mutable graphics path.

```
void CGPathAddRect (
    CGMutablePathRef path,
    const CGAffineTransform *m,
    CGRect rect
);
```

Parameters

path

The mutable path to change.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the rectangle before adding it to the path.

rect

The rectangle to add.

Discussion

This is a convenience function that adds a rectangle to a path, using the following sequence of operations:

```
// start at origin
CGPathMoveToPoint (path, m, CGRectGetMinX(rect), CGRectGetMinY(rect));

// add bottom edge
CGPathAddLineToPoint (path, m, CGRectGetMaxX(rect), CGRectGetMinY(rect));

// add right edge
CGPathAddLineToPoint (path, m, CGRectGetMaxX(rect), CGRectGetMaxY(rect));

// add top edge
CGPathAddLineToPoint (path, m, CGRectGetMinX(rect), CGRectGetMaxY(rect));

// add left edge and close
CGPathCloseSubpath (path);
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathAddRects

Appends an array of rectangles to a mutable graphics path.

```
void CGContextAddRects (
    CGContextRef path,
    const CGAffineTransform *m,
    const CGRect rects[],
    size_t count
);
```

Parameters*path*

The mutable path to change.

m

An affine transformation matrix, or NULL if no transformation is needed. If specified, Quartz applies the transformation to the rectangles before adding them to the path.

rects

The array of new rectangles to add.

count

The number of elements in the array.

Discussion

This is a convenience function that adds an array of rectangles to a path, using the following operation:

```
for (k = 0; k < count; k++) {
    CGContextAddRect (path, m, rects[k]);
}
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathApply

For each element in a graphics path, calls a custom applier function.

```
void CGContextApply (
    CGContextRef path,
    void *info,
    CGContextApplierFunction function
);
```

Parameters*path*

The path to which the function will be applied.

info

A pointer to the user data that Quartz will pass to the function being applied, or NULL.

*function*A pointer to the function to apply. See [CGContextApplierFunction](#) (page 247) for more information.**Discussion**

For each element in the specified path, Quartz calls the applier function, which can examine (but not modify) the element.

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathCloseSubpath

Closes and completes a subpath in a mutable graphics path.

```
void CGPathCloseSubpath (
    CGMutablePathRef path
);
```

Parameters

path

The path to change.

Discussion

Appends a line from the current point to the starting point of the current subpath and ends the subpath.

After closing the subpath, your application can begin a new subpath without first calling [CGPathMoveToPoint](#) (page 246). In this case, a new subpath is implicitly created with a starting and current point equal to the previous subpath's starting point.

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch

oalTouch

Declared In

CGPath.h

CGPathContainsPoint

Checks whether a point is contained in a graphics path.

```
bool CGPathContainsPoint (
    CGPathRef path,
    const CGAffineTransform *m,
    CGPoint point,
    bool eoFill
);
```

Parameters

path

The path to evaluate the point against.

m

An affine transform. If *m* is not `NULL` then the point is transformed by this affine transform prior to determining whether the path contains the point.

point

The point to check.

eoFill

A Boolean value that, if `true`, specifies to use the even-odd fill rule to evaluate the painted region of the path. If `false`, the winding fill rule is used.

Return Value

Returns `true` if the point is contained in the path; `false` otherwise.

Discussion

A point is contained in a path if it would be inside the painted region when the path is filled.

Availability

Available in iOS 2.0 and later.

Declared In

`CGPath.h`

CGPathCreateCopy

Creates an immutable copy of a graphics path.

```
CGPathRef CGPathCreateCopy (
    CGPathRef path
);
```

Parameters

path

The path to copy.

Return Value

A new, immutable copy of the specified path. You are responsible for releasing this object.

Availability

Available in iOS 2.0 and later.

Related Sample Code

`aurioTouch`

`oalTouch`

Declared In

`CGPath.h`

CGPathCreateMutable

Creates a mutable graphics path.

```
CGMutablePathRef CGPathCreateMutable (
    void
);
```

Return Value

A new mutable path. You are responsible for releasing this object.

Availability

Available in iOS 2.0 and later.

Related Sample Code

Breadcrumb

Formulaic

MoveMe

oalTouch

WeatherMap

Declared In

CGPath.h

CGPathCreateMutableCopy

Creates a mutable copy of an existing graphics path.

```
CGMutablePathRef CGPathCreateMutableCopy (
    CGPathRef path
);
```

Parameters

path

The path to copy.

Return Value

A new, mutable, copy of the specified path. You are responsible for releasing this object.

Discussion

You can modify a mutable graphics path by calling the various CGPath geometry functions, such as [CGPathAddArc](#) (page 231), [CGPathAddLineToPoint](#) (page 235), and [CGPathMoveToPoint](#) (page 246).

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathEqualToPath

Indicates whether two graphics paths are equivalent.

```
bool CGPathEqualToPath (
    CGPathRef path1,
    CGPathRef path2
);
```

Parameters

path1

The first path being compared.

path2

The second path being compared.

Return Value

A Boolean value that indicates whether or not the two specified paths contain the same sequence of path elements. If the paths are not the same, returns `false`.

Availability

Available in iOS 2.0 and later.

Declared In

`CGPath.h`

CGPathGetBoundingBox

Returns the bounding box containing all points in a graphics path.

```
CGRect CGPathGetBoundingBox (  
    CGPathRef path  
);
```

Parameters

path

The graphics path to evaluate.

Return Value

A rectangle that represents the bounding box of the specified path. If the path is empty, this function returns [CGRectNull](#) (page 392).

Discussion

The bounding box is the smallest rectangle completely enclosing all points in the path, including control points for Bézier and quadratic curves.

Availability

Available in iOS 2.0 and later.

See Also

[CGPathGetPathBoundingBox](#) (page 244)

Declared In

`CGPath.h`

CGPathGetCurrentPoint

Returns the current point in a graphics path.

```
CGPoint CGPathGetCurrentPoint (  
    CGPathRef path  
);
```

Parameters

path

The path to evaluate.

Return Value

The current point in the specified path.

Discussion

If the path is empty—that is, if it has no elements—this function returns [CGPointZero](#) (page 392) (see *CGGeometry Reference*). To determine whether a path is empty, use [CGPathIsEmpty](#) (page 245).

Availability

Available in iOS 2.0 and later.

Declared In

`CGPath.h`

CGPathGetPathBoundingBox

Returns the bounding box of a graphics path.

```
CGRect CGPathGetPathBoundingBox (  
    CGPathRef path  
);
```

Parameters

path

The graphics path to evaluate.

Return Value

A rectangle that represents the path bounding box of the specified path. If the path is empty, this function returns [CGRectNull](#) (page 392).

Discussion

The path bounding box is the smallest rectangle completely enclosing all points in the path but not including control points for Bézier and quadratic curves.

Availability

Available in iOS 4.0 and later.

See Also

[CGPathGetBoundingBox](#) (page 243)

Declared In

`CGPath.h`

CGPathGetTypeID

Returns the Core Foundation type identifier for Quartz graphics paths.

```
CTypeID CGPathGetTypeID (  
    void  
);
```

Return Value

The Core Foundation identifier for the opaque type [CGPathRef](#) (page 248).

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathIsEmpty

Indicates whether or not a graphics path is empty.

```
bool CGPathIsEmpty (  
    CGPathRef path  
);
```

Parameters*path*

The path to evaluate.

Return Value

A Boolean value that indicates whether the specified path is empty.

Discussion

An empty path contains no elements.

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathIsRect

Indicates whether or not a graphics path represents a rectangle.

```
bool CGPathIsRect (  
    CGPathRef path,  
    CGRect *rect  
);
```

Parameters*path*

The path to evaluate.

rect

On input, a pointer to an uninitialized rectangle. If the specified path represents a rectangle, on return contains a copy of the rectangle.

Return Value

A Boolean value that indicates whether the specified path represents a rectangle. If the path represents a rectangle, returns `true`.

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathMoveToPoint

Starts a new subpath at a specified location in a mutable graphics path.

```
void CGPathMoveToPoint (
    CGMutablePathRef path,
    const CGAffineTransform *m,
    CGFloat x,
    CGFloat y
);
```

Parameters

path

The mutable path to change.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the point before changing the path.

x

The x-coordinate of the new location.

y

The y-coordinate of the new location.

Discussion

This function ends the subpath already in progress (if any) and starts a new subpath, initializing the starting point and the current point to the specified location (x,y) after an optional transformation.

Availability

Available in iOS 2.0 and later.

Related Sample Code

[aurioTouch](#)

[Breadcrumb](#)

[Formulaic](#)

[MoveMe](#)

[WeatherMap](#)

Declared In

`CGPath.h`

CGPathRelease

Decrements the retain count of a graphics path.

```
void CGPathRelease (
    CGPathRef path
);
```

Parameters

path

The graphics path to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `path` parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch

Breadcrumb

MoveMe

oalTouch

WeatherMap

Declared In

CGPath.h

CGPathRetain

Increments the retain count of a graphics path.

```
CGPathRef CGPathRetain (  
    CGPathRef path  
);
```

Parameters

path

The graphics path to retain.

Return Value

The same path you passed in as the *path* parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the *path* parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

Callbacks

CGPathApplierFunction

Defines a callback function that can view an element in a graphics path.

```
typedef void (*CGPathApplierFunction) (  
    void *info,  
    const CGPathElement *element  
);
```

If you name your function `MyCGPathApplierFunc`, you would declare it like this:

```
void MyCGPathApplierFunc (  
    void *info,  
    const CGPathElement *element  
);
```

Discussion

See also [CGPathApply](#) (page 239).

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

Data Types

CGPathRef

An opaque type that represents an immutable graphics path.

```
typedef const struct CGPath *CGPathRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGMutablePathRef

An opaque type that represents a mutable graphics path.

```
typedef struct CGPath *CGMutablePathRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

CGPathElement

A data structure that provides information about a path element.


```

struct CGPathElement {
    CGPathElementType type;
    CGPoint * points;
};
typedef struct CGPathElement CGPathElement;

```

Fields

type

An element type (or operation).

points

An array of one or more points that serve as arguments.

Availability

Available in iOS 2.0 and later.

Declared In

CGPath.h

Constants

CGPathDrawingMode

Options for rendering a path.

```

enum CGPathDrawingMode {
    kCGPathFill,
    kCGPathEOFill,
    kCGPathStroke,
    kCGPathFillStroke,
    kCGPathEOFillStroke
};
typedef enum CGPathDrawingMode CGPathDrawingMode;

```

Constants

kCGPathFill

Render the area contained within the path using the non-zero winding number rule.

Available in iOS 2.0 and later.

Declared in CGContext.h.

kCGPathEOFill

Render the area within the path using the even-odd rule.

Available in iOS 2.0 and later.

Declared in CGContext.h.

kCGPathStroke

Render a line along the path.

Available in iOS 2.0 and later.

Declared in CGContext.h.

`kCGPathFillStroke`

First fill and then stroke the path, using the nonzero winding number rule.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

`kCGPathEOFillStroke`

First fill and then stroke the path, using the even-odd rule.

Available in iOS 2.0 and later.

Declared in `CGContext.h`.

Discussion

You can pass a path drawing mode constant to the function `CGContextDrawPath` (page 79) to specify how Quartz should paint a graphics context's current path.

CGPathElementType

The type of element found in a path.

```
enum CGPathElementType {
    kCGPathElementMoveToPoint,
    kCGPathElementAddLineToPoint,
    kCGPathElementAddQuadCurveToPoint,
    kCGPathElementAddCurveToPoint,
    kCGPathElementCloseSubpath
};
typedef enum CGPathElementType CGPathElementType;
```

Constants

`kCGPathElementMoveToPoint`

The path element that starts a new subpath. See the function `CGPathMoveToPoint` (page 246).

Available in iOS 2.0 and later.

Declared in `CGPath.h`.

`kCGPathElementAddLineToPoint`

The path element that adds a line from the current point to the specified point. See the function `CGPathAddLineToPoint` (page 235).

Available in iOS 2.0 and later.

Declared in `CGPath.h`.

`kCGPathElementAddQuadCurveToPoint`

The path element that adds a quadratic curve from the current point to the specified point. See the function `CGPathAddQuadCurveToPoint` (page 237).

Available in iOS 2.0 and later.

Declared in `CGPath.h`.

`kCGPathElementAddCurveToPoint`

The path element that adds a cubic curve from the current point to the specified point. See the function `CGPathAddCurveToPoint` (page 233).

Available in iOS 2.0 and later.

Declared in `CGPath.h`.

`kCGPathElementCloseSubpath`

The path element that closes and completes a subpath. See the function [CGPathCloseSubpath](#) (page 240).

Available in iOS 2.0 and later.

Declared in `CGPath.h`.

Discussion

For more information about paths, see [CGPathRef](#) (page 248).

CGPattern Reference

Derived From:	CType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPattern.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPatternRef` opaque type represents a pattern that you can use to stroke along or fill in a graphics path. Quartz tiles the pattern cell for you, based on parameters you specify when you call [CGPatternCreate](#) (page 254).

To create a dashed line, see [CGContextSetLineDash](#) (page 111) in *CGContext Reference*.

Functions by Task

Creating a Pattern

[CGPatternCreate](#) (page 254)
Creates a pattern object.

Getting the CType ID

[CGPatternGetTypeID](#) (page 255)
Returns the type identifier for Quartz patterns.

Retaining and Releasing a Pattern

[CGPatternRetain](#) (page 256)
Increments the retain count of a Quartz pattern.

[CGPatternRelease](#) (page 255)
Decrements the retain count of a Quartz pattern.

Functions

CGPatternCreate

Creates a pattern object.

```
CGPatternRef CGPatternCreate (
    void *info,
    CGRect bounds,
    CGAffineTransform matrix,
    CGFloat xStep,
    CGFloat yStep,
    CGPatternTiling tiling,
    bool isColored,
    const CGPatternCallbacks *callbacks
);
```

Parameters

info

A pointer to private storage used by your pattern drawing function, or `NULL`. For more information, see the discussion below.

bounds

The bounding box of the pattern cell, specified in pattern space. (Pattern space is an abstract space that maps to the default user space by the transformation matrix you specify with the `matrix` parameter.) The drawing done in your pattern drawing function is clipped to this rectangle.

matrix

A matrix that represents a transform from pattern space to the default user space of the context in which the pattern is used. If no transform is needed, pass the identity matrix.

xStep

The horizontal displacement between cells, specified in pattern space. For no additional horizontal space between cells (so that each pattern cell abuts the previous pattern cell in the horizontal direction), pass the width of the pattern cell.

yStep

The vertical displacement between cells, specified in pattern space. For no additional vertical space between cells (so that each pattern cell abuts the previous pattern cell in the vertical direction), pass the height of the pattern cell.

tiling

A `CGPatternTiling` constant that specifies the desired tiling method. For more information about tiling methods, see “[Tiling Patterns](#)” (page 259).

isColored

If you want to draw your pattern using its own intrinsic color, pass `true`. If you want to draw an uncolored (or masking) pattern that uses the fill or stroke color in the graphics state, pass `false`.

callbacks

A pointer to a pattern callback function table—your pattern drawing function is an entry in this table. See [CGPatternCallbacks](#) (page 258) for more information about callback function tables for patterns.

Return Value

A new Quartz pattern. You are responsible for releasing this object using [CGPatternRelease](#) (page 255).

Discussion

Quartz calls your drawing function at the appropriate time to draw the pattern cell. A pattern cell must be invariant—that is, the pattern cell should be drawn exactly the same way each time the drawing function is called.

The appearance of a pattern cell is unaffected by changes in the graphics state of the context in which the pattern is used.

See [CGPatternDrawPatternCallback](#) (page 256) for more information about pattern drawing functions.

Availability**Related Sample Code**

QuartzDemo

Declared In

CGPattern.h

CGPatternGetTypeID

Returns the type identifier for Quartz patterns.

```
CTypeID CGPatternGetTypeID (
    void
);
```

Return Value

The identifier for the opaque type [CGPatternRef](#) (page 258).

Availability

Available in iOS 2.0 and later.

Declared In

CGPattern.h

CGPatternRelease

Decrements the retain count of a Quartz pattern.

```
void CGPatternRelease (
    CGPatternRef pattern
);
```

Parameters

pattern

The pattern to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the *pattern* parameter is `NULL`.

Availability**Related Sample Code**

QuartzDemo

Declared In

CGPattern.h

CGPatternRetain

Increments the retain count of a Quartz pattern.

```
CGPatternRef CGPatternRetain (  
    CGPatternRef pattern  
);
```

Parameters*pattern*

The pattern to retain.

Return Value

The same pattern you passed in as the *pattern* parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the *pattern* parameter is `NULL`.

Availability**Declared In**

CGPattern.h

Callbacks

CGPatternDrawPatternCallback

Draws a pattern cell.

```
typedef void (*CGPatternDrawPatternCallback) (  
    void * info,  
    CGContextRef context  
);
```

If you name your function `MyDrawPattern`, you would declare it like this:

```
void MyDrawPattern (  
    void * info,  
    CGContextRef context  
);
```

Parameters*info*

A generic pointer to private data associated with the pattern. This is the same pointer you supplied to [CGPatternCreate](#) (page 254).

context

The graphics context for drawing the pattern cell.

Discussion

When a pattern is used to stroke or fill a graphics path, Quartz calls your custom drawing function at the appropriate time to draw the pattern cell. The cell should be drawn exactly the same way each time the drawing function is called.

In a drawing function associated with an uncolored pattern, you should not attempt to set a stroke or fill color or color space—if you do so, the result is undefined.

To learn how to associate your drawing function with a Quartz pattern, see [CGPatternCreate](#) (page 254) and [CGPatternCallbacks](#) (page 258).

Availability

Available in iOS 2.0 and later.

Declared In

CGPattern.h

CGPatternReleaseInfoCallback

Release private data or resources associated with the pattern.

```
typedef void (*CGPatternReleaseInfoCallback) (
    void * info
);
```

If you name your function `MyCGPatternReleaseInfo`, you would declare it like this:

```
void MyCGPatternReleaseInfo (
    void * info
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to [CGPatternCreate](#) (page 254).

Discussion

Quartz calls your release function when it frees your pattern object.

To learn how to associate your release function with a Quartz pattern, see [CGPatternCreate](#) (page 254) and [CGPatternCallbacks](#) (page 258).

Availability

Available in iOS 2.0 and later.

Declared In

CGPattern.h

Data Types

CGPatternRef

An opaque type that represents a pattern.

```
typedef struct CGPattern * CGPatternRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPattern.h

CGPatternCallbacks

A structure that holds a version and two callback functions for drawing a custom pattern.

```
struct CGPatternCallbacks {  
    unsigned int version;  
    CGPatternDrawPatternCallback drawPattern;  
    CGPatternReleaseInfoCallback releaseInfo;  
};  
typedef struct CGPatternCallbacks CGPatternCallbacks;
```

Fields

version

The version of the structure passed in as a parameter to the [CGPatternCreate](#) (page 254). For this version of the structure, you should set this value to zero.

drawPattern

A pointer to a custom function that draws the pattern. For information about this callback function, see [CGPatternDrawPatternCallback](#) (page 256).

releaseInfo

An optional pointer to a custom function that's invoked when the pattern is released. [CGPatternReleaseInfoCallback](#) (page 257).

Discussion

You supply a `CGPatternCallbacks` structure to the function [CGPatternCreate](#) (page 254) to create a data provider for direct access. The functions specified by the `CGPatternCallbacks` structure are responsible for drawing the pattern and for handling the pattern's memory management.

Availability

Available in iOS 2.0 and later.

Declared In

CGPattern.h

Constants

Tiling Patterns

Different methods for rendering a tiled pattern.

```
enum CGPatternTiling {  
    kCGPatternTilingNoDistortion,  
    kCGPatternTilingConstantSpacingMinimalDistortion,  
    kCGPatternTilingConstantSpacing  
};  
typedef enum CGPatternTiling CGPatternTiling;
```

Constants

`kCGPatternTilingNoDistortion`

The pattern cell is not distorted when painted. The spacing between pattern cells may vary by as much as 1 device pixel.

Available in iOS 2.0 and later.

Declared in `CGPattern.h`.

`kCGPatternTilingConstantSpacingMinimalDistortion`

Pattern cells are spaced consistently. The pattern cell may be distorted by as much as 1 device pixel when the pattern is painted.

Available in iOS 2.0 and later.

Declared in `CGPattern.h`.

`kCGPatternTilingConstantSpacing`

Pattern cells are spaced consistently, as with

`kCGPatternTilingConstantSpacingMinimalDistortion`. The pattern cell may be distorted additionally to permit a more efficient implementation.

Available in iOS 2.0 and later.

Declared in `CGPattern.h`.

Declared In

`CGPattern.h`

CGPDFArray Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFArray.h
Companion guide	Quartz 2D Programming Guide

Overview

The CGPDFArray header file defines an opaque type that encapsulates a PDF array. A PDF array represents an array structure in a PDF document. PDF arrays may be heterogeneous—that is, they may contain any other PDF objects, including PDF strings, PDF dictionaries, and other PDF arrays.

Many CGPDFArray functions to retrieve values from a PDF array take the form:

```
bool CGPDFArrayGet<DataType> (
    CGPDFArrayRef array,
    size_t index,
    <DataType>Ref *value
);
```

These functions test the data type of the object at the specified index. If the object is not of the expected type, the function returns `false`. If the object is of the expected type, the function returns `true`, and the object is passed back in the `value` parameter.

This opaque type is not derived from CType and therefore there are no functions for retaining and releasing it. CGPDFArray objects exist only as constituent parts of a CGPDFDocument object, and they are managed by their container.

Functions

CGPDFArrayGetArray

Returns whether an object at a given index in a PDF array is another PDF array and, if so, retrieves that array.

```
bool CGPDFArrayGetArray (
    CGPDFArrayRef array,
    size_t index,
    CGPDFArrayRef *value
);
```

Parameters*array*

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (0 to N-1, where N is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF array. If the value at the specified index is a PDF array, then on return that array, otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF array at the specified index, otherwise `false`.

Availability**Declared In**

CGPDFArray.h

CGPDFArrayGetBoolean

Returns whether an object at a given index in a PDF array is a PDF Boolean and, if so, retrieves that Boolean.

```
bool CGPDFArrayGetBoolean (
    CGPDFArrayRef array,
    size_t index,
    CGPDFBoolean *value
);
```

Parameters*array*

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of array (0 to N-1, where N is the count of array), the behavior is undefined.

value

On input, a pointer to a PDF Boolean. If the value at the specified index is a PDF Boolean, then on return that Boolean, otherwise the value is undefined.

Return Value

Returns `true` if there is a PDF Boolean at the specified index, otherwise `false`.

Availability**Declared In**

CGPDFArray.h

CGPDFArrayGetCount

Returns the number of items in a PDF array.

```
size_t CGPDFArrayGetCount (  
    CGPDFArrayRef array  
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

Return Value

Returns the number of items in the array.

Availability

Declared In

CGPDFArray.h

CGPDFArrayGetDictionary

Returns whether an object at a given index in a PDF array is a PDF dictionary and, if so, retrieves that dictionary.

```
bool CGPDFArrayGetDictionary (  
    CGPDFArrayRef array,  
    size_t index,  
    CGPDFDictionaryRef *value  
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (0 to N-1, where N is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF dictionary. If the value at the specified index is a PDF dictionary, then on return that dictionary, otherwise the value is undefined.

Return Value

Returns `true` if there is a PDF dictionary at the specified index, otherwise `false`.

Availability

Declared In

CGPDFArray.h

CGPDFArrayGetInteger

Returns whether an object at a given index in a PDF array is a PDF integer and, if so, retrieves that object.

```
bool CGPDFArrayGetInteger (
    CGPDFArrayRef array,
    size_t index,
    CGPDFInteger *value
);
```

Parameters*array*

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (0 to N-1, where N is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF integer. If the value at the specified index is a PDF integer value, then on return contains that value, otherwise the value is undefined.

Return Value

Returns `true` if there is a PDF integer at the specified index, otherwise `false`.

Availability**Declared In**

CGPDFArray.h

CGPDFArrayGetName

Returns whether an object at a given index in a PDF array is a PDF name reference (represented as a constant C string) and, if so, retrieves that name.

```
bool CGPDFArrayGetName (
    CGPDFArrayRef array,
    size_t index,
    const char **value
);
```

Parameters*array*

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (0 to N-1, where N is the count of the array), the behavior is undefined.

value

An uninitialized pointer to a constant C string. If the value at the specified index is a reference to a PDF name (represented by a constant C string) then upon return, contains that value; otherwise the value is undefined.

Return Value

Returns `true` if there is an array of characters at the specified index, otherwise `false`.

Availability**Declared In**

CGPDFArray.h

CGPDFArrayGetNull

Returns whether an object at a given index in a Quartz PDF array is a PDF null.

```
bool CGPDFArrayGetNull (
    CGPDFArrayRef array,
    size_t index
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (0 to N-1, where N is the count of the array), the behavior is undefined.

Return Value

Returns `true` if there is a PDF null at the specified index, otherwise `false`.

Availability**Declared In**

CGPDFArray.h

CGPDFArrayGetNumber

Returns whether an object at a given index in a PDF array is a PDF number and, if so, retrieves that object.

```
bool CGPDFArrayGetNumber (
    CGPDFArrayRef array,
    size_t index,
    CGPDFReal *value
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (0 to N-1, where N is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF number. If the value at the specified index is a PDF number, then on return contains that value, otherwise the value is undefined.

Return Value

Returns `true` if there is a PDF number at the specified index, otherwise `false`.

Availability**Declared In**

CGPDFArray.h

CGPDFArrayGetObject

Returns whether an object at a given index in a PDF array is a PDF object and, if so, retrieves that object.

```

bool CGPDFArrayGetObject (
    CGPDFArrayRef array,
    size_t index,
    CGPDFObjectRef *value
);

```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (0 to N-1, where N is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF object. If the value at the specified index is a PDF object, then on return contains that object, otherwise the value is undefined.

Return Value

Returns `true` if there is a PDF object at the specified index, otherwise `false`.

Availability

Declared In

CGPDFArray.h

CGPDFArrayGetStream

Returns whether an object at a given index in a PDF array is a PDF stream and, if so, retrieves that stream.

```

bool CGPDFArrayGetStream (
    CGPDFArrayRef array,
    size_t index,
    CGPDFStreamRef *value
);

```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (0 to N-1, where N is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF stream. If the value at the specified index is a PDF stream, then on return that stream, otherwise the value is undefined.

Return Value

Returns `true` if there is a PDF stream at the specified index, otherwise `false`.

Availability**Declared In**

CGPDFArray.h

CGPDFArrayGetString

Returns whether an object at a given index in a PDF array is a PDF string and, if so, retrieves that string.

```
bool CGPDFArrayGetString (
    CGPDFArrayRef array,
    size_t index,
    CGPDFStringRef *value
);
```

Parameters*array*

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (0 to N-1, where N is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF string. If the value at the specified index is a PDF string, then on return that string, otherwise the value is undefined.

Return Value

Returns `true` if there is a PDF stream at the specified index, otherwise `false`.

Availability**Declared In**

CGPDFArray.h

Data Types

CGPDFArrayRef

An opaque type that encapsulates a PDF array.

```
typedef struct CGPDFArray *CGPDFArrayRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFArray.h

CGPDFContentStream Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFContentStream.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFContentStreamRef` opaque type provides access to the data that describes the appearance of a PDF page. A `CGPDFContentStream` object represents one or more PDF content streams for a page and their associated resource dictionaries. A PDF content stream is a sequential set of instructions that specifies how to paint items on a PDF page. A resource dictionary contains information needed by the content stream in order to decode the sequential instructions of the content stream.

`CGPDFContentStream` functions can retrieve both the content streams and the resource dictionaries associated with a PDF page.

This opaque type is not derived from `CType` and therefore there are no functions for retaining and releasing it.

Functions by Task

Creating a PDF Content Stream Object

[`CGPDFContentStreamCreateWithPage`](#) (page 270)

Creates a content stream object from a PDF page object.

[`CGPDFContentStreamCreateWithStream`](#) (page 270)

Creates a PDF content stream object from an existing PDF content stream object.

Getting Data from a PDF Content Stream Object

[`CGPDFContentStreamGetStreams`](#) (page 271)

Gets the array of PDF content streams contained in a PDF content stream object.

[`CGPDFContentStreamGetResource`](#) (page 271)

Gets the specified resource from a PDF content stream object.

Retaining and Releasing a PDF Content Stream Object

[CGPDFContentStreamRetain](#) (page 272)

Increments the retain count of a PDF content stream object.

[CGPDFContentStreamRelease](#) (page 272)

Decrements the retain count of a PDF content stream object.

Functions

CGPDFContentStreamCreateWithPage

Creates a content stream object from a PDF page object.

```
CGPDFContentStreamRef CGPDFContentStreamCreateWithPage (
    CGPDFPageRef page
);
```

Parameters

page

A PDF page object.

Return Value

A new `CGPDFContentStream` object. You are responsible for releasing this object by calling the function `CGPDFContentStreamRelease`.

Discussion

A `CGPDFContentStream` object can contain more than one PDF content stream. To retrieve an array of the PDF content streams in the object, call the function [CGPDFContentStreamGetStreams](#) (page 271). To obtain the resources associated with a `CGPDFContentStream` object, call the function [CGPDFContentStreamGetResource](#) (page 271).

Availability

Declared In

`CGPDFContentStream.h`

CGPDFContentStreamCreateWithStream

Creates a PDF content stream object from an existing PDF content stream object.

```
CGPDFContentStreamRef CGPDFContentStreamCreateWithStream (
    CGPDFStreamRef stream,
    CGPDFDictionaryRef streamResources,
    CGPDFContentStreamRef parent
);
```

Parameters

stream

The PDF stream you want to create a content stream from.

streamResources

A PDF dictionary that contains the resources associated with the stream you want to retrieve.

parent

The content stream of the page on which *stream* appears. Supply the *parent* parameter when you create a content stream that's used within a page.

Return Value

A CGPDFContentStream object created from the *stream* parameter. You are responsible for releasing this object by calling the function [CGPDFContentStreamRelease](#) (page 272).

Discussion

You can use this function to get access to the contents of a form, pattern, Type3 font, or any PDF stream.

Availability

Declared In

CGPDFContentStream.h

CGPDFContentStreamGetResource

Gets the specified resource from a PDF content stream object.

```
CGPDFObjectRef CGPDFContentStreamGetResource (
    CGPDFContentStreamRef cs,
    const char *category,
    const char *name
);
```

Parameters

cs

A CGPDFContentStream object.

category

A string that specifies the category of the resource you want to obtain.

name

A string that specifies the name of the resource you want to obtain.

Return Value

The resource dictionary.

Discussion

You can use this function to obtain resources used by the content stream, such as forms, patterns, color spaces, and fonts.

Availability

Declared In

CGPDFContentStream.h

CGPDFContentStreamGetStreams

Gets the array of PDF content streams contained in a PDF content stream object.

```
CFArrayRef CGPDFContentStreamGetStreams (
    CGPDFContentStreamRef cs
);
```

Parameters*cs*

A CGPDFContentStream object.

Return Value

The array of PDF content streams that make up the content stream object represented by the *cs* parameter.

Availability**Declared In**

CGPDFContentStream.h

CGPDFContentStreamRelease

Decrements the retain count of a PDF content stream object.

```
void CGPDFContentStreamRelease (
    CGPDFContentStreamRef cs
);
```

Parameters*cs*

A PDF content stream.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFContentStream.h

CGPDFContentStreamRetain

Increments the retain count of a PDF content stream object.

```
CGPDFContentStreamRef CGPDFContentStreamRetain (
    CGPDFContentStreamRef cs
);
```

Parameters*cs*

A PDF content stream.

Return Value

The same PDF content stream you passed in as the *cs* parameter.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFContentStream.h

Data Types

CGPDFContentStreamRef

An opaque type that provides access to the data that describes the appearance of a PDF page.

```
typedef struct CGPDFContentStream *CGPDFContentStreamRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFContentStream.h

CGPDFContext Reference

Derived From:	CGContextRef (page 132)
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFContext.h
Companion guide	Quartz 2D Programming Guide

Overview

The CGPDFContext header file defines functions that create and get information about a Quartz PDF context. A CGPDFContext object is a type of [CGContextRef](#) (page 132) that is used for drawing PDF content. The functions in this reference operate only on Quartz PDF graphics contexts created using the functions [CGPDFContextCreate](#) (page 278) or [CGPDFContextCreateWithURL](#) (page 279).

When you draw to the PDF context using CGContext functions the drawing operations are recorded in PDF format. The PDF commands that represent the drawing are written to the destination specified when you create the PDF graphics context.

Functions by Task

Creating a Context

[CGPDFContextCreate](#) (page 278)

Creates a PDF graphics context.

[CGPDFContextCreateWithURL](#) (page 279)

Creates a URL-based PDF graphics context.

Beginning and Ending Pages

[CGPDFContextBeginPage](#) (page 277)

Begins a new page in a PDF graphics context.

[CGPDFContextEndPage](#) (page 279)

Ends the current page in the PDF graphics context.

Working with Destinations

[CGPDFContextAddDestinationAtPoint](#) (page 276)

Sets a destination to jump to when a point in the current page of a PDF graphics context is clicked.

[CGPDFContextSetDestinationForRect](#) (page 280)

Sets a destination to jump to when a rectangle in the current PDF page is clicked.

[CGPDFContextSetURLForRect](#) (page 280)

Sets the URL associated with a rectangle in a PDF graphics context.

Working with Metadata

[CGPDFContextAddDocumentMetadata](#) (page 277)

Associates custom metadata with the PDF document.

Closing a PDF Context

[CGPDFContextClose](#) (page 277)

Closes a PDF document.

Functions

CGPDFContextAddDestinationAtPoint

Sets a destination to jump to when a point in the current page of a PDF graphics context is clicked.

```
void CGPDFContextAddDestinationAtPoint (
    CGContextRef context,
    CFStringRef name,
    CGPoint point
);
```

Parameters

context

A PDF graphics context.

name

A destination name.

point

A location in the current page of the PDF graphics context.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFContext.h

CGPDFContextAddDocumentMetadata

Associates custom metadata with the PDF document.

```
void CGPDFContextAddDocumentMetadata(  
    CGContextRef context,  
    CFDataRef metadata  
);
```

Parameters

context

A PDF graphics context.

metadata

A stream of XML data that is formatted according to the Extensible Metadata Platform, as described in section 10.2.2., “Metadata Streams”, of the PDF 1.7 specification.

Availability

Available in iOS 4.0 and later.

Declared In

CGPDFContext.h

CGPDFContextBeginPage

Begins a new page in a PDF graphics context.

```
void CGPDFContextBeginPage (  
    CGContextRef context,  
    CFDictionaryRef pageInfo  
);
```

Parameters

context

A PDF graphics context.

pageInfo

A dictionary that contains key-value pairs that define the page properties.

Discussion

You must call the function [CGPDFContextEndPage](#) (page 279) to signal the end of the page.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFContext.h

CGPDFContextClose

Closes a PDF document.

```
void CGPDFContextClose(
    CGContextRef context
);
```

Parameters*context*

A PDF graphics context.

Discussion

After closing the context, all pending data is written to the context destination, and the PDF file is completed. No additional data can be written to the destination context after the PDF document is closed.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFContext.h

CGPDFContextCreate

Creates a PDF graphics context.

```
CGContextRef CGPDFContextCreate (
    CGDataConsumerRef consumer,
    const CGRect *mediaBox,
    CFDictionaryRef auxiliaryInfo
);
```

Parameters*consumer*

The data consumer to receive the PDF output data.

mediaBox

A pointer to a rectangle that defines the size and location of the PDF page, or `NULL`. The origin of the rectangle should typically be `(0, 0)`. Quartz uses this rectangle as the default bounds of the page's media box. If you pass `NULL`, Quartz uses a default page size of 8.5 by 11 inches (612 by 792 points).

auxiliaryInfo

A dictionary that specifies any additional information to be used by the PDF context when generating the PDF file, or `NULL`. The dictionary is retained by the new context, so on return you may safely release it. See [“Auxiliary Dictionary Keys”](#) (page 281) for keys you can include in the dictionary.

Return Value

A new PDF context, or `NULL` if the context cannot be created. You are responsible for releasing this object using [CGContextRelease](#) (page 93).

Discussion

This function creates a PDF drawing environment to your specifications. When you draw into the new context, Quartz renders your drawing as a sequence of PDF drawing commands that are passed to the data consumer object.

Availability**Declared In**

CGPDFContext.h

CGPDFContextCreateWithURL

Creates a URL-based PDF graphics context.

```
CGContextRef CGPDFContextCreateWithURL (
    CFURLRef url,
    const CGRect *mediaBox,
    CFDictionaryRef auxiliaryInfo
);
```

Parameters

url

A Core Foundation URL that specifies where you want to place the resulting PDF file.

mediaBox

A rectangle that specifies the bounds of the PDF. The origin of the rectangle should typically be (0,0). The `CGPDFContextCreateWithURL` function uses this rectangle as the default page media bounding box. If you pass `NULL`, `CGPDFContextCreateWithURL` uses a default page size of 8.5 by 11 inches (612 by 792 points).

auxiliaryInfo

A dictionary that specifies any additional information to be used by the PDF context when generating the PDF file, or `NULL`. The dictionary is retained by the new context, so on return you may safely release it.

Return Value

A new PDF context, or `NULL` if a context could not be created. You are responsible for releasing this object using [CGContextRelease](#) (page 93).

Discussion

When you call this function, Quartz creates a PDF drawing environment—that is, a graphics context—to your specifications. When you draw into the resulting context, Quartz renders your drawing as a series of PDF drawing commands stored in the specified location.

Availability

Declared In

`CGPDFContext.h`

CGPDFContextEndPage

Ends the current page in the PDF graphics context.

```
void CGPDFContextEndPage (
    CGContextRef context
);
```

Parameters

context

A PDF graphics context.

Discussion

You can call `CGPDFContextEndPage` only after you call the function [CGPDFContextBeginPage](#) (page 277).

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFContext.h

CGPDFContextSetDestinationForRect

Sets a destination to jump to when a rectangle in the current PDF page is clicked.

```
void CGPDFContextSetDestinationForRect (
    CGContextRef context,
    CFStringRef name,
    CGRect rect
);
```

Parameters*context*

A PDF graphics context.

name

A destination name.

rect

A rectangle that specifies an area of the current page of a PDF graphics context. The rectangle is specified in default user space (not device space).

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFContext.h

CGPDFContextSetURLForRect

Sets the URL associated with a rectangle in a PDF graphics context.

```
void CGPDFContextSetURLForRect (
    CGContextRef context,
    CFURLRef url,
    CGRect rect
);
```

Parameters*context*

A PDF graphics context.

url

A CFURL object that specifies the destination of the contents associated with the rectangle.

rect

A rectangle specified in default user space (not device space).

Availability**Declared In**

CGPDFContext.h

Constants

Auxiliary Dictionary Keys

Keys that used to set up a PDF context.

```
CFStringRef kCGPDFContextAuthor;
CFStringRef kCGPDFContextCreator;
CFStringRef kCGPDFContextTitle;
CFStringRef kCGPDFContextOwnerPassword;
CFStringRef kCGPDFContextUserPassword;
CFStringRef kCGPDFContextAllowsPrinting;
CFStringRef kCGPDFContextAllowsCopying;
CFStringRef kCGPDFContextSubject;
CFStringRef kCGPDFContextKeywords;
CFStringRef kCGPDFContextEncryptionKeyLength;
```

Constants

`kCGPDFContextAuthor`

The corresponding value is a string that represents the name of the person who created the document. This key is optional.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextCreator`

The corresponding value is a string that represents the name of the application used to produce the document. This key is optional.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextTitle`

The corresponding value is a string that represents the title of the document. This key is optional.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextOwnerPassword`

The owner password of the PDF document. If this key is specified, the document is encrypted using the value as the owner password; otherwise, the document will not be encrypted. The value of this key must be a CFString object that can be represented in ASCII encoding. Only the first 32 bytes are used for the password. There is no default value for this key. If the value of this key cannot be represented in ASCII, the document is not created and the creation function returns `NULL`.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextUserPassword`

The user password of the PDF document. If the document is encrypted, then the value of this key will be the user password for the document. If not specified, the user password is the empty string. The value of this key must be a CFString object that can be represented in ASCII encoding; only the first 32 bytes will be used for the password. If the value of this key cannot be represented in ASCII, the document is not created and the creation function returns `NULL`.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

kCGPDFContextAllowsPrinting

Whether the document allows printing when unlocked with the user password. The value of this key must be a `CFBoolean` value. The default value of this key is `kCFBooleanTrue`.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

kCGPDFContextAllowsCopying

Whether the document allows copying when unlocked with the user password. The value of this key must be a `CFBoolean` object. The default value of this key is `kCFBooleanTrue`.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

kCGPDFContextSubject

The subject of a document. Optional; if present, the value of this key must be a `CFString` object.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

kCGPDFContextKeywords

The keywords for this document. This key is optional. If the value of this key is a `CFString` object, the `/Keywords` entry will be the specified string. If the value of this key is a `CFArray` object, then it must be an array of `CFString` objects. The `/Keywords` entry will, in this case, be the concatenation of the specified strings separated by commas (", "). In addition, an entry with the key `"/AAPL:Keywords"` is stored in the document information dictionary; its value is an array consisting of each of the specified strings. The value of this key must be in one of the above forms; otherwise, this key is ignored.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

kCGPDFContextEncryptionKeyLength

The encryption key length in bits; see Table 3.18 "Entries common to all encryption dictionaries", PDF Reference: Adobe PDF version 1.5 (4th ed.) for more information. Optional; if present, the value of this key must be a `CFNumber` object with value which is a multiple of 8 between 40 and 128, inclusive. If this key is absent or invalid, the encryption key length defaults to 40 bits.

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

Discussion

For more information about using these keys in a PDF context, see [CGPDFContextCreate](#) (page 278) and [CGPDFContextCreateWithURL](#) (page 279).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGPDFContext.h`

Box Dictionary Keys

Keys that specify various PDF boxes.

```
CFStringRef kCGPDFContextMediaBox  
CFStringRef kCGPDFContextCropBox  
CFStringRef kCGPDFContextBleedBox  
CFStringRef kCGPDFContextTrimBox  
CFStringRef kCGPDFContextArtBox
```

Constants

`kCGPDFContextMediaBox`

The media box for the document or for a given page. This key is optional. If present, the value of this key must be a `CFData` object that contains a `CGRect` (stored by value, not by reference).

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextCropBox`

The crop box for the document or for a given page. This key is optional. If present, the value of this key must be a `CFData` object that contains a `CGRect` (stored by value, not by reference).

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextBleedBox`

The bleed box for the document or for a given page. This key is optional. If present, the value of this key must be a `CFData` object that contains a `CGRect` (stored by value, not by reference).

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextTrimBox`

The trim box for the document or for a given page. This key is optional. If present, the value of this key must be a `CFData` object that contains a `CGRect` (stored by value, not by reference).

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextArtBox`

The art box for the document or for a given page. This key is optional. If present, the value of this key must be a `CFData` object that contains a `CGRect` (stored by value, not by reference).

Available in iOS 2.0 and later.

Declared in `CGPDFContext.h`.

Discussion

For more information about using these keys in a PDF context, see [CGPDFContextCreate](#) (page 278) and [CGPDFContextCreateWithURL](#) (page 279).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGPDFContext.h`

CGPDFDictionary Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFDictionary.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFDictionaryRef` opaque type encapsulates a PDF dictionary whose key-value pairs can specify any kind of PDF object, including another dictionary. Dictionary objects are the main building blocks of a PDF document. A key-value pair within a dictionary is called an entry. In a PDF dictionary, the key must be an array of characters. Within a given dictionary, the keys are unique—that is, no two keys in a single dictionary are equal (as determined by `strcmp`). The value associated with a key can be any kind of PDF object, including another dictionary. Dictionary objects are the main building blocks of a PDF document.

Many functions that retrieve values from a PDF dictionary take the form:

```
bool CGPDFDictionaryGet<DataType> (
    CGPDFDictionaryRef dictionary,
    const char *key,
    <DataType>Ref *value
);
```

These functions test whether there is an object associated with the specified key. If there is an object associated with the specified key, they test its data type. If there is no associated object, or if there is but it is not of the expected type, the function returns `false`. If there is an object associated with the specified key and it is of the expected type, the function returns `true` and the object is passed back in the `value` parameter.

This opaque type is not derived from `CType` and therefore there are no functions for retaining and releasing it. `CGPDFDictionary` objects exist only as constituent parts of a `CGPDFDocument` object, and they are managed by their container.

Functions by Task

Applying a Function to All Entries

[CGPDFDictionaryApplyFunction](#) (page 286)

Applies a function to each entry in a dictionary.

Getting Data from a Dictionary

[CGPDFDictionaryGetArray](#) (page 287)

Returns whether there is a PDF array associated with a specified key in a PDF dictionary and, if so, retrieves that array.

[CGPDFDictionaryGetBoolean](#) (page 288)

Returns whether there is a PDF Boolean value associated with a specified key in a PDF dictionary and, if so, retrieves the Boolean value.

[CGPDFDictionaryGetCount](#) (page 288)

Returns the number of entries in a PDF dictionary.

[CGPDFDictionaryGetDictionary](#) (page 288)

Returns whether there is another PDF dictionary associated with a specified key in a PDF dictionary and, if so, retrieves that dictionary.

[CGPDFDictionaryGetInteger](#) (page 289)

Returns whether there is a PDF integer associated with a specified key in a PDF dictionary and, if so, retrieves that integer.

[CGPDFDictionaryGetName](#) (page 290)

Returns whether an object with a specified key in a PDF dictionary is a PDF name reference (represented as a constant C string) and, if so, retrieves that name.

[CGPDFDictionaryGetNumber](#) (page 290)

Returns whether there is a PDF number associated with a specified key in a PDF dictionary and, if so, retrieves that number.

[CGPDFDictionaryGetObject](#) (page 291)

Returns whether there is a PDF object associated with a specified key in a PDF dictionary and, if so, retrieves that object.

[CGPDFDictionaryGetStream](#) (page 291)

Returns whether there is a PDF stream associated with a specified key in a PDF dictionary and, if so, retrieves that stream.

[CGPDFDictionaryGetString](#) (page 292)

Returns whether there is a PDF string associated with a specified key in a PDF dictionary and, if so, retrieves that string.

Functions

CGPDFDictionaryApplyFunction

Applies a function to each entry in a dictionary.

```
void CGPDFDictionaryApplyFunction (
    CGPDFDictionaryRef dict,
    CGPDFDictionaryApplierFunction function,
    void *info
);
```

Parameters*dictionary*

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

function

The function to apply to each entry in the dictionary.

info

A pointer to contextual information to pass to the function.

Discussion

This function enumerates all of the entries in the dictionary, calling the function once for each. The current key, its associated value, and the contextual information are passed to the function (see also [CGPDFDictionaryApplierFunction](#) (page 292)).

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetArray

Returns whether there is a PDF array associated with a specified key in a PDF dictionary and, if so, retrieves that array.

```
bool CGPDFDictionaryGetArray (
    CGPDFDictionaryRef dict,
    const char *key,
    CGPDFArrayRef *value
);
```

Parameters*dictionary*

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, an uninitialized pointer to a PDF array. If the value associated with the specified key is a PDF array, then on return contains that array; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF array associated with the specified key; otherwise, `false`.

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetBoolean

Returns whether there is a PDF Boolean value associated with a specified key in a PDF dictionary and, if so, retrieves the Boolean value.

```
bool CGPDFDictionaryGetBoolean (
    CGPDFDictionaryRef dict,
    const char *key,
    CGPDFBoolean *value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF Boolean value. If the value associated with the specified key is a PDF Boolean value, then on return contains that value; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF Boolean value associated with the specified key; otherwise, `false`.

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetCount

Returns the number of entries in a PDF dictionary.

```
size_t CGPDFDictionaryGetCount (
    CGPDFDictionaryRef dict
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

Return Value

Returns the number of entries in the dictionary.

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetDictionary

Returns whether there is another PDF dictionary associated with a specified key in a PDF dictionary and, if so, retrieves that dictionary.


```
bool CGPDFDictionaryGetDictionary (
    CGPDFDictionaryRef dict,
    const char *key,
    CGPDFDictionaryRef *value
);
```

Parameters*dictionary*

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF dictionary. If the value associated with the specified key is a PDF dictionary, then on return contains that dictionary; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF dictionary associated with the specified key; otherwise, `false`.

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetInteger

Returns whether there is a PDF integer associated with a specified key in a PDF dictionary and, if so, retrieves that integer.

```
bool CGPDFDictionaryGetInteger (
    CGPDFDictionaryRef dict,
    const char *key,
    CGPDFInteger *value
);
```

Parameters*dictionary*

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF integer. If the value associated with the specified key is a PDF integer, then on return contains that value; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF integer associated with the specified key; otherwise, `false`.

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetName

Returns whether an object with a specified key in a PDF dictionary is a PDF name reference (represented as a constant C string) and, if so, retrieves that name.

```
bool CGPDFDictionaryGetName (
    CGPDFDictionaryRef dict,
    const char *key,
    const char **value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF name reference, represented as a constant C string. If the value associated with the specified key is a reference to a PDF name, then on return, the variable points to the name; otherwise, the value is undefined.

Return Value

Returns `true` if there is a character array associated with the specified key; otherwise, `false`.

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetNumber

Returns whether there is a PDF number associated with a specified key in a PDF dictionary and, if so, retrieves that number.

```
bool CGPDFDictionaryGetNumber (
    CGPDFDictionaryRef dict,
    const char *key,
    CGPDFReal *value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF number. If the value associated with the specified key is a PDF number (real or integer), then on return contains that value; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF number associated with the specified key; otherwise, `false`.

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetObject

Returns whether there is a PDF object associated with a specified key in a PDF dictionary and, if so, retrieves that object.

```
bool CGPDFDictionaryGetObject (
    CGPDFDictionaryRef dict,
    const char *key,
    CGPDFObjectRef *value
);
```

Parameters*dictionary*

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF object. If the value associated with the specified key is a PDF object, then on return contains that object; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF object associated with the specified key; otherwise, `false`.

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetStream

Returns whether there is a PDF stream associated with a specified key in a PDF dictionary and, if so, retrieves that stream.

```
bool CGPDFDictionaryGetStream (
    CGPDFDictionaryRef dict,
    const char *key,
    CGPDFStreamRef *value
);
```

Parameters*dictionary*

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to be retrieved.

value

On input, a pointer to a PDF stream. If the value associated with the specified key is a PDF stream, then on return contains that stream; otherwise, the value is unspecified.

Return Value

Returns `true` if there is a PDF stream associated with the specified key; otherwise, `false`.

Availability**Declared In**

CGPDFDictionary.h

CGPDFDictionaryGetString

Returns whether there is a PDF string associated with a specified key in a PDF dictionary and, if so, retrieves that string.

```
bool CGPDFDictionaryGetString (
    CGPDFDictionaryRef dict,
    const char *key,
    CGPDFStringRef *value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF string. If the value associated with the specified key is a PDF string, then on return contains that string; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF string associated with the specified key; otherwise, `false`.

Availability**Declared In**

CGPDFDictionary.h

Callbacks

CGPDFDictionaryApplierFunction

Performs custom processing on a key-value pair from a PDF dictionary, using optional contextual information.

```
typedef void (*CGPDFDictionaryApplierFunction) (
    const char *key,
    CGPDFObjectRef value,
    void *info,
);
```

If you name your function `MyFunction`, you would declare it like this:

```
void MyFunction (
```

```
    const char *key,  
    CGPDFObjectRef object,  
    void *info  
);
```

Parameters*key*

The current key in the dictionary.

object

The value in the dictionary associated with the key.

info

The contextual information that you provided in the `info` parameter in [CGPDFDictionaryApplyFunction](#) (page 286).

Discussion

`CGPDFDictionaryApplierFunction` defines the callback for `CGPDFDictionaryApplyFunction`, that enumerates all of the entries in the dictionary, calling your custom applier function once for each entry. The current key, its associated value, and the contextual information are passed to your applier function using the `key`, `value`, and `info` parameters respectively.

Availability

Available in iOS 2.0 and later.

Declared In

`CGPDFDictionary.h`

Data Types

CGPDFDictionaryRef

An opaque type that encapsulates a PDF dictionary.

```
typedef struct CGPDFDictionary *CGPDFDictionaryRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

`CGPDFDictionary.h`

CGPDFDocument Reference

Derived From:	CType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFDocument.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFDocumentRef` opaque type encapsulates a document that contains PDF (Portable Document Format) drawing information. PDF provides an efficient format for cross-platform exchange of documents with rich content. PDF files can contain multiple pages of images and text. A PDF document object contains all the information relating to a PDF document, including its catalog and contents.

Note that PDF documents may be encrypted, and that some operations may be restricted until a valid password is supplied—see the functions listed in [“Managing Encryption”](#) (page 296). Quartz also supports decrypting encrypted documents.

Quartz can both display and generate files that are compliant with the PDF standard. When imaging PDF files, `CGPDFDocumentRef` is the basic type used to represent a PDF document.

Functions by Task

Creating PDF Document Objects

[CGPDFDocumentCreateWithProvider](#) (page 297)

Creates a Quartz PDF document using a data provider.

[CGPDFDocumentCreateWithURL](#) (page 298)

Creates a Quartz PDF document using data specified by a URL.

Retaining and Releasing PDF Documents

[CGPDFDocumentRelease](#) (page 303)

Decrements the retain count of a PDF document.

[CGPDFDocumentRetain](#) (page 303)

Increments the retain count of a Quartz PDF document.

Getting the CType ID for a PDF Document Object

[CGPDFDocumentGetTypeID](#) (page 301)

Returns the type identifier for Quartz PDF documents.

Getting Information About Quartz PDF Documents

[CGPDFDocumentGetCatalog](#) (page 298)

Returns the document catalog of a Quartz PDF document.

[CGPDFDocumentGetNumberOfPages](#) (page 300)

Returns the number of pages in a PDF document.

[CGPDFDocumentGetPage](#) (page 300)

Returns a page from a Quartz PDF document.

[CGPDFDocumentGetVersion](#) (page 301)

Returns the major and minor version numbers of a Quartz PDF document.

[CGPDFDocumentGetInfo](#) (page 299)

Gets the information dictionary for a PDF document.

[CGPDFDocumentGetID](#) (page 299)

Gets the file identifier for a PDF document.

Managing Encryption

[CGPDFDocumentAllowsCopying](#) (page 296)

Returns whether the specified PDF document allows copying.

[CGPDFDocumentAllowsPrinting](#) (page 297)

Returns whether a PDF document allows printing.

[CGPDFDocumentIsEncrypted](#) (page 302)

Returns whether the specified PDF file is encrypted.

[CGPDFDocumentIsUnlocked](#) (page 302)

Returns whether the specified PDF document is currently unlocked.

[CGPDFDocumentUnlockWithPassword](#) (page 303)

Unlocks an encrypted PDF document, if a valid password is supplied.

Functions

CGPDFDocumentAllowsCopying

Returns whether the specified PDF document allows copying.


```
bool CGPDFDocumentAllowsCopying (
    CGPDFDocumentRef document
);
```

Parameters

document

A PDF document.

Return Value

A Boolean that, if `true`, indicates that the document allows copying. If the value is `false`, the document does not allow copying.

Discussion

This function returns `true` if the specified PDF document allows copying. It returns `false` if the document is encrypted and the current password doesn't grant permission to perform copying.

Availability**Declared In**

CGPDFDocument.h

CGPDFDocumentAllowsPrinting

Returns whether a PDF document allows printing.

```
bool CGPDFDocumentAllowsPrinting (
    CGPDFDocumentRef document
);
```

Parameters

document

A PDF document.

Return Value

A Boolean that, if `true`, indicates that the document allows printing. If the value is `false`, the document does not allow printing.

Discussion

This function returns `true` if the specified PDF document allows printing. It returns `false` if the document is encrypted and the current password doesn't grant permission to perform printing.

Availability**Declared In**

CGPDFDocument.h

CGPDFDocumentCreateWithProvider

Creates a Quartz PDF document using a data provider.

```
CGPDFDocumentRef CGPDFDocumentCreateWithProvider (
    CGDataProviderRef provider
);
```

Parameters*provider*

A data provider that supplies the PDF document data.

Return Value

A new Quartz PDF document, or `NULL` if a document can not be created. You are responsible for releasing the object using [CGPDFDocumentRelease](#) (page 303).

Discussion

Distributing individual pages of a PDF document to separate threads is not supported. If you want to use threads, consider creating a separate document for each thread and operating on a block of pages per thread.

Availability**Declared In**

`CGPDFDocument.h`

CGPDFDocumentCreateWithURL

Creates a Quartz PDF document using data specified by a URL.

```
CGPDFDocumentRef CGPDFDocumentCreateWithURL (
    CFURLRef url
);
```

Parameters*url*

The URL address at which the PDF document data is located.

Return Value

A new Quartz PDF document, or `NULL` if a document could not be created. You are responsible for releasing the object using [CGPDFDocumentRelease](#) (page 303).

Discussion

Distributing individual pages of a PDF document to separate threads is not supported. If you want to use threads, consider creating a separate document for each thread and operating on a block of pages per thread.

Availability**Related Sample Code**

QuartzDemo

ZoomingPDFViewer

Declared In

`CGPDFDocument.h`

CGPDFDocumentGetCatalog

Returns the document catalog of a Quartz PDF document.

```
CGPDFDictionaryRef CGPDFDocumentGetCatalog (
    CGPDFDocumentRef document
);
```

Parameters

document

A PDF document.

Return Value

The document catalog of the specified document.

Discussion

The entries in a PDF document catalog recursively describe the contents of the PDF document. You can access the contents of a PDF document catalog by calling the function `CGPDFDocumentGetCatalog`. For information on accessing PDF metadata, see *Quartz 2D Programming Guide*.

Availability**Declared In**

`CGPDFDocument.h`

CGPDFDocumentGetID

Gets the file identifier for a PDF document.

```
CGPDFArrayRef CGPDFDocumentGetID (
    CGPDFDocumentRef document
);
```

Parameters

document

The document whose file identifier you want to obtain.

Return Value

Returns the file identifier for the document.

Discussion

A PDF file identifier is defined in the PDF specification as an array of two strings, the first of which is a permanent identifier that doesn't change even when the file is updated. The second string changes each time the file is updated. For more information, see *PDF Reference: Version 1.3 (Second Edition)*, Adobe Systems Incorporated.

Availability**Declared In**

`CGPDFDocument.h`

CGPDFDocumentGetInfo

Gets the information dictionary for a PDF document.

```
CGPDFDictionaryRef CGPDFDocumentGetInfo (
    CGPDFDocumentRef document
);
```

Parameters

document

The document whose dictionary you want to obtain.

Return Value

The information dictionary for the document.

Availability**Declared In**

CGPDFDocument.h

CGPDFDocumentGetNumberOfPages

Returns the number of pages in a PDF document.

```
size_t CGPDFDocumentGetNumberOfPages (
    CGPDFDocumentRef document
);
```

Parameters

document

The PDF document to examine.

Return Value

The total number of pages in the PDF document.

Availability**Declared In**

CGPDFDocument.h

CGPDFDocumentGetPage

Returns a page from a Quartz PDF document.

```
CGPDFPageRef CGPDFDocumentGetPage (
    CGPDFDocumentRef document,
    size_t pageNumber
);
```

Parameters

document

A PDF document.

pageNumber

The number of the page requested.

Return Value

Return the PDF page corresponding to the specified page number, or `NULL` if no such page exists in the document. Pages are numbered starting at 1.

Availability**Related Sample Code**

QuartzDemo

ZoomingPDFViewer

Declared In

CGPDFDocument.h

CGPDFDocumentGetTypeID

Returns the type identifier for Quartz PDF documents.

```
CTypeID CGPDFDocumentGetTypeID (  
    void  
);
```

Return Value

The identifier for the opaque type [CGPDFDocumentRef](#) (page 304).

Availability**Declared In**

CGPDFDocument.h

CGPDFDocumentGetVersion

Returns the major and minor version numbers of a Quartz PDF document.

```
void CGPDFDocumentGetVersion (  
    CGPDFDocumentRef document,  
    int *majorVersion,  
    int *minorVersion  
);
```

Parameters*document*

A PDF document.

majorVersion

On return, contains the major version number of the document.

minorVersion

On return, contains the minor version number of the document.

Return Value

On return, the values of the *majorVersion* and *minorVersion* parameters are set to the major and minor version numbers of the document respectively.

Availability**Declared In**

CGPDFDocument.h

CGPDFDocumentIsEncrypted

Returns whether the specified PDF file is encrypted.

```
bool CGPDFDocumentIsEncrypted (  
    CGPDFDocumentRef document  
);
```

Parameters

document

A PDF document.

Return Value

A Boolean that, if `true`, indicates that the document is encrypted. If the value is `false`, the document is not encrypted.

Discussion

If the document is encrypted, a password must be supplied before certain operations are enabled. For more information, see [CGPDFDocumentUnlockWithPassword](#) (page 303).

Availability

Declared In

CGPDFDocument.h

CGPDFDocumentIsUnlocked

Returns whether the specified PDF document is currently unlocked.

```
bool CGPDFDocumentIsUnlocked (  
    CGPDFDocumentRef document  
);
```

Parameters

document

A PDF document.

Return Value

A Boolean that, if `true`, indicates that the document is not locked. If the value is `false`, the document is locked.

Discussion

There are two possible reasons why a PDF document is unlocked:

- The document is not encrypted.
- The document is encrypted, and a valid password was previously specified using [CGPDFDocumentUnlockWithPassword](#) (page 303).

Availability

Declared In

CGPDFDocument.h

CGPDFDocumentRelease

Decrements the retain count of a PDF document.

```
void CGPDFDocumentRelease (  
    CGPDFDocumentRef document  
);
```

Parameters

document

The PDF document to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `document` parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

ZoomingPDFViewer

Declared In

`CGPDFDocument.h`

CGPDFDocumentRetain

Increments the retain count of a Quartz PDF document.

```
CGPDFDocumentRef CGPDFDocumentRetain (  
    CGPDFDocumentRef document  
);
```

Parameters

document

The PDF document to retain.

Return Value

The same document you passed in as the *document* parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `document` parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Declared In

`CGPDFDocument.h`

CGPDFDocumentUnlockWithPassword

Unlocks an encrypted PDF document, if a valid password is supplied.

```
bool CGPDFDocumentUnlockWithPassword (
    CGPDFDocumentRef document,
    const char *password
);
```

Parameters*document*

A PDF document.

password

A pointer to a string that contains the password.

Return Value

A Boolean that, if `true`, indicates that the document has been successfully unlocked. If the value is `false`, the document has not been unlocked.

Discussion

Given an encrypted PDF document and a password, this function does the following:

- Sets the lock state of the document, based on the validity of the password.
- Returns `true` if the document is unlocked.
- Returns `false` if the document cannot be unlocked with the specified password.

Unlocking a PDF document makes it possible to decrypt the document and perform other privileged operations. Different passwords enable different operations.

Availability**Declared In**`CGPDFDocument.h`

Data Types

CGPDFDocumentRef

An opaque type that represents a PDF (Portable Document Format) document.

```
typedef struct CGPDFDocument * CGPDFDocumentRef;
```

Availability

Available in iOS 2.0 and later.

Declared In`CGPDFDocument.h`

CGPDFObject Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFObject.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFObjectRef` opaque type represents PDF objects in a PDF document. PDF supports several basic types of object: Boolean values, integer and real numbers, strings, names, arrays, dictionaries, and streams. Most of these are represented in Quartz by corresponding specific types. A `CGPDFObject` can represent any of these types. You use `CGPDFObject` functions to determine the type of the object, and retrieve the object value if it is of an expected type.

This opaque type is not derived from `CType` and therefore there are no functions for retaining and releasing it. `CGPDFObject` objects exist as constituent parts of a `CGPDFDocument` object, and are managed by their container.

Functions

CGPDFObjectGetType

Returns the PDF type identifier of an object.

```
CGPDFObjectType CGPDFObjectGetType (
    CGPDFObjectRef object
);
```

Parameters

object

A PDF object. If the value is not a PDF object, the behavior is unspecified.

Return Value

Returns the type of the `object` parameter. See “Data Types” (page 306).

Availability

Declared In

`CGPDFObject.h`

CGPDFObjectGetValue

Returns whether an object is of a given type and if it is, retrieves its value.

```

bool CGPDFObjectGetValue (
    CGPDFObjectRef object,
    CGPDFObjectType type,
    void *value
);

```

Parameters

object

A PDF object.

type

A PDF object type.

value

If the *object* parameter is a PDF object of the specified type, then on return contains that object, otherwise the value is unspecified.

Return Value

Returns `true` if the specified object is a PDF object of the specified type, otherwise `false`.

Discussion

The function gets the value of the *object* parameter. If the type of *object* is equal to the type specified, then:

- If the *value* parameter is not a null pointer, then the value of *object* is copied to *value*, and the function returns `true`.
- If the *value* parameter is a null pointer, then the function simply returns `true`. This allows you to test whether *object* is of the type specified.

If the type of *object* is `kCGPDFObjectTypeInteger` and *type* is equal to `kCGPDFObjectTypeReal`, then the value of *object* is converted to floating point, the result copied to *value*, and the function returns `true`. If none of the preceding conditions is met, returns `false`.

Availability

Declared In

`CGPDFObject.h`

Data Types

CGPDFObjectRef

An opaque type that contains information about a PDF object.

```
typedef union CGPDFObject *CGPDFObjectRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFObject.h

CGPDFBoolean

A PDF Boolean value.

```
typedef unsigned char CGPDFBoolean;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFObject.h

CGPDFInteger

A PDF integer value.

```
typedef long int CGPDFInteger;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFObject.h

CGPDFReal

A PDF real value.

```
typedef CGFloat CGPDFReal;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFObject.h

Constants

PDF Object Types

Types of PDF object.

```
enum CGPDFObjectType {
    kCGPDFObjectTypeNull = 1,
    kCGPDFObjectTypeBoolean,
    kCGPDFObjectTypeInteger,
    kCGPDFObjectTypeReal,
    kCGPDFObjectTypeName,
    kCGPDFObjectTypeString,
    kCGPDFObjectTypeArray,
    kCGPDFObjectTypeDictionary,
    kCGPDFObjectTypeStream
};typedef enum CGPDFObjectType CGPDFObjectType;
```

Constants

`kCGPDFObjectTypeNull`

The type for a PDF null.

Available in iOS 2.0 and later.

Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeBoolean`

The type for a PDF Boolean.

Available in iOS 2.0 and later.

Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeInteger`

The type for a PDF integer.

Available in iOS 2.0 and later.

Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeReal`

The type for a PDF real.

Available in iOS 2.0 and later.

Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeName`

Type for a PDF name.

Available in iOS 2.0 and later.

Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeString`

The type for a PDF string.

Available in iOS 2.0 and later.

Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeArray`

Type for a PDF array.

Available in iOS 2.0 and later.

Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeDictionary`

The type for a PDF dictionary.

Available in iOS 2.0 and later.

Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeStream`

The type for a PDF stream.

Available in iOS 2.0 and later.

Declared in `CGPDFObject.h`.

Declared In

`CGPDFObject.h`

CGPDFOperatorTable Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFOperatorTable.h

Overview

A CGPDFOperatorTable object stores callback functions for PDF operators. You pass an operator table and a PDF content stream to a CGPDFScanner object. When the scanner parses a PDF operator, Quartz invokes your callback for that operator. See also *CGPDFScanner Reference* and *CGPDFContentStream Reference*.

Note: This opaque type is not derived from CType and therefore you can't use the Core Foundation base functions on it, such as `CFRetain` and `CFRelease`. Memory management is handled by the specific functions [CGPDFOperatorTableRetain](#) (page 312) and [CGPDFOperatorTableRelease](#) (page 312).

For more about PDF operators, see the latest version of *PDF Reference*, Adobe Systems Incorporated.

Functions by Task

Creating a PDF Operator Table

[CGPDFOperatorTableCreate](#) (page 312)
Creates an empty PDF operator table.

Setting Callback Functions

[CGPDFOperatorTableSetCallback](#) (page 313)
Sets a callback function for a PDF operator.

Retaining and Releasing a PDF Operator Table

[CGPDFOperatorTableRetain](#) (page 312)
Increments the retain count of a CGPDFOperatorTable object.

[CGPDFOperatorTableRelease](#) (page 312)

Decrements the retain count of a CGPDFOperatorTable object.

Functions

CGPDFOperatorTableCreate

Creates an empty PDF operator table.

```
CGPDFOperatorTableRef CGPDFOperatorTableCreate (  
    void  
);
```

Return Value

An empty PDF operator table. You are responsible for releasing this object by calling [CGPDFOperatorTableRelease](#) (page 312).

Discussion

Call the function [CGPDFOperatorTableSetCallback](#) (page 313) to fill the operator table with callbacks.

Availability

Declared In

CGPDFOperatorTable.h

CGPDFOperatorTableRelease

Decrements the retain count of a CGPDFOperatorTable object.

```
void CGPDFOperatorTableRelease (  
    CGPDFOperatorTableRef table  
);
```

Parameters

table

A PDF operator table.

Availability

Declared In

CGPDFOperatorTable.h

CGPDFOperatorTableRetain

Increments the retain count of a CGPDFOperatorTable object.


```
CGPDFOperatorTableRef CGPDFOperatorTableRetain (
    CGPDFOperatorTableRef table
);
```

Parameters*table*

A PDF operator table.

Return ValueThe same PDF operator table you passed in as the *table* parameter.**Availability****Declared In**

CGPDFOperatorTable.h

CGPDFOperatorTableSetCallback

Sets a callback function for a PDF operator.

```
void CGPDFOperatorTableSetCallback (
    CGPDFOperatorTableRef table,
    const char *name,
    CGPDFOperatorCallback callback
);
```

Parameters*table*

A PDF operator table.

name

The name of the PDF operator you want to set a callback for.

*callback*The callback to invoke for the PDF operator specified by the *name* parameter.**Discussion**

You call the function `CGPDFOperatorTableSetCallback` for each PDF operator for which you want to provide a callback. See Appendix A in the *PDF Reference, Second Edition*, version 1.3, Adobe Systems Incorporated for a summary of PDF operators.

Availability**Declared In**

CGPDFOperatorTable.h

Callbacks

CGPDFOperatorCallback

Performs custom processing for PDF operators.

```
typedef void (*CGPDFOperatorCallback)(
    CGPDFScannerRef scanner,
    void *info
);
```

If you name your function `MyCGPDFOperatorCallback`, you would declare it like this:

```
void MyCGPDFOperatorCallback (
    CGPDFScannerRef scanner,
    void *info
);
```

Parameters

scanner

A `CGPDFScanner` object. Quartz passes the scanner to your callback function. The scanner contains the PDF content stream that has the PDF operator that corresponds to this callback.

info

A pointer to data passed to the callback.

Discussion

Your callback function takes any action that's appropriate for your application. For example, if you want to count the number of inline images in a PDF but ignore the image data, you would set a callback for the `EI` operator. In your callback you would increment a counter for each call.

Availability

Available in iOS 2.0 and later.

Declared In

`CGPDFOperatorTable.h`

Data Types

CGPDFOperatorTableRef

An opaque type that stores callback functions for PDF operators.

```
typedef struct CGPDFOperatorTable *CGPDFOperatorTableRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

`CGPDFOperatorTable.h`

CGPDFPage Reference

Derived From:	CType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFPage.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFPageRef` opaque type represents a page in a PDF document.

Functions by Task

Retaining and Releasing a PDF Page

[CGPDFPageRetain](#) (page 320)

Increments the retain count of a PDF page.

[CGPDFPageRelease](#) (page 319)

Decrements the retain count of a PDF page.

Getting the CType ID

[CGPDFPageGetTypeID](#) (page 319)

Returns the CType ID for PDF page objects.

Getting Page Information

[CGPDFPageGetBoxRect](#) (page 316)

Returns the rectangle that represents a type of box for a content region or page dimensions of a PDF page.

[CGPDFPageGetDictionary](#) (page 316)

Returns the dictionary of a PDF page.

[CGPDFPageGetDocument](#) (page 317)

Returns the document for a page.

[CGPDFPageGetDrawingTransform](#) (page 317)

Returns the affine transform that maps a box to a given rectangle on a PDF page.

[CGPDFPageGetPageNumber](#) (page 318)

Returns the page number of the specified PDF page.

[CGPDFPageGetRotationAngle](#) (page 319)

Returns the rotation angle of a PDF page.

Functions

CGPDFPageGetBoxRect

Returns the rectangle that represents a type of box for a content region or page dimensions of a PDF page.

```
CGRect CGPDFPageGetBoxRect (
    CGPDFPageRef page,
    CGPDFBox box
);
```

Parameters

page

A PDF page.

box

A `CGPDFBox` constant that specifies the type of box. For possible values, see “PDF Boxes” (page 321).

Return Value

Returns the rectangle associated with the type of box specified by the `box` parameter in the specified page.

Discussion

Returns the rectangle associated with the specified box in the specified page. This is the value of the corresponding entry (such as `/MediaBox`, `/ArtBox`, and so on) in the page’s dictionary.

Availability

Available in iOS 2.0 and later.

Related Sample Code

ZoomingPDFViewer

Declared In

`CGPDFPage.h`

CGPDFPageGetDictionary

Returns the dictionary of a PDF page.

```
CGPDFDictionaryRef CGPDFPageGetDictionary (  
    CGPDFPageRef page  
);
```

Parameters

page

A PDF page.

Return Value

Returns the PDF dictionary for the specified page.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFPage.h

CGPDFPageGetDocument

Returns the document for a page.

```
CGPDFDocumentRef CGPDFPageGetDocument (  
    CGPDFPageRef page  
);
```

Parameters

page

A PDF page.

Return Value

The PDF document with which the specified page is associated.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFPage.h

CGPDFPageGetDrawingTransform

Returns the affine transform that maps a box to a given rectangle on a PDF page.

```
CGAffineTransform CGPDFPageGetDrawingTransform (  
    CGPDFPageRef page,  
    CGPDFBox box,  
    CGRect rect,  
    int rotate,  
    bool preserveAspectRatio  
);
```

Parameters

page

A PDF page.

box

A `CGPDFBox` constant that specifies the type of box. For possible values, see “PDF Boxes” (page 321).

rect

A Quartz rectangle.

rotate

An integer, that must be a multiple of 90, that specifies the angle by which the specified rectangle is rotated clockwise.

preserveAspectRatio

A Boolean value that specifies whether or not the aspect ratio should be preserved. A value of `true` specifies that the aspect ratio should be preserved.

Return Value

An affine transform that maps the box specified by the `box` parameter to the rectangle specified by the `rect` parameter.

Discussion

Quartz constructs the affine transform as follows:

- Computes the effective rectangle by intersecting the rectangle associated with `box` and the `/MediaBox` entry of the specified page.
- Rotates the effective rectangle according to the page’s `/Rotate` entry.
- Centers the resulting rectangle on `rect`. If the value of the `rotate` parameter is non-zero, then the rectangle is rotated clockwise by `rotate` degrees. The value of `rotate` must be a multiple of 90.
- Scales the rectangle, if necessary, so that it coincides with the edges of `rect`. If the value of `preserveAspectRatio` parameter is `true`, then the final rectangle coincides with the edges of `rect` only in the more restrictive dimension.

Availability

Available in iOS 2.0 and later.

Related Sample Code

QuartzDemo

Declared In

`CGPDFPage.h`

CGPDFPageGetPageNumber

Returns the page number of the specified PDF page.

```
size_t CGPDFPageGetPageNumber (
    CGPDFPageRef page
);
```

Parameters

page

A PDF page.

Return Value

Returns the page number of the specified page.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFPage.h

CGPDFPageGetRotationAngle

Returns the rotation angle of a PDF page.

```
int CGPDFPageGetRotationAngle (
    CGPDFPageRef page
);
```

Parameters

page

A PDF page.

Return Value

The rotation angle (in degrees) of the specified page. This is the value of the `/Rotate` entry in the page's dictionary.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFPage.h

CGPDFPageGetTypeID

Returns the CType ID for PDF page objects.

```
CTypeID CGPDFPageGetTypeID (
    void
);
```

Return Value

Returns the Core Foundation type for a PDF page.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFPage.h

CGPDFPageRelease

Decrements the retain count of a PDF page.

```
void CGPDFPageRelease (
    CGPDFPageRef page
);
```

Parameters

page

A PDF page.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `page` parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Related Sample Code

ZoomingPDFViewer

Declared In

CGPDFPage.h

CGPDFPageRetain

Increments the retain count of a PDF page.

```
CGPDFPageRef CGPDFPageRetain (
    CGPDFPageRef page
);
```

Parameters

page

A PDF page.

Return Value

The same page you passed in as the `page` parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `page` parameter is `NULL`.

Availability

Available in iOS 2.0 and later.

Related Sample Code

ZoomingPDFViewer

Declared In

CGPDFPage.h

Data Types

CGPDFPageRef

An opaque type that represents a page in a PDF document.

```
typedef struct CGPDFPage *CGPDFPageRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFPage.h

Constants

PDF Boxes

Box types for a PDF page.

```
enum CGPDFBox {  
    kCGPDFMediaBox = 0,  
    kCGPDFCropBox = 1,  
    kCGPDFBleedBox = 2,  
    kCGPDFTrimBox = 3,  
    kCGPDFArtBox = 4  
};  
typedef enum CGPDFBox CGPDFBox;
```

Constants

kCGPDFMediaBox

The page media box—a rectangle, expressed in default user space units, that defines the boundaries of the physical medium on which the page is intended to be displayed or printed

Available in iOS 2.0 and later.

Declared in CGPDFPage.h.

kCGPDFCropBox

The page crop box—a rectangle, expressed in default user space units, that defines the visible region of default user space. When the page is displayed or printed, its contents are to be clipped to this rectangle.

Available in iOS 2.0 and later.

Declared in CGPDFPage.h.

kCGPDFBleedBox

The page bleed box—a rectangle, expressed in default user space units, that defines the region to which the contents of the page should be clipped when output in a production environment

Available in iOS 2.0 and later.

Declared in CGPDFPage.h.

`kCGPDFTrimBox`

The page trim box—a rectangle, expressed in default user space units, that defines the intended dimensions of the finished page after trimming.

Available in iOS 2.0 and later.

Declared in `CGPDFPage.h`.

`kCGPDFArtBox`

The page art box—a rectangle, expressed in default user space units, defining the extent of the page's meaningful content (including potential white space) as intended by the page's creator.

Available in iOS 2.0 and later.

Declared in `CGPDFPage.h`.

Declared In

`CGPDFPage.h`

CGPDFScanner Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFScanner.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFScannerRef` opaque type is used to parse a PDF content stream. You can set up the PDF scanner object to invoke callbacks when it encounters specific PDF operators in the stream.

This opaque type is not derived from `CType`. Use [CGPDFScannerRetain](#) (page 330) and [CGPDFScannerRelease](#) (page 329) to manage the retain count of `CGPDFScannerRef` instances; do not use `CFRetain` and `CFRelease`.

Functions by Task

Creating a PDF Scanner Object

[CGPDFScannerCreate](#) (page 324)
Creates a PDF scanner.

Retaining and Releasing PDF Scanner Objects

[CGPDFScannerRetain](#) (page 330)
Increments the retain count of a scanner object.

[CGPDFScannerRelease](#) (page 329)
Decrements the retain count of a scanner object.

Parsing Content

[CGPDFScannerScan](#) (page 330)
Parses the content stream of a PDF scanner object.

[CGPDFScannerGetContentStream](#) (page 325)

Returns the content stream associated with a PDF scanner object.

Getting PDF Objects from the Scanner Stack

[CGPDFScannerPopObject](#) (page 328)

Retrieves an object from the scanner stack.

[CGPDFScannerPopBoolean](#) (page 326)

Retrieves a Boolean object from the scanner stack.

[CGPDFScannerPopInteger](#) (page 327)

Retrieves an integer object from the scanner stack.

[CGPDFScannerPopNumber](#) (page 327)

Retrieves a real value object from the scanner stack.

[CGPDFScannerPopName](#) (page 327)

Retrieves a character string from the scanner stack.

[CGPDFScannerPopString](#) (page 329)

Retrieves a string object from the scanner stack.

[CGPDFScannerPopArray](#) (page 325)

Retrieves an array object from the scanner stack.

[CGPDFScannerPopDictionary](#) (page 326)

Retrieves a PDF dictionary object from the scanner stack.

[CGPDFScannerPopStream](#) (page 328)

Retrieves a PDF stream object from the scanner stack.

Functions

CGPDFScannerCreate

Creates a PDF scanner.

```
CGPDFScannerRef CGPDFScannerCreate (
    CGPDFContentStreamRef cs,
    CGPDFOperatorTableRef table,
    void *info
);
```

Parameters

cs

A PDF content stream object. (See *CGPDFContentStream Reference*.)

table

A table of callbacks for the PDF operators you want to handle.

info

A pointer to data you want passed to your callback function. (See *CGPDFOperatorTable Reference*.)

Return Value

A PDF scanner object. You are responsible for releasing this object by calling the function `CGPDFScannerRelease`.

Discussion

When you want to parse the contents of the PDF stream, call the function [CGPDFScannerScan](#) (page 330).

Availability

Available in iOS 2.0 and later.

Declared In

`CGPDFScanner.h`

CGPDFScannerGetContentStream

Returns the content stream associated with a PDF scanner object.

```
CGPDFContentStreamRef CGPDFScannerGetContentStream (  
    CGPDFScannerRef scanner  
);
```

Parameters

scanner

The scanner object whose content stream you want to obtain.

Return Value

The content stream associated with *scanner*.

Availability

Available in iOS 2.0 and later.

Declared In

`CGPDFScanner.h`

CGPDFScannerPopArray

Retrieves an array object from the scanner stack.

```
bool CGPDFScannerPopArray (  
    CGPDFScannerRef scanner,  
    CGPDFArrayRef *value  
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the PDF array object popped from the scanner stack.

Return Value

`true` if the array object is retrieved successfully; otherwise, `false`.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFScannerPopBoolean

Retrieves a Boolean object from the scanner stack.

```
bool CGPDFScannerPopBoolean (
    CGPDFScannerRef scanner,
    CGPDFBoolean *value
);
```

Parameters*scanner*

A valid scanner object.

value

On output, points to the PDF Boolean object popped from the scanner stack.

Return Value

true if the boolean object is retrieved successfully; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFScannerPopDictionary

Retrieves a PDF dictionary object from the scanner stack.

```
bool CGPDFScannerPopDictionary (
    CGPDFScannerRef scanner,
    CGPDFDictionaryRef *value
);
```

Parameters*scanner*

A valid scanner object.

value

On output, points to the PDF dictionary object popped from the scanner stack.

Return Value

true if the PDF dictionary object is retrieved successfully; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFScannerPopInteger

Retrieves an integer object from the scanner stack.

```
bool CGPDFScannerPopInteger (
    CGPDFScannerRef scanner,
    CGPDFInteger *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the PDF integer object popped from the scanner stack.

Return Value

true if the PDF integer is retrieved successfully; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFScannerPopName

Retrieves a character string from the scanner stack.

```
bool CGPDFScannerPopName (
    CGPDFScannerRef scanner,
    const char **value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the character string popped from the scanner stack.

Return Value

true if the string is retrieved successfully; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFScannerPopNumber

Retrieves a real value object from the scanner stack.

```
bool CGPDFScannerPopNumber (
    CGPDFScannerRef scanner,
    CGPDFReal *value
);
```

Parameters*scanner*

A valid scanner object.

value

On output, points to the real value object popped from the scanner stack.

Return Value

true if the real value is retrieved successfully; otherwise, false.

Discussion

The number retrieved from the scanner can be a real value or an integer value. However, the result is always converted to a value of type `CGPDFReal`.

Availability

Available in iOS 2.0 and later.

Declared In`CGPDFScanner.h`**CGPDFScannerPopObject**

Retrieves an object from the scanner stack.

```
bool CGPDFScannerPopObject (
    CGPDFScannerRef scanner,
    CGPDFObjectRef *value
);
```

Parameters*scanner*

A valid scanner object.

value

On output, points to the object popped from the scanner stack.

Return Value

true if the object is retrieved successfully; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In`CGPDFScanner.h`**CGPDFScannerPopStream**

Retrieves a PDF stream object from the scanner stack.


```
bool CGPDFScannerPopStream (
    CGPDFScannerRef scanner,
    CGPDFStreamRef *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the PDF stream object popped from the scanner stack.

Return Value

true if the stream object is retrieved successfully; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFScannerPopString

Retrieves a string object from the scanner stack.

```
bool CGPDFScannerPopString (
    CGPDFScannerRef scanner,
    CGPDFStringRef *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the string object popped from the scanner stack.

Return Value

true if the string is retrieved successfully; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFScannerRelease

Decrements the retain count of a scanner object.

```
void CGPDFScannerRelease (
    CGPDFScannerRef scanner
);
```

Parameters

scanner

The scanner object to release.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFScannerRetain

Increments the retain count of a scanner object.

```
CGPDFScannerRef CGPDFScannerRetain (
    CGPDFScannerRef scanner
);
```

Parameters

scanner

The scanner object to retain.

Return Value

The same scanner object passed to the function in the *scanner* parameter.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFScannerScan

Parses the content stream of a PDF scanner object.

```
bool CGPDFScannerScan (
    CGPDFScannerRef scanner
);
```

Parameters

scanner

The scanner object whose content stream you want to parse.

Return Value

`true` if the entire stream is parsed successfully; `false` if parsing fails (for example, if the stream data is corrupted).

Discussion

The function `CGPDFScannerScan` parses the PDF content stream associated with the scanner. Each time Quartz parses a PDF operator for which you register a callback, Quartz invokes your callback.

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

Data Types

CGPDFScannerRef

An opaque type used to parse a PDF content stream.

```
typedef struct CGPDFScanner *CGPDFScannerRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFScanner.h

CGPDFStream Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFStream.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFStreamRef` opaque type represents a PDF stream. A PDF stream consists of a dictionary that describes a sequence of bytes. Streams typically represent objects with potentially large amounts of data, such as images and page descriptions.

This opaque type is not derived from `CType` and therefore there are no functions for retaining and releasing it.

Functions

CGPDFStreamCopyData

Returns the data associated with a PDF stream.

```
CFDataRef CGPDFStreamCopyData (
    CGPDFStreamRef stream,
    CGPDFDataFormat *format
);
```

Parameters

stream

A PDF stream.

format

On return, contains a constant that specifies the format of the data returned—[CGPDFDataFormatRaw](#) (page 335), [CGPDFDataFormatJPEGEncoded](#) (page 335), or [CGPDFDataFormatJPEG2000](#) (page 335).

Return Value

A `CFData` object that contains a copy of the stream data. You are responsible for releasing this object.

Availability**Declared In**

CGPDFStream.h

CGPDFStreamGetDictionary

Returns the dictionary associated with a PDF stream.

```
CGPDFDictionaryRef CGPDFStreamGetDictionary (  
    CGPDFStreamRef stream  
);
```

Parameters*stream*

A PDF stream.

Return Value

The PDF dictionary for the specified stream.

Availability**Declared In**

CGPDFStream.h

Data Types

CGPDFStream

An opaque type that represents a PDF stream.

```
typedef struct CGPDFStream *CGPDFStreamRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFStream.h

Constants

CGPDFDataFormat

The encoding format of PDF data.

```
enum CGPDFDataFormat {  
    CGPDFDataFormatRaw,  
    CGPDFDataFormatJPEGEncoded,  
    CGPDFDataFormatJPEG2000  
};  
typedef enum CGPDFDataFormat CGPDFDataFormat;
```

Constants

CGPDFDataFormatRaw

The data stream is not encoded.

Available in iOS 2.0 and later.

Declared in CGPDFStream.h.

CGPDFDataFormatJPEGEncoded

The data stream is encoded in JPEG format.

Available in iOS 2.0 and later.

Declared in CGPDFStream.h.

CGPDFDataFormatJPEG2000

The data stream is encoded in JPEG-2000 format.

Available in iOS 2.0 and later.

Declared in CGPDFStream.h.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

CGPDFStream.h

CGPDFString Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFString.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFStringRef` opaque type represents a string in a PDF document. A PDF string object is a series of bytes—unsigned integer values in the range 0 to 255. The string elements are not integer objects, but are stored in a more compact format. For more information on the representation of strings in PDF, see the latest version of *PDF Reference*, Adobe Systems Incorporated.

This opaque type is not derived from `CType` and therefore there are no functions for retaining and releasing it. `CGPDFString` objects exist as constituent parts of a `CGPDFDocument` object, and are managed by their container.

Functions by Task

Converting PDF Strings

[CGPDFStringCopyTextString](#) (page 338)

Returns a `CFString` object that represents a PDF string as a text string.

[CGPDFStringCopyDate](#) (page 338)

Converts a string to a date.

Getting PDF String Data

[CGPDFStringGetBytePtr](#) (page 338)

Returns a pointer to the bytes of a PDF string.

[CGPDFStringGetLength](#) (page 339)

Returns the number of bytes in a PDF string.

Functions

CGPDFStringCopyDate

Converts a string to a date.

```
CFDateRef CGPDFStringCopyDate (  
    CGPDFStringRef string  
);
```

Parameters

string

The string to convert to a date.

Return Value

A CFDate object.

Discussion

The PDF specification defines a specific format for strings that represent dates. This function converts strings in that form to CFDate objects.

Availability

Declared In

CGPDFString.h

CGPDFStringCopyTextString

Returns a CFString object that represents a PDF string as a text string.

```
CFStringRef CGPDFStringCopyTextString (  
    CGPDFStringRef string  
);
```

Parameters

string

A PDF string. If this value is NULL, it will cause an error.

Return Value

Returns a CFString object that represents the specified PDF string as a text string. You are responsible for releasing this object.

Availability

Declared In

CGPDFString.h

CGPDFStringGetBytePtr

Returns a pointer to the bytes of a PDF string.

```
const unsigned char * CGPDFStringGetBytePtr (  
    CGPDFStringRef string  
);
```

Parameters

string

A PDF string.

Return Value

Returns a pointer to the bytes of the specified string. If the string is `NULL`, the function returns `NULL`.

Availability**Declared In**

CGPDFString.h

CGPDFStringGetLength

Returns the number of bytes in a PDF string.

```
size_t CGPDFStringGetLength (  
    CGPDFStringRef string  
);
```

Parameters

string

A PDF string.

Return Value

Returns the number of bytes referenced by the string, or 0 if the string is `NULL`.

Availability**Declared In**

CGPDFString.h

Data Types

CGPDFStringRef

An opaque data type that represents a string in a PDF document.

```
typedef struct CGPDFString *CGPDFStringRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGPDFString.h

CGShading Reference

Derived From:	CType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGShading.h
Companion guide	Quartz 2D Programming Guide

Overview

`CGShadingRef` is an opaque type used to define linear (axial) and radial gradient fills whose color transitions are controlled by a function ([CGFunctionRef](#) (page 193)) that you provide. Shading means to fill using a smooth transition between colors across an area. To paint with a Quartz shading, you call [CGContextDrawShading](#) (page 81). This function fills the current clipping path using the specified color gradient, calling your parametric function repeatedly as it draws.

An alternative to using a `CGShading` object is to use the [CGGradientRef](#) (page 199) opaque type. For applications that run in Mac OS X v10.5 and later, `CGGradient` objects are much simpler to use. (See *CGGradient Reference*.)

Functions by Task

Creating Shading Objects

[CGShadingCreateAxial](#) (page 342)
Creates a shading object to use for axial shading.

[CGShadingCreateRadial](#) (page 343)
Creates a shading object to use for radial shading.

Retaining and Releasing Shading Objects

[CGShadingRetain](#) (page 344)
Increments the retain count of a shading object.

[CGShadingRelease](#) (page 344)
Decrements the retain count of a shading object.

Getting the CType ID

[CGShadingGetTypeID](#) (page 343)

Returns the Core Foundation type identifier for Quartz shading objects.

Functions

CGShadingCreateAxial

Creates a shading object to use for axial shading.

```
CGShadingRef CGShadingCreateAxial (
    CGColorSpaceRef colorspace,
    CGPoint start,
    CGPoint end,
    CGFunctionRef function,
    bool extendStart,
    bool extendEnd
);
```

Parameters

colorspace

The color space in which color values are expressed. Quartz retains this object; upon return, you may safely release it.

start

The starting point of the axis, in the shading's target coordinate space.

end

The ending point of the axis, in the shading's target coordinate space.

function

A `CGFunction` object created by the function `CGFunctionCreate`. This object refers to your function for creating an axial shading. Quartz retains this object; upon return, you may safely release it.

extendStart

A Boolean value that specifies whether to extend the shading beyond the starting point of the axis.

extendEnd

A Boolean value that specifies whether to extend the shading beyond the ending point of the axis.

Return Value

A new Quartz axial shading. You are responsible for releasing this object using [CGShadingRelease](#) (page 344).

Discussion

An axial shading is a color blend that varies along a linear axis between two endpoints and extends indefinitely perpendicular to that axis. When you are ready to draw the shading, call the function [CGContextDrawShading](#) (page 81).

Availability

Declared In

`CGShading.h`

CGShadingCreateRadial

Creates a shading object to use for radial shading.

```
CGShadingRef CGShadingCreateRadial (
    CGColorSpaceRef colorspace,
    CGPoint start,
    CGFloat startRadius,
    CGPoint end,
    CGFloat endRadius,
    CGFunctionRef function,
    bool extendStart,
    bool extendEnd
);
```

Parameters

colorspace

The color space in which color values are expressed. Quartz retains this object; upon return, you may safely release it.

start

The center of the starting circle, in the shading's target coordinate space.

startRadius

The radius of the starting circle, in the shading's target coordinate space.

end

The center of the ending circle, in the shading's target coordinate space.

endRadius

The radius of the ending circle, in the shading's target coordinate space.

function

A CGFunction object created by the function `CGFunctionCreate`. This object refers to your function for creating a radial shading. Quartz retains this object; upon return, you may safely release it.

extendStart

A Boolean value that specifies whether to extend the shading beyond the starting circle.

extendEnd

A Boolean value that specifies whether to extend the shading beyond the ending circle.

Return Value

A new Quartz radial shading. You are responsible for releasing this object using [CGShadingRelease](#) (page 344).

Discussion

A radial shading is a color blend that varies between two circles. To draw the shading, call the function [CGContextDrawShading](#) (page 81).

Availability

Declared In

`CGShading.h`

CGShadingGetTypeID

Returns the Core Foundation type identifier for Quartz shading objects.

```
CTypeID CGShadingGetTypeID (  
    void  
);
```

Return Value

The Core Foundation identifier for the opaque type [CGShadingRef](#) (page 345).

Availability**Declared In**

CGShading.h

CGShadingRelease

Decrements the retain count of a shading object.

```
void CGShadingRelease (  
    CGShadingRef shading  
);
```

Parameters

shading

The shading object to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the *shading* parameter is `NULL`.

Availability**Declared In**

CGShading.h

CGShadingRetain

Increments the retain count of a shading object.

```
CGShadingRef CGShadingRetain (  
    CGShadingRef shading  
);
```

Parameters

shading

The shading object to retain.

Return Value

The same shading object you passed in as the *shading* parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the *shading* parameter is `NULL`.

Availability**Declared In**

CGShading.h

Data Types

CGShadingRef

An opaque type that represents a Quartz shading.

```
typedef struct CGShading *CGShadingRef;
```

Availability

Available in iOS 2.0 and later.

Declared In

CGShading.h

Constants

Core Graphics Constants Reference

Declared in `CGError.h`

Overview

This document describes assorted constants and result codes related to Core Graphics and Quartz 2D.

Result Codes

This table lists the result codes returned by functions in Core Graphics.

Result Code	Value	Description
<code>kCGErrorSuccess</code>	0	The requested operation was completed successfully. Available in iOS 2.0 and later.
<code>kCGErrorFailure</code>	1000	A general failure occurred. Available in iOS 2.0 and later.
<code>kCGErrorIllegalArgument</code>	1001	One or more of the parameters passed to a function are invalid. Check for <code>NULL</code> pointers. Available in iOS 2.0 and later.
<code>kCGErrorInvalidConnection</code>	1002	The parameter representing a connection to the window server is invalid. Available in iOS 2.0 and later.
<code>kCGErrorInvalidContext</code>	1003	The <code>CPSPProcessSerNum</code> or context identifier parameter is not valid. Available in iOS 2.0 and later.

Result Code	Value	Description
<code>kCGErrorCannotComplete</code>	1004	The requested operation is inappropriate for the parameters passed in, or the current system state. Available in iOS 2.0 and later.
<code>kCGErrorNameTooLong</code>	1005	A parameter, typically a C string, is too long to be used without truncation. Available in iOS 2.0 and later.
<code>kCGErrorNotImplemented</code>	1006	Return value from obsolete function stubs present for binary compatibility, but not normally called. Available in iOS 2.0 and later.
<code>kCGErrorRangeCheck</code>	1007	A parameter passed in has a value that is inappropriate, or which does not map to a useful operation or value. Available in iOS 2.0 and later.
<code>kCGErrorTypeCheck</code>	1008	A data type or token was encountered that did not match the expected type or token. Available in iOS 2.0 and later.
<code>kCGErrorNoCurrentPoint</code>	1009	An operation relative to a known point or coordinate could not be done, as there is no known point. Available in iOS 2.0 and later.
<code>kCGErrorInvalidOperation</code>	1010	The requested operation is not valid for the parameters passed in, or the current system state. Available in iOS 2.0 and later.
<code>kCGErrorNoneAvailable</code>	1011	The requested operation could not be completed as the indicated resources were not found. Available in iOS 2.0 and later.

Result Code	Value	Description
<code>kCGErrorApplicationRequiresNewerSystem</code>	1015	The application requires a newer version of the operating system to run than is currently available. Available in iOS 2.0 and later.
<code>kCGErrorApplicationNotPermittedToExecute</code>	1016	The application is not permitted to run. Available in iOS 2.0 and later.
<code>kCGErrorApplicationIncorrectExecutableFormatFound</code>	1023	The application does not have any executable code for the current system. Available in iOS 2.0 and later.
<code>kCGErrorApplicationIsLaunching</code>	1024	The application is in the process of launching but has not yet checked in with the window server. Available in iOS 2.0 and later.
<code>kCGErrorApplicationAlreadyRunning</code>	1025	The application being launched was already running and checked in with the window server. Available in iOS 2.0 and later.
<code>kCGErrorApplicationCanOnlyBeRunInOneSessionAtATime</code>	1026	The application being launched is incompatible with multiple user sessions and is already running in another user session. Available in iOS 2.0 and later.
<code>kCGErrorClassicApplicationsMustBeLaunchedByClassic</code>	1027	The window server could not launch the application. It must be launched by the Classic environment. Available in iOS 2.0 and later.
<code>kCGErrorForkFailed</code>	1028	The system was unable to fork a new process to launch the application. Available in iOS 2.0 and later.
<code>kCGErrorRetryRegistration</code>	1029	The application should retry its registration shortly. Available in iOS 2.0 and later.

Other References

CGAffineTransform Reference

Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGAffineTransform.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGAffineTransform` data structure represents a matrix used for affine transformations. A transformation specifies how points in one coordinate system map to points in another coordinate system. An affine transformation is a special type of mapping that preserves parallel lines in a path but does not necessarily preserve lengths or angles. Scaling, rotation, and translation are the most commonly used manipulations supported by affine transforms, but skewing is also possible.

Quartz provides functions that create, concatenate, and apply affine transformations using the `CGAffineTransform` data structure. For information on how to use affine transformation functions, see *Quartz 2D Programming Guide*.

You typically do not need to create an affine transform directly—*CGContext Reference* describes functions that modify the current affine transform. If you don't plan to reuse an affine transform, you may want to use [CGContextScaleCTM](#) (page 97), [CGContextRotateCTM](#) (page 95), [CGContextTranslateCTM](#) (page 132), or [CGContextConcatCTM](#) (page 74).

Functions by Task

Creating an Affine Transformation Matrix

[CGAffineTransformMake](#) (page 358)

Returns an affine transformation matrix constructed from values you provide.

[CGAffineTransformMakeRotation](#) (page 359)

Returns an affine transformation matrix constructed from a rotation value you provide.

[CGAffineTransformMakeScale](#) (page 360)

Returns an affine transformation matrix constructed from scaling values you provide.

[CGAffineTransformMakeTranslation](#) (page 361)

Returns an affine transformation matrix constructed from translation values you provide.

Modifying Affine Transformations

[CGAffineTransformTranslate](#) (page 363)

Returns an affine transformation matrix constructed by translating an existing affine transform.

[CGAffineTransformScale](#) (page 363)

Returns an affine transformation matrix constructed by scaling an existing affine transform.

[CGAffineTransformRotate](#) (page 362)

Returns an affine transformation matrix constructed by rotating an existing affine transform.

[CGAffineTransformInvert](#) (page 357)

Returns an affine transformation matrix constructed by inverting an existing affine transform.

[CGAffineTransformConcat](#) (page 356)

Returns an affine transformation matrix constructed by combining two existing affine transforms.

Applying Affine Transformations

[CGPointApplyAffineTransform](#) (page 364)

Returns the point resulting from an affine transformation of an existing point.

[CGSizeApplyAffineTransform](#) (page 365)

Returns the height and width resulting from a transformation of an existing height and width.

[CGRectApplyAffineTransform](#) (page 365)

Applies an affine transform to a rectangle.

Evaluating Affine Transforms

[CGAffineTransformIsIdentity](#) (page 358)

Checks whether an affine transform is the identity transform.

[CGAffineTransformEqualToTransform](#) (page 357)

Checks whether two affine transforms are equal.

Functions

CGAffineTransformConcat

Returns an affine transformation matrix constructed by combining two existing affine transforms.

```
CGAffineTransform CGAffineTransformConcat (
    CGAffineTransform t1,
    CGAffineTransform t2
);
```

Parameters

t1

The first affine transform.

t2

The second affine transform. This affine transform is concatenated to the first affine transform.

Return Value

A new affine transformation matrix. That is, $t' = t1 * t2$.

Discussion

Concatenation combines two affine transformation matrices by multiplying them together. You might perform several concatenations in order to create a single affine transform that contains the cumulative effects of several transformations.

Note that matrix operations are not commutative—the order in which you concatenate matrices is important. That is, the result of multiplying matrix *t1* by matrix *t2* does not necessarily equal the result of multiplying matrix *t2* by matrix *t1*.

Availability**Declared In**

CGAffineTransform.h

CGAffineTransformEqualToTransform

Checks whether two affine transforms are equal.

```
bool CGAffineTransformEqualToTransform (
    CGAffineTransform t1,
    CGAffineTransform t2
);
```

Parameters*t1*

An affine transform.

t2

An affine transform.

Return Value

Returns `true` if *t1* and *t2* are equal, `false` otherwise.

Availability**Declared In**

CGAffineTransform.h

CGAffineTransformInvert

Returns an affine transformation matrix constructed by inverting an existing affine transform.

```
CGAffineTransform CGAffineTransformInvert (
    CGAffineTransform t
);
```

Parameters*t*

An existing affine transform.

Return Value

A new affine transformation matrix. If the affine transform passed in parameter `t` cannot be inverted, Quartz returns the affine transform unchanged.

Discussion

Inversion is generally used to provide reverse transformation of points within transformed objects. Given the coordinates (x,y) , which have been transformed by a given matrix to new coordinates (x',y') , transforming the coordinates (x',y') by the inverse matrix produces the original coordinates (x,y) .

Availability**Declared In**

CGAffineTransform.h

CGAffineTransformIsIdentity

Checks whether an affine transform is the identity transform.

```
bool CGAffineTransformIsIdentity (
    CGAffineTransform t
);
```

Parameters

t

The affine transform to check.

Return Value

Returns `true` if `t` is the identity transform, `false` otherwise.

Availability**Declared In**

CGAffineTransform.h

CGAffineTransformMake

Returns an affine transformation matrix constructed from values you provide.

```
CGAffineTransform CGAffineTransformMake (
    CGFloat a,
    CGFloat b,
    CGFloat c,
    CGFloat d,
    CGFloat tx,
    CGFloat ty
);
```

Parameters

a

The value at position [1,1] in the matrix.

b

The value at position [1,2] in the matrix.

c

The value at position [2,1] in the matrix.

d

The value at position [2,2] in the matrix.

tx

The value at position [3,1] in the matrix.

ty

The value at position [3,2] in the matrix.

Return Value

A new affine transform matrix constructed from the values you specify.

Discussion

This function creates a `CGAffineTransform` structure that represents a new affine transformation matrix, which you can use (and reuse, if you want) to transform a coordinate system. The matrix takes the following form:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Because the third column is always $(0, 0, 1)$, the `CGAffineTransform` data structure returned by this function contains values for only the first two columns.

If you want only to transform an object to be drawn, it is not necessary to construct an affine transform to do so. The most direct way to transform your drawing is by calling the appropriate `CGContext` function to adjust the current transformation matrix. For a list of functions, see *CGContext Reference*.

Availability**Related Sample Code**

PVRTextureLoader

Declared In`CGAffineTransform.h`**CGAffineTransformMakeRotation**

Returns an affine transformation matrix constructed from a rotation value you provide.

```
CGAffineTransform CGAffineTransformMakeRotation (
    CGFloat angle
);
```

Parameters*angle*

The angle, in radians, by which this matrix rotates the coordinate system axes. In iOS, a positive value specifies counterclockwise rotation and a negative value specifies clockwise rotation. In Mac OS X, a positive value specifies clockwise rotation and a negative value specifies counterclockwise rotation.

Return Value

A new affine transformation matrix.

Discussion

This function creates a `CGAffineTransform` structure, which you can use (and reuse, if you want) to rotate a coordinate system. The matrix takes the following form:

$$\begin{bmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The actual direction of rotation is dependent on the coordinate system orientation of the target platform, which is different in iOS and Mac OS X. Because the third column is always $(0, 0, 1)$, the `CGAffineTransform` data structure returned by this function contains values for only the first two columns.

These are the resulting equations that Quartz uses to apply the rotation to a point (x, y) :

$$x' = x \cos a - y \sin a$$

$$y' = x \sin a + y \cos a$$

If you want only to rotate an object to be drawn, it is not necessary to construct an affine transform to do so. The most direct way to rotate your drawing is by calling the function `CGContextRotateCTM` (page 95).

Availability**Related Sample Code**

`aurioTouch`

`GKTank`

`SimpleGestureRecognizer`

Declared In

`CGAffineTransform.h`

CGAffineTransformMakeScale

Returns an affine transformation matrix constructed from scaling values you provide.

```
CGAffineTransform CGAffineTransformMakeScale (
    CGFloat sx,
    CGFloat sy
);
```

Parameters

`sx`

The factor by which to scale the x-axis of the coordinate system.

`sy`

The factor by which to scale the y-axis of the coordinate system.

Return Value

A new affine transformation matrix.

Discussion

This function creates a `CGAffineTransform` structure, which you can use (and reuse, if you want) to scale a coordinate system. The matrix takes the following form:

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Because the third column is always $(0, 0, 1)$, the `CGAffineTransform` data structure returned by this function contains values for only the first two columns.

These are the resulting equations that Quartz uses to scale the coordinates of a point (x, y) :

$$x' = x \cdot s_x$$

$$y' = y \cdot s_y$$

If you want only to scale an object to be drawn, it is not necessary to construct an affine transform to do so. The most direct way to scale your drawing is by calling the function [CGContextScaleCTM](#) (page 97).

Availability**Related Sample Code**

MoveMe

QuartzDemo

Touches

Declared In

`CGAffineTransform.h`

CGAffineTransformMakeTranslation

Returns an affine transformation matrix constructed from translation values you provide.

```
CGAffineTransform CGAffineTransformMakeTranslation (
    CGFloat tx,
    CGFloat ty
);
```

Parameters

tx

The value by which to move the x-axis of the coordinate system.

ty

The value by which to move the y-axis of the coordinate system.

Return Value

A new affine transform matrix.

Discussion

This function creates a `CGAffineTransform` structure, which you can use (and reuse, if you want) to move a coordinate system. The matrix takes the following form:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Because the third column is always $(0, 0, 1)$, the `CGAffineTransform` data structure returned by this function contains values for only the first two columns.

These are the resulting equations Quartz uses to apply the translation to a point (x, y) :

$$x' = x + t_x$$

$$y' = y + t_y$$

If you want only to move the location where an object is drawn, it is not necessary to construct an affine transform to do so. The most direct way to move your drawing is by calling the function [CGContextTranslateCTM](#) (page 132).

Availability

Declared In

`CGAffineTransform.h`

CGAffineTransformRotate

Returns an affine transformation matrix constructed by rotating an existing affine transform.

```
CGAffineTransform CGAffineTransformRotate (
    CGAffineTransform t,
    CGFloat angle
);
```

Parameters

t

An existing affine transform.

angle

The angle, in radians, by which to rotate the affine transform. In iOS, a positive value specifies counterclockwise rotation and a negative value specifies clockwise rotation. In Mac OS X, a positive value specifies clockwise rotation and a negative value specifies counterclockwise rotation.

Return Value

A new affine transformation matrix.

Discussion

You use this function to create a new affine transformation matrix by adding a rotation value to an existing affine transform. The resulting structure represents a new affine transform, which you can use (and reuse, if you want) to rotate a coordinate system.

The actual direction of rotation is dependent on the coordinate system orientation of the target platform, which is different in iOS and Mac OS X.

Availability**Related Sample Code**

MoviePlayer

Touches

Declared In

CGAffineTransform.h

CGAffineTransformScale

Returns an affine transformation matrix constructed by scaling an existing affine transform.

```
CGAffineTransform CGAffineTransformScale (
    CGAffineTransform t,
    CGFloat sx,
    CGFloat sy
);
```

Parameters*t*

An existing affine transform.

sx

The value by which to scale x values of the affine transform.

sy

The value by which to scale y values of the affine transform.

Return Value

A new affine transformation matrix.

Discussion

You use this function to create a new affine transformation matrix by adding scaling values to an existing affine transform. The resulting structure represents a new affine transform, which you can use (and reuse, if you want) to scale a coordinate system.

Availability**Related Sample Code**

Touches

Declared In

CGAffineTransform.h

CGAffineTransformTranslate

Returns an affine transformation matrix constructed by translating an existing affine transform.

```
CGAffineTransform CGAffineTransformTranslate (
    CGAffineTransform t,
    CGFloat tx,
    CGFloat ty
);
```

Parameters*t*

An existing affine transform.

tx

The value by which to move x values with the affine transform.

ty

The value by which to move y values with the affine transform.

Return Value

A new affine transformation matrix.

Discussion

You use this function to create a new affine transform by adding translation values to an existing affine transform. The resulting structure represents a new affine transform, which you can use (and reuse, if you want) to move a coordinate system.

Availability**Related Sample Code**

MoviePlayer

Declared In

CGAffineTransform.h

CGPointApplyAffineTransform

Returns the point resulting from an affine transformation of an existing point.

```
CGPoint CGPointApplyAffineTransform (
    CGPoint point,
    CGAffineTransform t
);
```

Parameters*point*

A point that specifies the x- and y-coordinates to transform.

t

The affine transform to apply.

Return Value

A new point resulting from applying the specified affine transform to the existing point.

Availability**Declared In**

CGAffineTransform.h

CGRectApplyAffineTransform

Applies an affine transform to a rectangle.

```
CGRect CGRectApplyAffineTransform (
    CGRect rect,
    CGAffineTransform t
);
```

Parameters

rect

The rectangle whose corner points you want to transform.

t

The affine transform to apply to the *rect* parameter.

Return Value

The transformed rectangle.

Discussion

Because affine transforms do not preserve rectangles in general, the function `CGRectApplyAffineTransform` returns the smallest rectangle that contains the transformed corner points of the *rect* parameter. If the affine transform *t* consists solely of scaling and translation operations, then the returned rectangle coincides with the rectangle constructed from the four transformed corners.

Availability

Declared In

`CGAffineTransform.h`

CSizeApplyAffineTransform

Returns the height and width resulting from a transformation of an existing height and width.

```
CSize CSizeApplyAffineTransform (
    CSize size,
    CGAffineTransform t
);
```

Parameters

size

A size that specifies the height and width to transform.

t

The affine transform to apply.

Return Value

A new size resulting from applying the specified affine transform to the existing size.

Availability

Declared In

`CGAffineTransform.h`

Data Types

CGAffineTransform

A structure for holding an affine transformation matrix.

```
struct CGAffineTransform {
    CGFloat a;
    CGFloat b;
    CGFloat c;
    CGFloat d;
    CGFloat tx;
    CGFloat ty;
};
typedef struct CGAffineTransform CGAffineTransform;
```

Fields

a	The entry at position [1,1] in the matrix.
b	The entry at position [1,2] in the matrix.
c	The entry at position [2,1] in the matrix.
d	The entry at position [2,2] in the matrix.
tx	The entry at position [3,1] in the matrix.
ty	The entry at position [3,2] in the matrix.

Discussion

In Quartz 2D, an affine transformation matrix is used to rotate, scale, translate, or skew the objects you draw in a graphics context. The `CGAffineTransform` type provides functions for creating, concatenating, and applying affine transformations.

In Quartz, affine transforms are represented by a 3 by 3 matrix:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Because the third column is always $(0, 0, 1)$, the `CGAffineTransform` data structure contains values for only the first two columns.

Conceptually, a Quartz affine transform multiplies a row vector representing each point (x,y) in your drawing by this matrix, producing a vector that represents the corresponding point (x',y') :

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \times \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Given the 3 by 3 matrix, Quartz uses the following equations to transform a point (x, y) in one coordinate system into a resultant point (x',y') in another coordinate system.

$$x' = ax + cy + t_x$$

$$y' = bx + dy + t_y$$

The matrix thereby “links” two coordinate systems—it specifies how points in one coordinate system map to points in another.

Note that you do not typically need to create affine transforms directly. If you want only to draw an object that is scaled or rotated, for example, it is not necessary to construct an affine transform to do so. The most direct way to manipulate your drawing—whether by movement, scaling, or rotation—is to call the functions [CGContextTranslateCTM](#) (page 132), [CGContextScaleCTM](#) (page 97), or [CGContextRotateCTM](#) (page 95), respectively. You should generally only create an affine transform if you want to reuse it later.

Availability

Available in iOS 2.0 and later.

Declared In

CGAffineTransform.h

Constants

CGAffineTransformIdentity

The identity transform.

```
const CGAffineTransform CGAffineTransformIdentity;
```

Constants

CGAffineTransformIdentity

$$\text{The identity transform: } \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Available in iOS 2.0 and later.

Declared in CGAffineTransform.h.

Declared In

CGAffineTransform.h

CGGeometry Reference

Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGGeometry.h
Companion guide	Quartz 2D Programming Guide

Overview

CGGeometry Reference defines structures for geometric primitives and functions that operate on them. The data structure `CGPoint` represents a point in a two-dimensional coordinate system. The data structure `CGRect` represents the location and dimensions of a rectangle. The data structure `CGSize` represents the dimensions of width and height.

The height and width stored in a `CGRect` data structure can be negative. For example, a rectangle with an origin of `[0.0, 0.0]` and a size of `[10.0, 10.0]` is exactly equivalent to a rectangle with an origin of `[10.0, 10.0]` and a size of `[-10.0, -10.0]`. Your application can standardize a rectangle—that is, ensure that the height and width are stored as positive values—by calling the `CGRectStandardize` function. All functions described in this reference that take `CGRect` data structures as inputs implicitly standardize those rectangles before calculating their results. For this reason, your applications should avoid directly reading and writing the data stored in the `CGRect` data structure. Instead, use the functions described here to manipulate rectangles and to retrieve their characteristics.

Functions by Task

Creating a Dictionary Representation from a Geometric Primitive

[CGPointCreateDictionaryRepresentation](#) (page 372)

Returns a dictionary representation of the specified point.

[CGSizeCreateDictionaryRepresentation](#) (page 387)

Returns a dictionary representation of the specified size.

[CGRectCreateDictionaryRepresentation](#) (page 375)

Returns a dictionary representation of the provided rectangle.

Creating a Geometric Primitive from a Dictionary Representation

[CGPointMakeWithDictionaryRepresentation](#) (page 373)

Fills in a point using the contents of the specified dictionary.

[CGSizeMakeWithDictionaryRepresentation](#) (page 389)

Fills in a size using the contents of the specified dictionary.

[CGRectMakeWithDictionaryRepresentation](#) (page 385)

Fills in a rectangle using the contents of the specified dictionary.

Creating a Geometric Primitive from Values

[CGPointMake](#) (page 373)

Returns a point with the specified coordinates.

[CGRectMake](#) (page 385)

Returns a rectangle with the specified coordinate and size values.

[CGSizeMake](#) (page 388)

Returns a size with the specified dimension values.

Modifying Rectangles

[CGRectDivide](#) (page 375)

Divides a source rectangle into two component rectangles.

[CGRectInset](#) (page 381)

Returns a rectangle that is smaller or larger than the source rectangle, with the same center point.

[CGRectIntegral](#) (page 382)

Returns the smallest rectangle that results from converting the source rectangle values to integers.

[CGRectIntersection](#) (page 382)

Returns the intersection of two rectangles.

[CGRectOffset](#) (page 386)

Returns a rectangle with an origin that is offset from that of the source rectangle.

[CGRectStandardize](#) (page 387)

Returns a rectangle with a positive width and height.

[CGRectUnion](#) (page 387)

Returns the smallest rectangle that contains the two source rectangles.

Comparing Values

[CGPointEqualToPoint](#) (page 372)

Returns whether two points are equal.

[CGSizeEqualToSize](#) (page 388)

Returns whether two sizes are equal.

[CGRectEqualToRect](#) (page 376)

Returns whether two rectangles are equal in size and position.

[CGRectIntersectsRect](#) (page 383)

Returns whether two rectangles intersect.

Checking for Membership

[CGRectContainsPoint](#) (page 374)

Returns whether a rectangle contains a specified point.

[CGRectContainsRect](#) (page 374)

Returns whether the first rectangle contains the second rectangle.

Getting Min, Mid, and Max Values

[CGRectGetMinX](#) (page 379)

Returns the x-coordinate that establishes the left edge of a rectangle.

[CGRectGetMinY](#) (page 380)

Returns the y-coordinate that establishes the bottom edge of a rectangle.

[CGRectGetMidX](#) (page 378)

Returns the x-coordinate that establishes the center of a rectangle.

[CGRectGetMidY](#) (page 379)

Returns the y-coordinate that establishes the center of a rectangle.

[CGRectGetMaxX](#) (page 377)

Returns the x-coordinate that establishes the right edge of a rectangle.

[CGRectGetMaxY](#) (page 377)

Returns the y-coordinate that establishes the top edge of a rectangle.

Getting Height and Width

[CGRectGetHeight](#) (page 376)

Returns the height of a rectangle.

[CGRectGetWidth](#) (page 380)

Returns the width of a rectangle.

Checking Rectangle Characteristics

[CGRectIsEmpty](#) (page 383)

Returns whether a rectangle has zero width or height, or is a null rectangle.

[CGRectIsNull](#) (page 384)

Returns whether the rectangle is equal to the null rectangle.

[CGRectIsInfinite](#) (page 384)

Returns whether a rectangle is infinite.

Functions

CGPointCreateDictionaryRepresentation

Returns a dictionary representation of the specified point.

```
CFDictionaryRef CGPointCreateDictionaryRepresentation(  
    CGPoint point  
);
```

Parameters

point
A point.

Return Value

The dictionary representation of the point.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGPointEqualToPoint

Returns whether two points are equal.

```
bool CGPointEqualToPoint (  
    CGPoint point1,  
    CGPoint point2  
);
```

Parameters

point1
The first point to examine.

point2
The second point to examine.

Return Value

true if the two specified points are the same; otherwise, false.

Availability

Available in iOS 2.0 and later.

Related Sample Code

CopyPasteTile

Touches

Declared In

CGGeometry.h

CGPointMake

Returns a point with the specified coordinates.

```
CGPoint CGPointMake (  
    CGFloat x,  
    CGFloat y  
);
```

Parameters

x

The x-coordinate of the point to construct.

y

The y-coordinate of the point to construct.

Return Value

A point.

Availability

Available in iOS 2.0 and later.

Related Sample Code

CopyPasteTile

GKRocket

iAdSuite

QuartzDemo

TheElements

Declared In

CGGeometry.h

CGPointMakeWithDictionaryRepresentation

Fills in a point using the contents of the specified dictionary.

```
bool CGPointMakeWithDictionaryRepresentation(  
    CFDictionaryRef dict,  
    CGPoint *point  
);
```

Parameters

dict

A dictionary that was previously returned from the function [CGPointCreateDictionaryRepresentation](#) (page 372).

point

On return, the point created from the provided dictionary.

Return Value

true if successful; otherwise false.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGRectContainsPoint

Returns whether a rectangle contains a specified point.

```
bool CGRectContainsPoint (
    CGRect rect,
    CGPoint point
);
```

Parameters*rect*

The rectangle to examine.

point

The point to examine.

Return Value

true if the rectangle is not null or empty and the point is located within the rectangle; otherwise, false.

Discussion

A point is considered inside the rectangle if its coordinates lie inside the rectangle or on the minimum X or minimum Y edge.

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch

Touches

Declared In

CGGeometry.h

CGRectContainsRect

Returns whether the first rectangle contains the second rectangle.

```
bool CGRectContainsRect (
    CGRect rect1,
    CGRect rect2
);
```

Parameters*rect1*

The rectangle to examine for containment of the rectangle passed in *rect2*.

rect2

The rectangle to examine for being contained in the rectangle passed in *rect1*.

Return Value

true if the rectangle specified by *rect2* is contained in the rectangle passed in *rect1*; otherwise, false. The first rectangle contains the second if the union of the two rectangles is equal to the first rectangle.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGRectCreateDictionaryRepresentation

Returns a dictionary representation of the provided rectangle.

```
CFDictionaryRef CGRectCreateDictionaryRepresentation(
    CGRect rect
);
```

Parameters

rect

A rectangle.

Return Value

The dictionary representation of the rectangle.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGRectDivide

Divides a source rectangle into two component rectangles.

```
void CGRectDivide (
    CGRect rect,
    CGRect *slice,
    CGRect *remainder,
    CGFloat amount,
    CGRectEdge edge
);
```

Parameters

rect

The source rectangle.

slice

On input, a pointer to an uninitialized rectangle. On return, the rectangle is filled in with the specified edge and values that extends the distance beyond the edge specified by the *amount* parameter.

remainder

On input, a pointer to an uninitialized rectangle. On return, the rectangle contains the portion of the source rectangle that remains after `CGRectEdge` produces the “slice” rectangle.

amount

A distance from the rectangle side that is specified in the *edge* parameter. This distance defines the line, parallel to the specified side, that Quartz uses to divide the source rectangle.

edge

An edge value that specifies the side of the rectangle from which the distance passed in the *amount* parameter is measured. `CGRectDivide` produces a “slice” rectangle that contains the specified edge and extends *amount* distance beyond it.

Availability

Available in iOS 2.0 and later.

Declared In

`CGGeometry.h`

CGRectEqualToRect

Returns whether two rectangles are equal in size and position.

```
bool CGRectEqualToRect (
    CGRect rect1,
    CGRect rect2
);
```

Parameters

rect1

The first rectangle to examine.

rect2

The second rectangle to examine.

Return Value

`true` if the two specified rectangles have equal size and origin values, or if both rectangles are null rectangles. Otherwise, `false`.

Availability

Available in iOS 2.0 and later.

Declared In

`CGGeometry.h`

CGRectGetHeight

Returns the height of a rectangle.

```
CGFloat CGRectGetHeight (
    CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value

The height of the specified rectangle.

Discussion

Regardless of whether the height is stored in the `CGRect` data structure as a positive or negative number, this function returns the height as if the rectangle were standardized. That is, the result is never a negative number.

Availability

Available in iOS 2.0 and later.

Related Sample Code

`aurioTouch`

`Formulaic`

`oalTouch`

`PhotoPicker`

Declared In

`CGGeometry.h`

CGRectGetMaxX

Returns the x-coordinate that establishes the right edge of a rectangle.

```
CGFloat CGRectGetMaxX (  
    CGRect rect  
);
```

Parameters

rect

The rectangle to examine.

Return Value

The x-coordinate of the top-right corner of the specified rectangle.

Availability

Available in iOS 2.0 and later.

Related Sample Code

`avTouch`

`HeadsUpUI`

`PhotoScroller`

`ScrollViewSuite`

`SpeakHere`

Declared In

`CGGeometry.h`

CGRectGetMaxY

Returns the y-coordinate that establishes the top edge of a rectangle.

```
CGFloat CGRectGetMaxY (  
    CGRect rect  
);
```

Parameters

rect

The rectangle to examine.

Return Value

The y-coordinate of the top-right corner of the specified rectangle.

Availability

Available in iOS 2.0 and later.

Related Sample Code

avTouch

GKRocket

iAdSuite

ScrollViewSuite

SpeakHere

Declared In

CGGeometry.h

CGRectGetMidX

Returns the x- coordinate that establishes the center of a rectangle.

```
CGFloat CGRectGetMidX (  
    CGRect rect  
);
```

Parameters

rect

The rectangle to examine.

Return Value

The x-coordinate of the center of the specified rectangle.

Availability

Available in iOS 2.0 and later.

Related Sample Code

GKRocket

iAdSuite

PhotoScroller

QuartzDemo

ScrollViewSuite

Declared In

CGGeometry.h

CGRectGetMidY

Returns the y-coordinate that establishes the center of a rectangle.

```
CGFloat CGRectGetMidY (  
    CGRect rect  
);
```

Parameters

rect

The rectangle to examine.

Return Value

The y-coordinate of the center of the specified rectangle.

Availability

Available in iOS 2.0 and later.

Related Sample Code

Formulaic

GKRocket

PhotoScroller

QuartzDemo

Declared In

CGGeometry.h

CGRectGetMinX

Returns the x-coordinate that establishes the left edge of a rectangle.

```
CGFloat CGRectGetMinX (  
    CGRect rect  
);
```

Parameters

rect

The rectangle to examine.

Return Value

The x-coordinate of the bottom-left corner of the specified rectangle.

Availability

Available in iOS 2.0 and later.

Related Sample Code

avTouch

HeadsUpUI

PhotoScroller

ScrollViewSuite

SpeakHere

Declared In

CGGeometry.h

CGRectGetMinY

Returns the y-coordinate that establishes the bottom edge of a rectangle.

```
CGFloat CGRectGetMinY (  
    CGRect rect  
);
```

Parameters

rect

The rectangle to examine.

Return Value

The y-coordinate of the bottom-left corner of the specified rectangle.

Availability

Available in iOS 2.0 and later.

Related Sample Code

avTouch

HeadsUpUI

oalTouch

PhotoScroller

SpeakHere

Declared In

CGGeometry.h

CGRectGetWidth

Returns the width of a rectangle.

```
CGFloat CGRectGetWidth (  
    CGRect rect  
);
```

Parameters

rect

The rectangle to examine.

Return Value

The width of the specified rectangle.

Discussion

Regardless of whether the width is stored in the `CGRect` data structure as a positive or negative number, this function returns the width as if the rectangle were standardized. That is, the result is never a negative number.

Availability

Available in iOS 2.0 and later.

Related Sample Code

aurioTouch

Formulaic

oalTouch

PhotoPicker
PhotoScroller

Declared In
CGGeometry.h

CGRectInset

Returns a rectangle that is smaller or larger than the source rectangle, with the same center point.

```
CGRect CGRectInset (  
    CGRect rect,  
    CGFloat dx,  
    CGFloat dy  
);
```

Parameters

rect

The source CGRect structure.

dx

The x-coordinate value to use for adjusting the source rectangle. To create an inset rectangle, specify a positive value. To create a larger, encompassing rectangle, specify a negative value.

dy

The y-coordinate value to use for adjusting the source rectangle. To create an inset rectangle, specify a positive value. To create a larger, encompassing rectangle, specify a negative value.

Return Value

A rectangle. The origin value is offset in the x-axis by the distance specified by the *dx* parameter and in the y-axis by the distance specified by the *dy* parameter, and its size adjusted by $(2*dx, 2*dy)$, relative to the source rectangle. If *dx* and *dy* are positive values, then the rectangle's size is decreased. If *dx* and *dy* are negative values, the rectangle's size is increased.

Discussion

The rectangle is standardized and then the inset parameters are applied. If the resulting rectangle would have a negative height or width, a null rectangle is returned.

Availability

Available in iOS 2.0 and later.

Related Sample Code

avTouch
oalTouch
QuartzDemo
SpeakHere
WiTap

Declared In
CGGeometry.h

CGRectIntegral

Returns the smallest rectangle that results from converting the source rectangle values to integers.

```
CGRect CGRectIntegral (  
    CGRect rect  
);
```

Parameters

rect

The source rectangle.

Return Value

A rectangle with the smallest integer values for its origin and size that contains the source rectangle. That is, given a rectangle with fractional origin or size values, `CGRectIntegral` rounds the rectangle's origin downward and its size upward to the nearest whole integers, such that the result contains the original rectangle.

Availability

Available in iOS 2.0 and later.

Declared In

`CGGeometry.h`

CGRectIntersection

Returns the intersection of two rectangles.

```
CGRect CGRectIntersection (  
    CGRect r1,  
    CGRect r2  
);
```

Parameters

rect1

The first source rectangle.

rect2

The second source rectangle.

Return Value

A rectangle that represents the intersection of the two specified rectangles. If the two rectangles do not intersect, returns the null rectangle. To check for this condition, use [CGRectIsNull](#) (page 384).

Discussion

Both rectangles are standardized prior to calculating the intersection.

Availability

Available in iOS 2.0 and later.

Related Sample Code

[PhotoScroller](#)

Declared In

`CGGeometry.h`

CGRectIntersectsRect

Returns whether two rectangles intersect.

```
bool CGRectIntersectsRect (
    CGRect rect1,
    CGRect rect2
);
```

Parameters

rect1

The first rectangle to examine.

rect2

The second rectangle to examine.

Return Value

`true` if the two specified rectangles intersect; otherwise, `false`. The first rectangle intersects the second if the intersection of the rectangles is not equal to the null rectangle.

Availability

Available in iOS 2.0 and later.

Related Sample Code

AccelerometerGraph

GKRocket

GKTank

ScrollViewSuite

Declared In

CGGeometry.h

CGRectIsEmpty

Returns whether a rectangle has zero width or height, or is a null rectangle.

```
bool CGRectIsEmpty (
    CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value

`true` if the specified rectangle is empty; otherwise, `false`.

Discussion

An empty rectangle is either a null rectangle or a valid rectangle with zero height or width.

Availability

Available in iOS 2.0 and later.

See Also

[CGRectIsNull](#) (page 384)

Declared In

CGGeometry.h

CGRectIsInfinite

Returns whether a rectangle is infinite.

```
bool CGRectIsInfinite (
    CGRect rect
);
```

Parameters*rect*

The rectangle to examine.

Return Value

Returns `true` if the specified rectangle is infinite; otherwise, `false`.

Discussion

An infinite rectangle is one that has no defined bounds. Infinite rectangles can be created as output from a tiling filter. For example, the Core Image framework perspective tile filter creates an image whose extent is described by an infinite rectangle.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGRectIsNull

Returns whether the rectangle is equal to the null rectangle.

```
bool CGRectIsNull (
    CGRect rect
);
```

Parameters*rect*

The rectangle to examine.

Return Value

`true` if the specified rectangle is null; otherwise, `false`.

Discussion

A null rectangle is the equivalent of an empty set. For example, the result of intersecting two disjoint rectangles is a null rectangle. A null rectangle cannot be drawn and interacts with other rectangles in special ways.

Availability

Available in iOS 2.0 and later.

See Also

[CGRectIsEmpty](#) (page 383)

Declared In

CGGeometry.h

CGRectMake

Returns a rectangle with the specified coordinate and size values.

```
CGRect CGRectMake (
    CGFloat x,
    CGFloat y,
    CGFloat width,
    CGFloat height
);
```

Parameters

x
The x-coordinate of the rectangle's origin point.

y
The y-coordinate of the rectangle's origin point.

width
The width of the rectangle.

height
The height of the rectangle.

Return Value

A rectangle with the specified location and dimensions.

Availability

Available in iOS 2.0 and later.

Related Sample Code

avTouch
iPhoneCoreDataRecipes
QuartzDemo
SpeakHere
UICatalog

Declared In

CGGeometry.h

CGRectMakeWithDictionaryRepresentation

Fills in a rectangle using the contents of the specified dictionary.

```
bool CGRectMakeWithDictionaryRepresentation(
    CFDictionaryRef dict,
    CGRect *rect
);
```

Parameters*dict*

A dictionary that was previously returned from the function [CGRectCreateDictionaryRepresentation](#) (page 375).

rect

On return, the rectangle created from the specified dictionary.

Return Value

true if successful; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGRectOffset

Returns a rectangle with an origin that is offset from that of the source rectangle.

```
CGRect CGRectOffset (
    CGRect rect,
    CGFloat dx,
    CGFloat dy
);
```

Parameters*rect*

The source rectangle.

dx

The offset value for the x-coordinate.

dy

The offset value for the y-coordinate.

Return Value

A rectangle that is the same size as the source, but with its origin offset by *dx* units along the x-axis and *dy* units along the y-axis with respect to the source.

Availability

Available in iOS 2.0 and later.

Related Sample Code

TheElements

Declared In

CGGeometry.h

CGRectStandardize

Returns a rectangle with a positive width and height.

```
CGRect CGRectStandardize (  
    CGRect rect  
);
```

Parameters

rect

The source rectangle.

Return Value

A rectangle that represents the source rectangle, but with positive width and height values.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGRectUnion

Returns the smallest rectangle that contains the two source rectangles.

```
CGRect CGRectUnion (  
    CGRect r1,  
    CGRect r2  
);
```

Parameters

r1

The first source rectangle.

r2

The second source rectangle.

Return Value

The smallest rectangle that completely contains both of the source rectangles.

Discussion

Both rectangles are standardized prior to calculating the union. If either of the rectangles is a null rectangle, a copy of the other rectangle is returned (resulting in a null rectangle if both rectangles are null). Otherwise a rectangle that completely contains the source rectangles is returned.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGSizeCreateDictionaryRepresentation

Returns a dictionary representation of the specified size.

```
CFDictionaryRef CGSizeCreateDictionaryRepresentation(  
    CGSize size  
);
```

Parameters

size

A size.

Return Value

The dictionary representation of the size.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGSizeEqualToSize

Returns whether two sizes are equal.

```
bool CGSizeEqualToSize (  
    CGSize size1,  
    CGSize size2  
);
```

Parameters

size1

The first size to examine.

size2

The second size to examine.

Return Value

true if the two specified sizes are equal; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGSizeMake

Returns a size with the specified dimension values.

```
CGSize CGSizeMake (  
    CGFloat width,  
    CGFloat height  
);
```

Parameters

width

A width value.

height

A height value.

Return Value

Returns a `CGSize` structure with the specified width and height.

Availability

Available in iOS 2.0 and later.

Related Sample Code

GKRocket

LazyTableImages

MapCallouts

TheElements

ZoomingPDFViewer

Declared In

`CGGeometry.h`

CGSizeMakeWithDictionaryRepresentation

Fills in a size using the contents of the specified dictionary.

```
bool CGSizeMakeWithDictionaryRepresentation(
    CFDictionaryRef dict,
    CGSize *size
);
```

Parameters

dict

A dictionary that was previously returned from the function [CGSizeCreateDictionaryRepresentation](#) (page 387).

size

On return, the size created from the specified dictionary.

Return Value

true if successful; otherwise, false.

Availability

Available in iOS 2.0 and later.

Declared In

`CGGeometry.h`

Data Types

CGFloat

The basic type for all floating-point values.

```
typedef float CGFloat; // 32-bit
```

```
typedef double CGFloat; // 64-bit
```

Availability

Available in iOS 4.0 and later.

Declared In

CGBase.h

CGPoint

A structure that contains a point in a two-dimensional coordinate system.

```
struct CGPoint {
    CGFloat x;
    CGFloat y;
};
typedef struct CGPoint CGPoint;
```

Fields

x
The x-coordinate of the point.

y
The y-coordinate of the point.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGRect

A structure that contains the location and dimensions of a rectangle.

```
struct CGRect {
    CGPoint origin;
    CGSize size;
};
typedef struct CGRect CGRect;
```

Fields

origin
A point that specifies the coordinates of the rectangle's origin.

size
A size that specifies the height and width of the rectangle.

Discussion

In the default Quartz coordinate space, the origin is located in the lower-left corner of the rectangle and the rectangle extends towards the upper-right corner. If the context has a flipped-coordinate space—often the case on iOS—the origin is in the upper-left corner and the rectangle extends towards the lower-right corner.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

CGSize

A structure that contains width and height values.

```
struct CGSize {
    CGFloat width;
    CGFloat height;
};
typedef struct CGSize CGSize;
```

Fields

width

A width value.

height

A height value.

Availability

Available in iOS 2.0 and later.

Declared In

CGGeometry.h

Constants

CGRectInfinite

A rectangle that has infinite extent.

```
const CGRect CGRectInfinite;
```

Constants

CGRectInfinite

A rectangle that has infinite extent.

Available in iOS 2.0 and later.

Declared in CGGeometry.h.

Availability

Available in Mac OS X v10.4 and later.

Declared In

CGGeometry.h

Geometric Zeros

A zero point, zero rectangle, or zero size.

```
const CGPoint CGPointZero;
const CGRect CGRectZero;
const CGSize CGSizeZero;
```

Constants**CGPointZero**

A point constant with location (0,0). The zero point is equivalent to `CGPointMake(0,0)`.

Available in iOS 2.0 and later.

Declared in `CGGeometry.h`.

CGRectZero

A rectangle constant with location (0,0), and width and height of 0. The zero rectangle is equivalent to `CGRectMake(0,0,0,0)`.

Available in iOS 2.0 and later.

Declared in `CGGeometry.h`.

CGSizeZero

A size constant with width and height of 0. The zero size is equivalent to `CGSizeMake(0,0)`.

Available in iOS 2.0 and later.

Declared in `CGGeometry.h`.

Declared In

`CGGeometry.h`

Geometrical Null

The null or empty rectangle.

```
const CGRect CGRectNull;
```

Constants**CGRectNull**

The null rectangle. This is the rectangle returned when, for example, you intersect two disjoint rectangles. Note that the null rectangle is not the same as the zero rectangle. For example, the union of a rectangle with the null rectangle is the original rectangle (that is, the null rectangle contributes nothing).

Available in iOS 2.0 and later.

Declared in `CGGeometry.h`.

Declared In

`CGGeometry.h`

CGRectEdge

Coordinates that establish the edges of a rectangle.


```
enum CGRectEdge {
    CGRectMinXEdge,
    CGRectMinYEdge,
    CGRectMaxXEdge,
    CGRectMaxYEdge
};
typedef enum CGRectEdge CGRectEdge;
```

Constants**CGRectMinXEdge**

The x-coordinate that establishes the left edge of a rectangle.

Available in iOS 2.0 and later.

Declared in `CGGeometry.h`.

CGRectMinYEdge

The y-coordinate that establishes the minimum edge of a rectangle. In Mac OS X, this is typically the bottom edge of the rectangle. If the coordinate system is flipped (or if you are using the default coordinate system in iOS), this constant refers to the top edge of the rectangle.

Available in iOS 2.0 and later.

Declared in `CGGeometry.h`.

CGRectMaxXEdge

The x-coordinate that establishes the right edge of a rectangle.

Available in iOS 2.0 and later.

Declared in `CGGeometry.h`.

CGRectMaxYEdge

The y-coordinate that establishes the maximum edge of a rectangle. In Mac OS X, this is typically the top edge of the rectangle. If the coordinate system is flipped (or if you are using the default coordinate system in iOS), this constant refers to the bottom edge of the rectangle.

Available in iOS 2.0 and later.

Declared in `CGGeometry.h`.

Declared In

`CGGeometry.h`

CGFloat Informational Macros

Informational macros for the `CGFloat` type.

```
#define CGFLOAT_MIN FLT_MIN // 32-bit
#define CGFLOAT_MAX FLT_MAX
#define CGFLOAT_IS_DOUBLE 0

#define CGFLOAT_MIN DBL_MIN // 64-bit
#define CGFLOAT_MAX DBL_MAX
#define CGFLOAT_IS_DOUBLE 1
```

Constants**CGFLOAT_MIN**

The minimum allowable value for a `CGFloat` type. For 32-bit code, this value is $1.17549435 \times 10^{-38}$. For 64-bit code, it is $2.2250738585072014 \times 10^{-308}$.

Available in iOS 2.0 and later.

Declared in `CABase.h`.

CGFLOAT_MAX

The maximum allowable value for a `CGFloat` type. For 32-bit code, this value is $3.40282347 \times 10^{38}$. For 64-bit code, it is $1.7976931348623157 \times 10^{308}$.

Available in iOS 2.0 and later.

Declared in `CABase.h`.

CGFLOAT_IS_DOUBLE

Indicates whether `CGFloat` is defined as a `float` or `double` type.

Available in iOS 2.0 and later.

Declared in `CABase.h`.

Document Revision History

This table describes the changes to *Core Graphics Framework Reference*.

Date	Notes
2009-05-14	Added Core Graphics Constants Reference.
2008-05-02	New document that describes the classes in the Core Graphics framework.

