CGContext Reference

Graphics & Imaging > Quartz

2008-04-08

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

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

CGContext Reference

Derived From:	*CFType Reference*
Framework:	ApplicationServices/ApplicationServices.h
Companion guide	Quartz 2D Programming Guide
Declared in	CGContext.h

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 Carbon and Cocoa frameworks provide 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 41)
  Forces all pending drawing operations in a window context to be rendered immediately to the destination device.

`CGContextGetTypeID` (page 45)
  Returns the type identifier for Quartz graphics contexts.

`CGContextRelease` (page 48)
  Decrements the retain count of a graphics context.

`CGContextRetain` (page 49)
  Increments the retain count of a graphics context.

`CGContextSynchronize` (page 82)
  Marks a window context for update.

Saving and Restoring the Current Graphics State

`CGContextSaveGState` (page 50)
: Pushes a copy of the current graphics state onto the graphics state stack for the context.

`CGContextRestoreGState` (page 49)
: Sets the current graphics state to the state most recently saved.

Getting and Setting Graphics State Parameters

`CGContextGetInterpolationQuality` (page 43)
: Returns the current level of interpolation quality for a graphics context.

`CGContextSetFlatness` (page 59)
: Sets the accuracy of curved paths in a graphics context.

`CGContextSetInterpolationQuality` (page 62)
: Sets the level of interpolation quality for a graphics context.

`CGContextSetLineCap` (page 62)
: Sets the style for the endpoints of lines drawn in a graphics context.

`CGContextSetLineDash` (page 63)
: Sets the pattern for dashed lines in a graphics context.

`CGContextSetLineJoin` (page 64)
: Sets the style for the joins of connected lines in a graphics context.

`CGContextSetLineWidth` (page 64)
: Sets the line width for a graphics context.

`CGContextSetMiterLimit` (page 65)
: Sets the miter limit for the joins of connected lines in a graphics context.

`CGContextSetPatternPhase` (page 65)
: Sets the pattern phase of a context.

`CGContextSetFillPattern` (page 58)
: Sets the fill pattern in the specified graphics context.

`CGContextSetRenderingIntent` (page 66)
: Sets the rendering intent in the current graphics state.

`CGContextSetShouldAntialias` (page 70)
: Sets anti-aliasing on or off for a graphics context.

`CGContextSetShouldSmoothFonts` (page 70)
: Enables or disables font smoothing in a graphics context.

`CGContextSetStrokePattern` (page 72)
: Sets the stroke pattern in the specified graphics context.

`CGContextSetBlendMode` (page 53)
: Sets how Quartz composites sample values for a graphics context.

`CGContextSetAllowsAntialiasing` (page 52)
: Sets whether or not to allow anti-aliasing for a graphics context.

Constructing Paths

These functions are used to define the geometry of the current path.

`CGContextAddArc` (page 14)
Adds an arc of a circle to the current path, using a center point, radius, and end point.

`CGContextAddArcToPoint` (page 15)
Adds an arc of a circle to the current path, using a radius and tangent points.

`CGContextAddCurveToPoint` (page 16)
Appends a cubic Bézier curve from the current point, using the provided control points and end point .

`CGContextAddLines` (page 18)
Adds a sequence of connected straight-line segments to the current path.

`CGContextAddLineToPoint` (page 19)
Appends a straight line segment from the current point to the provided point .

`CGContextAddPath` (page 19)
Adds a previously created Quartz path object to the current path in a graphics context.

`CGContextAddQuadCurveToPoint` (page 20)
Appends a quadratic Bézier curve from the current point, using a control point and an end point you specify.

`CGContextAddRect` (page 21)
Adds a rectangular path to the current path.

`CGContextAddRects` (page 21)
Adds a set rectangular paths to the current path.

`CGContextBeginPath` (page 22)
Creates a new empty path in a graphics context.

`CGContextClosePath` (page 27)
Closes and terminates an open path.

`CGContextMoveToPoint` (page 46)
Begins a new path at the point you specify.

`CGContextAddEllipseInRect` (page 17)
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 24)
Paints a transparent rectangle.

`CGContextDrawPath` (page 33)
Draws the current path using the provided drawing mode.

`CGContextEOFillPath` (page 39)
Paints the area within the current path, using the even-odd fill rule.

`CGContextFillPath` (page 40)
Paints the area within the current path, using the nonzero winding number rule.

`CGContextFillRect` (page 40)
Paints the area contained within the provided rectangle, using the fill color in the current graphics state.

`CGContextFillRects` (page 41)
Paints the areas contained within the provided rectangles, using the fill color in the current graphics state.

`CGContextFillEllipseInRect` (page 39)
Paints the area of the ellipse that fits inside the provided rectangle, using the fill color in the current graphics state.

`CGContextStrokePath` (page 80)
Paints a line along the current path.

`CGContextStrokeRect` (page 80)
Paints a rectangular path.

`CGContextStrokeRectWithWidth` (page 81)
Paints a rectangular path, using the specified line width.

`CGContextReplacePathWithStrokedPath` (page 48)
Replaces the path in the graphics context with the stroked version of the path.

`CGContextStrokeEllipseInRect` (page 79)
Strokes an ellipse that fits inside the specified rectangle.

`CGContextStrokeLineSegments` (page 79)
Strokes a sequence of line segments.

Getting Information About Paths

`CGContextIsPathEmpty` (page 46)
Indicates whether the current path contains any subpaths.

`CGContextGetPathCurrentPoint` (page 44)
Returns the current point in a non-empty path.

`CGContextGetPathBoundingBox` (page 43)
Returns the smallest rectangle that contains the current path.

`CGContextPathContainsPoint` (page 47)
Checks to see whether the specified point is contained in the current path.

Modifying Clipping Paths

`CGContextClip` (page 25)
Modifies the current clipping path, using the nonzero winding number rule.

`CGContextEOClip` (page 38)
Modifies the current clipping path, using the even-odd rule.

`CGContextClipToRect` (page 26)
Sets the clipping path to the intersection of the current clipping path with the area defined by the specified rectangle.

CGContextClipToRects (page 27)
Sets the clipping path to the intersection of the current clipping path with the region defined by an array of rectangles.

CGContextGetClipBoundingBox (page 42)
Returns the bounding box of a clipping path.

CGContextClipToMask (page 25)
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 53)
Sets the opacity level for objects drawn in a graphics context.

CGContextSetCMYKFillColor (page 54)
Sets the current fill color to a value in the DeviceCMYK color space.

CGContextSetFillColor (page 57)
Sets the current fill color.

CGContextSetCMYKStrokeColor (page 56)
Sets the current stroke color to a value in the DeviceCMYK color space.

CGContextSetFillColorSpace (page 57)
Sets the fill color space in a graphics context.

CGContextSetFillColorWithColor (page 58)
Sets the current fill color in a graphics context, using a Quartz color.

CGContextSetGrayFillColor (page 60)
Sets the current fill color to a value in the DeviceGray color space.

CGContextSetGrayStrokeColor (page 61)
Sets the current stroke color to a value in the DeviceGray color space.

CGContextSetRGBFillColor (page 66)
Sets the current fill color to a value in the DeviceRGB color space.

CGContextSetRGBStrokeColor (page 67)
Sets the current stroke color to a value in the DeviceRGB color space.

CGContextSetShadow (page 68)
Enables shadowing in a graphics context.

CGContextSetShadowWithColor (page 69)
Enables shadowing with color a graphics context.

CGContextSetStrokeColor (page 71)
Sets the current stroke color.

CGContextSetStrokeColorSpace (page 71)
Sets the stroke color space in a graphics context.

CGContextSetStrokeColorWithColor (page 72)
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 28)
Transforms the user coordinate system in a context using a specified matrix.

`CGContextGetCTM` (page 42)
Returns the current transformation matrix.

`CGContextRotateCTM` (page 50)
Rotates the user coordinate system in a context.

`CGContextScaleCTM` (page 51)
Changes the scale of the user coordinate system in a context.

`CGContextTranslateCTM` (page 82)
Changes the origin of the user coordinate system in a context.

Using Transparency Layers

`CGContextBeginTransparencyLayer` (page 23)
Begins a transparency layer.

`CGContextBeginTransparencyLayerWithRect` (page 24)
Begins a transparency layer whose contents are bounded by the specified rectangle.

`CGContextEndTransparencyLayer` (page 38)
Ends a transparency layer.

Drawing an Image to a Graphics Context

`CGContextDrawTiledImage` (page 36)
Repeatedly draws an image, scaled to the provided rectangle, to fill the current clip region.

`CGContextDrawImage` (page 32)
Draws an image into a graphics context.

Drawing PDF Content to a Graphics Context

`CGContextDrawPDFDocument` (page 34)
Draws a page of a PDF document into a graphics context.

`CGContextDrawPDFPage` (page 34)
Draws a page in the current user space of a PDF context.

Drawing With a Gradient

`CGContextDrawLinearGradient` (page 32)
Paints a gradient fill that varies along the line defined by the provided starting and ending points.

`CGContextDrawRadialGradient` (page 35)
: Paints a gradient fill that varies along the area defined by the provided starting and ending circles.

Drawing With a Shading

`CGContextDrawShading` (page 36)
: Fills the clipping path of a context with the specified shading.

Setting Up a Page-Based Graphics Context

`CGContextBeginPage` (page 22)
: Starts a new page in a page-based graphics context.

`CGContextEndPage` (page 37)
: Ends the current page in a page-based graphics context.

Drawing Glyphs

`CGContextShowGlyphs` (page 75)
: Displays an array of glyphs at the current text position.

`CGContextShowGlyphsAtPoint` (page 75)
: Displays an array of glyphs at a position you specify.

`CGContextShowGlyphsWithAdvances` (page 76)
: Draws an array of glyphs with varying offsets.

`CGContextShowGlyphsAtPositions` (page 76)
: Draws glyphs at the provided position.

Drawing Text

`CGContextGetTextMatrix` (page 44)
: Returns the current text matrix.

`CGContextGetTextPosition` (page 45)
: Returns the location at which text is drawn.

`CGContextSelectFont` (page 52)
: Sets the font and font size in a graphics context.

`CGContextSetCharacterSpacing` (page 54)
: Sets the current character spacing.

`CGContextSetFont` (page 59)
: Sets the platform font in a graphics context.

`CGContextSetFontSize` (page 60)
: Sets the current font size.

`CGContextSetTextDrawingMode` (page 73)
: Sets the current text drawing mode.

`CGContextSetTextMatrix` (page 73)
    Sets the current text matrix.

`CGContextSetTextPosition` (page 74)
    Sets the location at which text is drawn.

`CGContextShowText` (page 77)
    Displays a character array at the current text position, a point specified by the current text matrix.

`CGContextShowTextAtPoint` (page 78)
    Displays a character string at a position you specify.

Converting Between Device Space and User Space

`CGContextGetUserSpaceToDeviceSpaceTransform` (page 46)
    Returns an affine transform that maps user space coordinates to device space coordinates.

`CGContextConvertPointToDeviceSpace` (page 29)
    Returns a point that is transformed from user space coordinates to device space coordinates.

`CGContextConvertPointToUserSpace` (page 29)
    Returns a point that is transformed from device space coordinates to user space coordinates.

`CGContextConvertSizeToDeviceSpace` (page 31)
    Returns a size that is transformed from user space coordinates to device space coordinates.

`CGContextConvertSizeToUserSpace` (page 31)
    Returns a size that is transformed from device space coordinates to user space coordinates

`CGContextConvertRectToDeviceSpace` (page 30)
    Returns a rectangle that is transformed from user space coordinate to device space coordinates.

`CGContextConvertRectToUserSpace` (page 30)
    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, using a center point, radius, and end point.

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

Parameters

`context`
    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

Pass `1` to draw the arc clockwise; `0` otherwise.

Discussion

When you call this function, Quartz builds an arc of a circle centered on the point you provide. The arc is of the specified radius and extends between the start and end point. (You can also use `CGContextAddArc` as a convenient way to draw a circle, by setting the start point to `0` and the end point to `2*Pi`.)

If the current path already contains a subpath, Quartz additionally appends a straight line segment from the current point to the starting point of the arc. If the current path is empty, Quartz creates a new subpath for the arc and does not add the initial straight line segment.

After adding the arc, the current point is reset to the end point of arc (the second tangent point).

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextAddArcToPoint` (page 15)

Related Sample Code

CarbonSketch

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

context

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 draws 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. After adding the arc, the current point is reset to the end point of arc (the second tangent point).

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextAddArc` (page 14)

`CGContextAddArcToPoint` (page 15)

Related Sample Code

CarbonSketch

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

context

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. After adding the segment, the current point is reset from the beginning of the new segment to the end point of that segment.

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextAddQuadCurveToPoint` (page 20)

`CGContextAddArcToPoint` (page 15)

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.

Availability

Available in Mac OS X v10.4 and later.

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

context

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 is 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 the current path in a graphics context. Quartz connects each point in the array with the subsequent point in the array, using straight line segments.

On return, the current point is the last point in the array. This function does not automatically close the path created by the line segments. If you want to close the path, you must call `CGContextClosePath` (page 27).

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextAddLineToPoint` (page 19)

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

context

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 reset from the beginning of the new line segment to the endpoint of that line segment.

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextAddLines` (page 18)

Related Sample Code

CALayerEssentials

CarbonSketch

HID Calibrator

HID Explorer

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

Quartz applies the current transformation matrix (CTM) to the points in the new path before they are added to the current path in the graphics context.

Availability

Available in Mac OS X version 10.2 and later.

Related Sample Code

CALayerEssentials

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

context

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 reset from the beginning of the new segment to the end point of that segment.

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextAddCurveToPoint` (page 16)

`CGContextAddArcToPoint` (page 15)

Declared In

`CGContext.h`

CGContextAddRect

Adds a rectangular path to the current path.

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

Parameters

context

A graphics context.

rect

A rectangle, specified in user space coordinates.

Availability

Available in Mac OS X version 10.0 and later.

See Also

CGContextAddRects (page 21)

Related Sample Code

CarbonSketch

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

context

A graphics context.

rects

An array of rectangles, specified in user space coordinates.

count

The number of rectangles in the rects array.

Availability

Available in Mac OS X version 10.0 and later.

See Also

CGContextAddRect (page 21)

Declared In

CGContext.h

CGContextBeginPage

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

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

Parameters

context

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 37) 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 Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextBeginPath

Creates a new empty path in a graphics context.

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

Parameters

context

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 replaces the previous current path with the new path. In this case, 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 Mac OS X version 10.0 and later.

See Also
`CGContextClosePath` (page 27)

Related Sample Code
CarbonSketch
HID Calibrator
HID Explorer

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 38), 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 Mac OS X version 10.3 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 23) except that the content of the transparency layer is within the bounds of the provided rectangle.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CGContext.h`

CGContextClearRect

Paints a transparent rectangle.

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

Parameters

`context`
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 Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In

`CGContext.h`

CGContextClip

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

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

Parameters

`context`

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.

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.

See also`CGContextEOClip` (page 38)

Availability

Available in Mac OS X version 10.0 and later.

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 Mac OS X v10.4 and later.

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

context

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.

See also `CGContextClipToRects` (page 27).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

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

context
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.

See also `CGContextClipToRect` (page 26).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextClosePath

Closes and terminates an open path.

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

Parameters

context
A graphics context.

Discussion
If a path is open, this function closes and terminate the path. Quartz closes a path by drawing a straight line that connects the current point to the starting point. If the current point and the starting point are the same, you must still call this function to close the path. After Quartz terminates the path, the current point is no longer defined. If there is no open path, this function does nothing.

When you fill or clip an open path, Quartz implicitly closes the subpath for you.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextAddPath` (page 19)

Related Sample Code
CALayerEssentials

CarbonSketch

HID Explorer

Declared In
`CGContext.h`

CGContextConcatCTM

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

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

Parameters

`context`
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: `CTMnew = transform * CTMcontext.`

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

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 Mac OS X v10.4 and later.

See Also

`CGContextConvertPointToUserSpace` (page 29)

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 Mac OS X v10.4 and later.

See Also

`CGContextConvertPointToDeviceSpace` (page 29)

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

context

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 Mac OS X v10.4 and later.

See Also

`CGContextConvertRectToUserSpace` (page 30)

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

context

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 Mac OS X v10.4 and later.

See Also

`CGContextConvertRectToDeviceSpace` (page 30)

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 Mac OS X v10.4 and later.

See Also

`CGContextConvertSizeToUserSpace` (page 31)

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

context

A graphics context.

size

The size, in device space coordinates, to transform.

Return Value

The size in user space coordinates.

Availability

Available in Mac OS X v10.4 and later.

See Also

CGContextConvertSizeToDeviceSpace (page 31)

Declared In

CGContext.h

CGContextDrawImage

Draws an image into a graphics context.

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

Parameters

context

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 Mac OS X version 10.0 and later.

Related Sample Code

CarbonCocoa_PictureCursor

WhackedTV

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 or kCGGradientDrawsAfterEndLocation) 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 Mac OS X v10.5 and later.

Declared In

CGContext.h

CGContextDrawPath

Draws the current path using the provided drawing mode.

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

Parameters

context

A graphics context that contains a path to paint.

mode

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

Discussion

This function draws the current path using the specified drawing mode. If the current path contains several disjoint portions (or subpaths), Quartz fills each one independently. Any subpath that you did not explicitly close by calling CGContextClosePath (page 27) is closed implicitly by the fill routines.

Availability

Available in Mac OS X version 10.0 and later.

See Also

CGContextFillPath (page 40)

CGContextEOFillPath (page 39)

CGContextStrokePath (page 80)

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextDrawPDFDocument

Draws a page of a PDF document into a graphics context.

```
void CGContextDrawPDFDocument (
   CGContextRef c,
   CGRect rect,
   CGPDFDocumentRef document,
   int page
);
```

Parameters

`context`
The graphics context in which to draw the PDF page.

`rect`
A `CGRect` value that specifies the dimensions and location of the area in which to draw the PDF page, in units of the user space. When drawn, Quartz scales the media box of the page to fit the rectangle you specify.

`document`
The PDF document to draw.

`page`
A value that specifies the PDF page number to draw. If the specified page does not exist, the function does nothing.

Special Considerations

For applications running in Mac OS X version 10.3 and later, it is recommended that you replace this function with `CGContextDrawPDFPage` (page 34). If you do so, and want to specify the drawing rectangle, you should use `CGPDFPageGetDrawingTransform` to get an appropriate transform, concatenate it with the current transformation matrix, clip to the rectangle, and then call `CGContextDrawPDFPage` (page 34).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextDrawPDFPage

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

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

Parameters

`context`

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 Mac OS X version 10.3 and later.

Related Sample Code

CarbonSketch

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 or kCGGradientDrawsAfterEndLocation) 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 Mac OS X v10.5 and later.

Declared In

CGContext.h

CGContextDrawShading

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

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

Parameters

context

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 Mac OS X v10.2 and later.

See Also

CGContextDrawLinearGradient (page 32)

CGContextDrawRadialGradient (page 35)

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 tile size. 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 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 Mac OS X v10.5 and later.

Declared In

`CGContext.h`

CGContextEndPage

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

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

Parameters

context

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.

For more information, see `CGContextBeginPage` (page 22).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextEndTransparencyLayer

Ends a transparency layer.

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

Parameters

context

A graphics context.

Discussion

See the discussion for CGContextBeginTransparencyLayer (page 23).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

CGContext.h

CGContextEOClip

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

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

Parameters

context

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.

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.

See also CGContextClip (page 25).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

CGContext.h

CGContextEOFillPath

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

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

Parameters

context

A graphics context that contains a path to fill.

Discussion

If the current path contains several disjoint portions (or subpaths), Quartz fills each one independently. Any subpath that you did not explicitly close by calling CGContextClosePath (page 27) is closed implicitly by the fill routines.

Availability

Available in Mac OS X version 10.0 and later.

See Also

CGContextFillPath (page 40)

CGContextStrokePath (page 80)

CGContextDrawPath (page 33)

Related Sample Code

CALayerEssentials

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 Mac OS X v10.4 and later.

Related Sample Code

HID Calibrator

HID Config Save

HID Explorer

Declared In

`CGContext.h`

CGContextFillPath

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

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

Parameters

`context`

A graphics context that contains a path to fill.

Discussion

If the current path contains several disjoint portions (or subpaths), Quartz fills each one independently. Any subpath that you did not explicitly close by calling `CGContextClosePath` (page 27) is closed implicitly by the fill routines.

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextEOFillPath` (page 39)

`CGContextStrokePath` (page 80)

`CGContextDrawPath` (page 33)

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

`context`

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 Mac OS X version 10.0 and later.

See Also
`CGContextFillRects` (page 41)

Related Sample Code
CALayerEssentials

CarbonSketch

HID Calibrator

HID Explorer

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

`context`
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 Mac OS X version 10.0 and later.

See Also
`CGContextFillRect` (page 40)

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

context

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 Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextGetClipBoundingBox

Returns the bounding box of a clipping path.

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

Parameters

context

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 Mac OS X version 10.3 and later.

Related Sample Code

CALayerEssentials

Declared In

`CGContext.h`

CGContextGetCTM

Returns the current transformation matrix.

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

Parameters

context

A graphics context.

Return Value

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

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextGetInterpolationQuality

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

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

Parameters

context

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 Mac OS X version 10.1 and later.

See Also

`CGContextSetInterpolationQuality` (page 62)

Declared In

`CGContext.h`

CGContextGetPathBoundingBox

Returns the smallest rectangle that contains the current path.

```
CGRect CGContextGetPathBoundingBox (
   CGContextRef c
);
```

Parameters

context

The graphics context, containing a path, to examine.

Return Value

A CGRect 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 CGRectNull.

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 Mac OS X version 10.0 and later.

Declared In

CGContext.h

CGContextGetPathCurrentPoint

Returns the current point in a non-empty path.

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

Parameters

context

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 Mac OS X version 10.0 and later.

Declared In

CGContext.h

CGContextGetTextMatrix

Returns the current text matrix.

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

Parameters

context

The graphics context for which to obtain the text matrix.

Return Value

The current text matrix.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextGetTextPosition

Returns the location at which text is drawn.

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

Parameters

context

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 Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextGetTypeID

Returns the type identifier for Quartz graphics contexts.

```
CFTypeID CGContextGetTypeID (
   void
);
```

Return Value

The identifier for the opaque type `CGContextRef` (page 83).

Availability

Available in Mac OS X version 10.2 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 Mac OS X v10.4 and later.

Declared In

`CGContext.h`

CGContextIsPathEmpty

Indicates whether the current path contains any subpaths.

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

Parameters

context

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 Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextMoveToPoint

Begins a new path at the point you specify.

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

Parameters

context

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 current point. It defines the starting point of the next line segment.

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CALayerEssentials

CarbonSketch

HID Calibrator

HID Explorer

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 27), for paths drawn directly to the current context, or `CGPathCloseSubpath`, for paths first created as `CGPath` objects and then drawn to the current context.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGContext.h`

CGContextRelease

Decrements the retain count of a graphics context.

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

Parameters

context

The graphics context to release.

Discussion

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

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

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. 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 25).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGContext.h`

CGContextRestoreGState

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

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

Parameters

context

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 Mac OS X version 10.0 and later.

See Also

`CGContextSaveGState` (page 50)

Related Sample Code

CarbonSketch

HID Calibrator

Declared In

`CGContext.h`

CGContextRetain

Increments the retain count of a graphics context.

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

Parameters

context

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 that it does not cause an error if the `context` parameter is `NULL`.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextRotateCTM

Rotates the user coordinate system in a context.

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

Parameters

context

A graphics context.

angle

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

Availability

Available in Mac OS X version 10.0 and later.

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

context

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 the function `CGContextRestoreGState` (page 49).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch
HID Calibrator

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

context

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 Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

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

context
: 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 "`Text Encodings`" (page 92).

Discussion
For information about when to use this function, see `CGContextShowText` (page 77) and `CGContextShowTextAtPoint` (page 78).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
HID Calibrator

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`.

Availability

Available in Mac OS X v10.4 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

context

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, you should use the function `CGContextClearRect` (page 24).

Availability

Available in Mac OS X version 10.0 and later.

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 "`Blend Modes`" (page 83) for a list of the constants you can supply.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGContext.h`

CGContextSetCharacterSpacing

Sets the current character spacing.

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

Parameters

`context`

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 73).

Availability

Available in Mac OS X version 10.0 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

context

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.

See also `CGContextSetCMYKStrokeColor` (page 56).

Availability

Available in Mac OS X version 10.0 and later.

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

context

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.

See also `CGContextSetCMYKFillColor` (page 54).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetFillColor

Sets the current fill color.

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

Parameters

context

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 58). Note that the preferred API to use is now `CGContextSetFillColorWithColor` (page 58).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextSetFillColorSpace

Sets the fill color space in a graphics context.

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

Parameters

context

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 57). Note that the preferred API to use is now `CGContextSetFillColorWithColor` (page 58).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

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

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

`color`
The new fill color.

Discussion
See also `CGContextSetFillColor` (page 57).

Availability
Available in Mac OS X version 10.3 and later.

Related Sample Code
CALayerEssentials

Declared In
`CGContext.h`

CGContextSetFillPattern

Sets the fill pattern in the specified graphics context.

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

Parameters

`context`
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, then call the function `CGContextSetFillColorWithColor` (page 58).

Availability

Available in Mac OS X version 10.1 and later.

Declared In

`CGContext.h`

CGContextSetFlatness

Sets the accuracy of curved paths in a graphics context.

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

Parameters

context

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 Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetFont

Sets the platform font in a graphics context.

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

Parameters

context

The graphics context for which to set the font.

font

A Quartz font.

Discussion

For information about when to use this function, see `CGFontCreateWithPlatformFont`.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetFontSize

Sets the current font size.

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

Parameters

context

A graphics context.

size

A font size, expressed in text space units.

Availability

Available in Mac OS X version 10.0 and later.

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

context

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.

See also `CGContextSetGrayStrokeColor` (page 61).

Availability

Available in Mac OS X version 10.0 and later.

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

context

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.

See also `CGContextSetGrayFillColor` (page 60).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetInterpolationQuality

Sets the level of interpolation quality for a graphics context.

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

Parameters

context

The graphics context to modify.

quality

A `CGInterpolationQuality` constant that specifies the required level of interpolation quality. For possible values, see "`Interpolation Qualities`" (page 88).

Discussion

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

Availability

Available in Mac OS X version 10.1 and later.

See Also

`CGContextGetInterpolationQuality` (page 43)

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

context

The graphics context to modify.

cap

A line cap style constant—kCGLineCapButt (page 89) (the default), kCGLineCapRound (page 89), or kCGLineCapSquare (page 89). See "Line Cap Styles" (page 89).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

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

context

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 Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

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

`context`

The graphics context to modify.

`join`

A line join value—`kCGLineJoinMiter` (page 90) (the default), `kCGLineJoinRound` (page 90), or `kCGLineJoinBevel` (page 90). See "`Line Joins`" (page 90).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextSetLineWidth

Sets the line width for a graphics context.

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

Parameters

`context`

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 Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

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

`context`
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 64)), 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 Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextSetPatternPhase

Sets the pattern phase of a context.

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

Parameters

`context`
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 Mac OS X version 10.2 and later.

Declared In

`CGContext.h`

CGContextSetRenderingIntent

Sets the rendering intent in the current graphics state.

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

Parameters

`context`

The graphics context to modify.

`intent`

A rendering intent constant—`kCGRenderingIntentDefault`, `kCGRenderingIntentAbsoluteColorimetric`, `kCGRenderingIntentRelativeColorimetric`, `kCGRenderingIntentPerceptual`, or `kCGRenderingIntentSaturation`. 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 Mac OS X version 10.0 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

context

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.

See also `CGContextSetRGBStrokeColor` (page 67).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CALayerEssentials

CarbonSketch

HID Calibrator

HID Config Save

HID Explorer

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

context

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.

See also CGContextSetRGBFillColor (page 66).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

HID Calibrator

HID Config Save

HID Explorer

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., 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 69) to set the shadow color to a fully transparent color (or pass NULL as the color).

Availability

Available in Mac OS X version 10.3 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.

Discussion

See also `CGContextSetShadow` (page 68).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGContext.h`

CGContextSetShouldAntialias

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

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

Parameters

context

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 Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetShouldSmoothFonts

Enables or disables font smoothing in a graphics context.

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

Parameters

context

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.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGContext.h`

CGContextSetStrokeColor

Sets the current stroke color.

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

Parameters

context

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 72). Note that the preferred API is now `CGContextSetStrokeColorWithColor` (page 72).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextSetStrokeColorSpace

Sets the stroke color space in a graphics context.

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

Parameters

context

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 71). Note that the preferred API is now `CGContextSetStrokeColorWithColor` (page 72).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextSetStrokeColorWithColor

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

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

Parameters

context
The graphics context to modify.

color
The new stroke color.

Discussion

See also `CGContextSetStrokeColor` (page 71).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGContext.h`

CGContextSetStrokePattern

Sets the stroke pattern in the specified graphics context.

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

Parameters

context
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, then call the function `CGContextSetStrokeColorWithColor` (page 72).

Availability

Available in Mac OS X version 10.1 and later.

Declared In

`CGContext.h`

CGContextSetTextDrawingMode

Sets the current text drawing mode.

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

Parameters

context

A graphics context.

mode

A text drawing mode (such as `kCGTextFill` (page 91) or `kCGTextStroke` (page 91)) that specifies how Quartz renders individual glyphs in a graphics context. See "`Text Drawing Modes`" (page 90) for a complete list.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetTextMatrix

Sets the current text matrix.

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

Parameters

context

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 Mac OS X version 10.0 and later.

Related Sample Code

HID Calibrator

Declared In

`CGContext.h`

CGContextSetTextPosition

Sets the location at which text is drawn.

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

Parameters

context

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.

Availability

Available in Mac OS X version 10.0 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

context

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.

See also CGContextShowGlyphsAtPoint (page 75), CGContextShowText (page 77), CGContextShowTextAtPoint (page 78), and CGContextShowGlyphsWithAdvances (page 76).

Availability

Available in Mac OS X version 10.0 and later.

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

context

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 text space.

See also CGContextShowText (page 77), CGContextShowGlyphs (page 75), CGContextShowGlyphs (page 75), and CGContextShowGlyphsWithAdvances (page 76).

Availability

Available in Mac OS X version 10.0 and later.

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

context

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 Mac OS X v10.5 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

context

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.

See also `CGContextShowText` (page 77), `CGContextShowGlyphs` (page 75), and `CGContextShowGlyphs` (page 75), and `CGContextShowGlyphsAtPoint` (page 75).

Availability

Available in Mac OS X version 10.3 and later.

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

context

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 52). Don't use `CGContextShowTextAtPoint` in conjunction with `CGContextSetFont` (page 59).

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextShowTextAtPoint` (page 78)

`CGContextShowGlyphs` (page 75)

`CGContextShowGlyphsAtPoint` (page 75)

`CGContextShowGlyphsWithAdvances` (page 76)

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

context

A graphics context .

x

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

y

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

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 52). Don't use `CGContextShowTextAtPoint` in conjunction with `CGContextSetFont` (page 59).

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextShowText` (page 77)

`CGContextShowGlyphs` (page 75)

`CGContextShowGlyphsAtPoint` (page 75)

`CGContextShowGlyphsWithAdvances` (page 76)

Related Sample Code

HID Calibrator

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 Mac OS X v10.4 and later.

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 Mac OS X v10.4 and later.

Declared In

`CGContext.h`

CGContextStrokePath

Paints a line along the current path.

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

Parameters

context

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 Mac OS X version 10.0 and later.

See Also

`CGContextDrawPath` (page 33)

`CGContextFillPath` (page 40)

`CGContextEOFillPath` (page 39)

Related Sample Code

CarbonSketch

HID Calibrator

HID Explorer

Declared In

`CGContext.h`

CGContextStrokeRect

Paints a rectangular path.

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

Parameters

context

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 Mac OS X version 10.0 and later.

See Also

`CGContextStrokeRectWithWidth` (page 81)

Related Sample Code

CarbonSketch

HID Calibrator

HID Config Save

Declared In

`CGContext.h`

CGContextStrokeRectWithWidth

Paints a rectangular path, using the specified line width.

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

Parameters

context

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 Mac OS X version 10.0 and later.

See Also

`CGContextStrokeRect` (page 80)

Declared In

`CGContext.h`

CGContextSynchronize

Marks a window context for update.

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

Parameters

context

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 Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

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

context

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 Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

Data Types

CGContextRef

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

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

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGContext.h`

Constants

Blend Modes

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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 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 Mac OS X v10.4 and later.

Declared in `CGContext.h`.

`kCGBlendModeClear`

`R = 0`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeCopy`

`R = S`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeSourceIn`

`R = S*Da`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeSourceOut`

`R = S*(1 - Da)`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeSourceAtop`

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

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeDestinationOver`

`R = S*(1 - Da) + D`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeDestinationIn`

`R = D*Sa`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeDestinationOut`

`R = D*(1 - Sa)`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeDestinationAtop`

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

Available in Mac OS X v10.5 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 Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModePlusDarker`
: `R = MAX(0, (1 - D) + (1 - S))`

  Available in Mac OS X v10.5 and later.

  Declared in `CGContext.h`.

`kCGBlendModePlusLighter`
: `R = MIN(1, S + D)`

  Available in Mac OS X v10.5 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`

Interpolation Qualities

Levels of interpolation quality for rendering an image.

```
enum CGInterpolationQuality {
    kCGInterpolationDefault,
    kCGInterpolationNone,
    kCGInterpolationLow,
    kCGInterpolationHigh
};
typedef enum CGInterpolationQuality CGInterpolationQuality;
```

Constants

`kCGInterpolationDefault`
: The default level of quality.

  Available in Mac OS X v10.1 and later.

  Declared in `CGContext.h`.

`kCGInterpolationNone`

No interpolation.

Available in Mac OS X v10.1 and later.

Declared in `CGContext.h`.

`kCGInterpolationLow`

A low level of interpolation quality. This setting may speed up image rendering.

Available in Mac OS X v10.1 and later.

Declared in `CGContext.h`.

`kCGInterpolationHigh`

A high level of interpolation quality. This setting may slow down image rendering.

Available in Mac OS X v10.1 and later.

Declared in `CGContext.h`.

Discussion

You use the function `CGContextSetInterpolationQuality` (page 62) to set the interpolation quality in a graphics context.

Declared In

`CGContext.h`

Line Cap Styles

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 Mac OS X v10.0 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 Mac OS X v10.0 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 Mac OS X v10.0 and later.

Declared in `CGContext.h`.

Discussion

A line cap specifies the method used by CGContextStrokePath (page 80) to draw the endpoint of the line. To change the line cap style in a graphics context, you use the function CGContextSetLineCap (page 62).

Declared In

CGContext.h

Line Joins

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 65)

Available in Mac OS X v10.0 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 Mac OS X v10.0 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 Mac OS X v10.0 and later.

Declared in CGContext.h.

Discussion

A line join specifies how CGContextStrokePath (page 80) draws the junction between connected line segments. To set the line join style in a graphics context, you use the function CGContextSetLineJoin (page 64).

Declared In

CGContext.h

Text Drawing Modes

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 Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGTextStroke`

Perform a stroke operation on the text.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGTextFillStroke`

Perform fill, then stroke operations on the text.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGTextInvisible`

Do not draw the text, but do update the text position.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGTextFillClip`

Perform a fill operation, then intersect the text with the current clipping path.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGTextStrokeClip`

Perform a stroke operation, then intersect the text with the current clipping path.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGTextFillStrokeClip`

Perform fill then stroke operations, then intersect the text with the current clipping path.

Available in Mac OS X v10.0 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 Mac OS X v10.0 and later.

Declared in `CGContext.h`.

Discussion

You provide a text drawing mode constant to the function CGContextSetTextDrawingMode (page 73) 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

Text Encodings

Text encodings for fonts.

```
enum CGTextEncoding {
    kCGEncodingFontSpecific,
    kCGEncodingMacRoman
};
typedef enum CGTextEncoding CGTextEncoding;
```

Constants

`kCGEncodingFontSpecific`

The built-in encoding of the font.

Available in Mac OS X v10.0 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 Mac OS X v10.0 and later.

Declared in `CGContext.h`.

Discussion

For more information on setting the font in a graphics context, see CGContextSelectFont (page 52).

Declared In

CGContext.h

Document Revision History

This table describes the changes to *CGContext Reference*.

Date	Notes
2008-04-08	Made a minor typographical change.
2007-07-17	Updated for Mac OS X v10.5.
	All instances of the `float` data type were changed to the `CGFloat` data type.
	Added the Porter-Duff blend modes to "Blend Modes" (page 83).
	Added `CGContextDrawLinearGradient` (page 32), `CGContextDrawRadialGradient` (page 35), `CGContextDrawTiledImage` (page 36), `CGContextShowGlyphsAtPositions` (page 76), and `CGContextBeginTransparencyLayerWithRect` (page 24).
	Modified the description of the auxiliaryInfo parameter in `CGContextBeginTransparencyLayer` (page 23).
	Added a performance tip to `CGContextBeginTransparencyLayer` (page 23).
	Clarified the definition of "inside" for `CGContextPathContainsPoint` (page 47).
2006-01-10	Added information to the discussion for blend mode constants.
	Improved wording concerning text clipping in "Text Drawing Modes" (page 90).
2005-11-09	Revised information about text space and user space.
2005-04-29	Updated for Mac OS X v10.4.

Date	Notes
	Added the functions CGContextSetBlendMode (page 53), CGContextAddEllipseInRect (page 17), CGContextReplacePathWithStrokedPath (page 48), CGContextPathContainsPoint (page 47), CGContextFillEllipseInRect (page 39), CGContextStrokeEllipseInRect (page 79), CGContextStrokeLineSegments (page 79), CGContextClipToMask (page 25), CGContextSetAllowsAntialiasing (page 52), CGContextGetUserSpaceToDeviceSpaceTransform (page 46), CGContextConvertPointToDeviceSpace (page 29), CGContextConvertPointToUserSpace (page 29), CGContextConvertSizeToDeviceSpace (page 31), CGContextConvertSizeToUserSpace (page 31), CGContextConvertRectToDeviceSpace (page 30), and CGContextConvertRectToUserSpace (page 30).
	Added constants for "Blend Modes" (page 83).
	Changed the description of what to pass for the components parameter for the functions CGContextSetFillPattern (page 58) and CGContextSetStrokePattern (page 72).
2004-08-31	Updated graphics context information and added cross references to conceptual documentation.
2004-02-26	First version of this document. An earlier version of this information appeared in *Quartz 2D Reference*.

Index