
CALayer Class Reference

Graphics & Animation: Animation



2010-06-11



Apple Inc.
© 2010 Apple Inc.
All rights reserved.

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

The Apple logo is a trademark of Apple Inc.

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

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

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

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

Apple, the Apple logo, iPhone, Mac, Mac OS, Objective-C, Quartz, and Spaces are trademarks of Apple Inc., registered in the United States and other countries.

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

OpenGL is a registered trademark of Silicon Graphics, Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY,

MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

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

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

Contents

CALayer Class Reference 7

Overview	7
Tasks	7
Creating a Layer	7
Accessing the Presentation Layer	8
Modifying the Layer Geometry	8
Providing Layer Content	8
Style Attributes	9
Managing the Layer Hierarchy	10
Updating Layer Display	10
Layer Animations	11
Managing Layer Resizing and Layout	11
Actions	11
Mapping Between Coordinate and Time Spaces	12
Hit Testing	12
Rendering	12
Scrolling	12
Modifying the Delegate	13
Key-Value Coding Extensions	13
Properties	13
actions	13
anchorPoint	14
anchorPointZ	14
backgroundColor	14
backgroundFilters	15
borderColor	15
borderWidth	15
bounds	16
compositingFilter	16
contents	17
contentsCenter	17
contentsGravity	18
contentsRect	18
contentsScale	19
cornerRadius	19
delegate	19
doubleSided	20
edgeAntialiasingMask	20
filters	20
frame	21
geometryFlipped	21

hidden	22
magnificationFilter	22
mask	22
masksToBounds	23
minificationFilter	23
minificationFilterBias	23
name	24
needsDisplayOnBoundsChange	24
opacity	24
opaque	24
position	25
rasterizationScale	25
shadowColor	26
shadowOffset	26
shadowOpacity	26
shadowPath	27
shadowRadius	27
shouldRasterize	27
style	28
sublayers	28
sublayerTransform	29
superlayer	29
transform	29
visibleRect	30
zPosition	30
Class Methods	30
defaultActionForKey:	30
defaultValueForKey:	31
layer	31
needsDisplayForKey:	32
Instance Methods	32
actionForKey:	32
addAnimation:forKey:	33
addSublayer:	34
affineTransform	34
animationForKey:	34
animationKeys	35
containsPoint:	35
contentsAreFlipped	35
convertPoint:fromLayer:	36
convertPoint:toLayer:	36
convertRect:fromLayer:	37
convertRect:toLayer:	37
convertTime:fromLayer:	38
convertTime:toLayer:	38
display	38

- displayIfNeeded 39
- drawInContext: 39
- hitTest: 40
- init 40
- initWithLayer: 40
- insertSublayer:above: 41
- insertSublayer:atIndex: 41
- insertSublayer:below: 42
- layoutIfNeeded 42
- layoutSublayers 43
- modelLayer 43
- needsDisplay 43
- needsLayout 44
- preferredFrameSize 44
- presentationLayer 44
- removeAllAnimations 45
- removeAnimationForKey: 45
- removeFromSuperlayer 45
- renderInContext: 45
- replaceSublayer:with: 46
- scrollPoint: 46
- scrollRectToVisible: 47
- setAffineTransform: 47
- setNeedsDisplay 47
- setNeedsDisplayInRect: 48
- setNeedsLayout 48
- shouldArchiveValueForKey: 48
- Delegate Methods 49
 - actionForLayer:forKey: 49
 - displayLayer: 49
 - drawLayer:inContext: 50
- Constants 50
 - Action Identifiers 50
 - Edge Antialiasing Mask 51
 - Contents Gravity Values 52
 - Identity Transform 53
 - Scaling Filters 54
 - Transform 54

Document Revision History 57

CALayer Class Reference

Inherits from	NSObject
Conforms to	NSCoding CAMediaTiming NSObject (NSObject)
Framework	/System/Library/Frameworks/QuartzCore.framework
Availability	Available in iOS 2.0 and later.
Declared in	CALayer.h CASScrollLayer.h CATransform3D.h
Companion guides	Core Animation Programming Guide Core Animation Cookbook

Overview

The `CALayer` class is the model class for layer-tree objects. It encapsulates the position, size, and transform of a layer, which defines its coordinate system. It also encapsulates the duration and pacing of a layer and its animations by adopting the `CAMediaTiming` protocol, which defines a layer's time space.

Tasks

Creating a Layer

- + `layer` (page 31)
Creates and returns an instance of `CALayer`.
- `init` (page 40)
Returns an initialized `CALayer` object.
- `initWithLayer:` (page 40)
Override to copy or initialize custom fields of the specified layer.

Accessing the Presentation Layer

- [presentationLayer](#) (page 44)
Returns a copy of the layer containing all properties as they were at the start of the current transaction, with any active animations applied.
- [modelLayer](#) (page 43)
Returns the model layer of the receiver, if it represents a current presentation layer.

Modifying the Layer Geometry

- [frame](#) (page 21) *property*
Specifies receiver's frame rectangle in the super-layer's coordinate space.
- [bounds](#) (page 16) *property*
Specifies the bounds rectangle of the receiver. Animatable.
- [position](#) (page 25) *property*
Specifies the receiver's position in the superlayer's coordinate system. Animatable.
- [zPosition](#) (page 30) *property*
Specifies the receiver's position on the z axis. Animatable.
- [anchorPointZ](#) (page 14) *property*
The Z component of the layer's anchor point. Animatable.
- [anchorPoint](#) (page 14) *property*
Defines the anchor point of the layer's bounds rectangle. Animatable.
- [contentsScale](#) (page 19) *property*
The scale factor applied to the layer.
- [affineTransform](#) (page 34)
Convenience method for getting the [transform](#) (page 29) property as an affine transform.
- [setAffineTransform:](#) (page 47)
Convenience method for setting the [transform](#) (page 29) property as an affine transform.
- [transform](#) (page 29) *property*
Specifies the transform applied to the receiver, relative to the center of its bounds. Animatable.
- [sublayerTransform](#) (page 29) *property*
Specifies a transform applied to each sublayer when rendering. Animatable.

Providing Layer Content

- [contents](#) (page 17) *property*
An object that provides the contents of the layer. Animatable.
- [contentsRect](#) (page 18) *property*
A rectangle, in the unit coordinate space, defining the subrectangle of [contents](#) (page 17) that the receiver should draw. Animatable.
- [contentsCenter](#) (page 17) *property*
Specifies the area of the content image that should be scaled. Animatable.
- [display](#) (page 38)
Reload the content of this layer.

- [displayLayer:](#) (page 49) *delegate method*
Allows the delegate to override the `display` (page 38) implementation.
- [drawInContext:](#) (page 39)
Draws the receiver's content in the specified graphics context.
- [drawLayer:inContext:](#) (page 50) *delegate method*
Allows the delegate to override the layer's `drawInContext:` implementation.
- [opaque](#) (page 24) *property*
This property affects how the layer creates the content directly.
- [edgeAntialiasingMask](#) (page 20) *property*
A bitmask defining how the edges of the receiver are rasterized.
- [contentsAreFlipped](#) (page 35)
Returns whether the layer content is implicitly flipped when rendered.
- [geometryFlipped](#) (page 21) *property*
Determines if the geometry of the layer and its sublayers are flipped vertically.

Style Attributes

- [contentsGravity](#) (page 18) *property*
Determines how the receiver's contents are positioned within its bounds.
- [opacity](#) (page 24) *property*
Determines the opacity of the receiver. Animatable.
- [hidden](#) (page 22) *property*
Determines whether the receiver is displayed. Animatable.
- [masksToBounds](#) (page 23) *property*
Determines if the sublayers are clipped to the receiver's bounds. Animatable.
- [doubleSided](#) (page 20) *property*
Determines whether the receiver is displayed when facing away from the viewer. Animatable.
- [mask](#) (page 22) *property*
An optional layer whose alpha channel is used as a mask to select between the layer's background and the result of compositing the layer's contents with its filtered background.
- [cornerRadius](#) (page 19) *property*
Specifies a radius used to draw the rounded corners of the receiver's background. Animatable.
- [borderWidth](#) (page 15) *property*
Specifies the width of the receiver's border. Animatable.
- [borderColor](#) (page 15) *property*
The color of the receiver's border. Animatable.
- [backgroundColor](#) (page 14) *property*
Specifies the background color of the receiver. Animatable.
- [backgroundFilters](#) (page 15) *property*
An optional array of CoreImage filters that are applied to the receiver's background. Animatable.
- [shadowOpacity](#) (page 26) *property*
Specifies the opacity of the receiver's shadow. Animatable.
- [shadowRadius](#) (page 27) *property*
Specifies the blur radius used to render the receiver's shadow. Animatable.

[shadowOffset](#) (page 26) *property*

Specifies the offset of the receiver's shadow. Animatable.

[shadowColor](#) (page 26) *property*

Specifies the color of the receiver's shadow. Animatable.

[shadowPath](#) (page 27) *property*

Defines the shape of the shadow.

[filters](#) (page 20) *property*

An array of CoreImage filters that are applied to the contents of the receiver and its sublayers. Animatable.

[compositingFilter](#) (page 16) *property*

A CoreImage filter used to composite the receiver's contents with the background. Animatable.

[style](#) (page 28) *property*

An optional dictionary referenced to find property values that aren't explicitly defined by the receiver.

[minificationFilter](#) (page 23) *property*

The filter used when reducing the size of the content.

[minificationFilterBias](#) (page 23) *property*

The bias factor used by the minification filter to determine the levels of detail.

[magnificationFilter](#) (page 22) *property*

The filter used when increasing the size of the content.

Managing the Layer Hierarchy

[sublayers](#) (page 28) *property*

An array containing the receiver's sublayers.

[superlayer](#) (page 29) *property*

Specifies receiver's superlayer. (read-only)

- [addSublayer:](#) (page 34)

Appends the layer to the receiver's [sublayers](#) (page 28) array.

- [removeFromSuperlayer](#) (page 45)

Removes the layer from the [sublayers](#) (page 28) array or [mask](#) (page 22) property of the receiver's [superlayer](#) (page 29).

- [insertSublayer:atIndex:](#) (page 41)

Inserts the layer as a sublayer of the receiver at the specified index.

- [insertSublayer:below:](#) (page 42)

Inserts the layer into the receiver's sublayers array, below the specified sublayer.

- [insertSublayer:above:](#) (page 41)

Inserts the layer into the receiver's sublayers array, above the specified sublayer.

- [replaceSublayer:with:](#) (page 46)

Replaces the layer in the receiver's sublayers array with the specified new layer.

Updating Layer Display

- [setNeedsDisplay](#) (page 47)

Marks the receiver as needing display before the content is next committed.

- `needsDisplayOnBoundsChange` (page 24) *property*
Returns whether the receiver must be redisplayed when the bounds rectangle is updated.
- `displayIfNeeded` (page 39)
Displays the layer if it has been marked as needing display.
- `needsDisplay` (page 43)
Returns whether the layer has been marked as requiring display.
- + `needsDisplayForKey:` (page 32)
Returns whether changes to the specified key requires the layer to be redisplayed.
- `setNeedsDisplayInRect:` (page 48)
Marks the region of the receiver within the specified rectangle as needing display.

Layer Animations

- `addAnimation:forKey:` (page 33)
Add an animation object to the receiver's render tree for the specified key.
- `animationForKey:` (page 34)
Returns the animation added to the receiver with the specified identifier.
- `removeAllAnimations` (page 45)
Remove all animations attached to the receiver.
- `removeAnimationForKey:` (page 45)
Remove the animation attached to the receiver with the specified key.
- `animationKeys` (page 35)
Returns an array containing the keys of all animations currently attached to the receiver.

Managing Layer Resizing and Layout

- `setNeedsLayout` (page 48)
Called when the preferred size of the receiver may have changed.
- `name` (page 24) *property*
The name of the receiver.
- `preferredFrameSize` (page 44)
Returns the preferred frame size of the layer in the coordinate space of the superlayer.
- `layoutIfNeeded` (page 42)
Recalculate the receiver's layout, if required.
- `layoutSublayers` (page 43)
Called when the layer requires layout.
- `needsLayout` (page 44)
Returns whether the layer has been marked as requiring layout.

Actions

- `actions` (page 13) *property*
A dictionary mapping keys to objects that implement the `CAAction` protocol.

- + [defaultActionForKey:](#) (page 30)
Returns an object that implements the default action for the specified identifier.
- [actionForKey:](#) (page 32)
Returns an object that implements the action for the specified identifier.
- [actionForLayer:forKey:](#) (page 49) *delegate method*
Allows the delegate to customize the action for a layer.

Mapping Between Coordinate and Time Spaces

- [convertPoint:fromLayer:](#) (page 36)
Converts the point from the specified layer's coordinate system to the receiver's coordinate system.
- [convertPoint:toLayer:](#) (page 36)
Converts the point from the receiver's coordinate system to the specified layer's coordinate system.
- [convertRect:fromLayer:](#) (page 37)
Converts the rectangle from the specified layer's coordinate system to the receiver's coordinate system.
- [convertRect:toLayer:](#) (page 37)
Converts the rectangle from the receiver's coordinate system to the specified layer's coordinate system.
- [convertTime:fromLayer:](#) (page 38)
Converts the time interval from the specified layer's time space to the receiver's time space.
- [convertTime:toLayer:](#) (page 38)
Converts the time interval from the receiver's time space to the specified layer's time space

Hit Testing

- [hitTest:](#) (page 40)
Returns the farthest descendant of the receiver in the layer hierarchy (including itself) that contains a specified point.
- [containsPoint:](#) (page 35)
Returns whether the receiver contains a specified point.

Rendering

- [renderInContext:](#) (page 45)
Renders the receiver and its sublayers into the specified context.
- [shouldRasterize](#) (page 27) *property*
A Boolean that indicates whether the layer is rendered as a bitmap before compositing. Animatable
- [rasterizationScale](#) (page 25) *property*
The scale at which to rasterize content, relative to the coordinate space of the layer. Animatable

Scrolling

- [visibleRect](#) (page 30) *property*
Returns the visible region of the receiver, in its own coordinate space. (read-only)

- [scrollPoint:](#) (page 46)
Scrolls the receiver's closest ancestor `CAScrollLayer` so that the specified point lies at the origin of the layer.
- [scrollRectToVisible:](#) (page 47)
Scrolls the receiver's closest ancestor `CAScrollLayer` the minimum distance needed so that the specified rectangle becomes visible.

Modifying the Delegate

- [delegate](#) (page 19) *property*
Specifies the receiver's delegate object.

Key-Value Coding Extensions

- [shouldArchiveValueForKey:](#) (page 48)
Specifies whether the value of the property for a given key is archived.
- + [defaultValueForKey:](#) (page 31)
Specifies the default value of the property with the specified key.

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

actions

A dictionary mapping keys to objects that implement the `CAAction` protocol.

```
@property(copy) NSDictionary *actions
```

Discussion

The default value is `nil`. See [actionForKey:](#) (page 32) for a description of the action search pattern.

Availability

Available in iOS 2.0 and later.

See Also

- [actionForKey:](#) (page 32)
- [actionForLayer:forKey:](#) (page 49)
- + [defaultActionForKey:](#) (page 30)
- [@property style](#) (page 28)

Declared In

`CALayer.h`

anchorPoint

Defines the anchor point of the layer's bounds rectangle. Animatable.

```
@property CGPoint anchorPoint
```

Discussion

Described in the unit coordinate space. The value of this property is specified in points. Defaults to (0.5, 0.5), the center of the bounds rectangle.

See “Layer Geometry and Transforms” in *Core Animation Programming Guide* for more information on the relationship between the [bounds](#) (page 16), [anchorPoint](#) (page 14) and [position](#) (page 25) properties.

Availability

Available in iOS 2.0 and later.

See Also

[@property position](#) (page 25)

Declared In

CALayer.h

anchorPointZ

The Z component of the layer's anchor point. Animatable.

```
@property CGFloat anchorPointZ
```

Discussion

The `anchorPointZ` value is expressed as a distance along the Z axis. Defaults to 0.

Availability

Available in iOS 3.0 and later.

See Also

[@property anchorPoint](#) (page 14)

Declared In

CALayer.h

backgroundColor

Specifies the background color of the receiver. Animatable.

```
@property UIColorRef backgroundColor
```

Discussion

The default is `nil`.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

backgroundFilters

An optional array of CoreImage filters that are applied to the receiver's background. Animatable.

```
@property(copy) NSArray *backgroundFilters
```

Discussion

Once an array of filters is set properties should be modified by invoking `setValue:forKeyPath:` using the appropriate key path. This requires that you set the name of the background filter to be modified. For example:

```
CIFilter *filter = ...;
CALayer *layer = ...;

filter.name = @"myFilter";
layer.filters = [NSArray arrayWithObject:filter];
[layer setValue:[NSNumber numberWithInt:1]
  forKeyPath:@"filters.myFilter.inputScale"];
```

If the inputs of a background filter are directly modified after the filter is attached to a layer, the behavior is undefined.

Special Considerations

While the `CALayer` class exposes this property, Core Image is not available in iOS. Currently the filters available for this property are undefined.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

borderColor

The color of the receiver's border. Animatable.

```
@property CGColorRef borderColor
```

Discussion

Defaults to opaque black.

Availability

Available in iOS 3.0 and later.

Declared In

`CALayer.h`

borderWidth

Specifies the width of the receiver's border. Animatable.

```
@property CGFloat borderWidth
```

Discussion

The border is drawn inset from the receiver's bounds by `borderWidth`. It is composited above the receiver's [contents](#) (page 17) and [sublayers](#) (page 28) and includes the effects of the [cornerRadius](#) (page 19) property. The default is 0.0.

Availability

Available in iOS 3.0 and later.

Declared In

CALayer.h

bounds

Specifies the bounds rectangle of the receiver. Animatable.

```
@property CGRect bounds
```

Discussion

The default is an empty rectangle. The value of this property is specified in points.

See “Layer Geometry and Transforms” in *Core Animation Programming Guide* for more information on the relationship between the [bounds](#) (page 16), [anchorPoint](#) (page 14) and [position](#) (page 25) properties.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

compositingFilter

A CoreImage filter used to composite the receiver's contents with the background. Animatable.

```
@property(retain) id compositingFilter
```

Discussion

If `nil`, the contents are composited using source-over. The default value is `nil`.

Once a filter is set its properties should be modified by invoking `setValue:forKeyPath:` using the appropriate key path. For example:

```
CIFilter *filter = ...;
CALayer *layer = ...;

layer.compositingFilter = filter;
[layer setValue:[NSNumber numberWithInt:1]
forKeyPath:@"compositingFilter.inputScale"];
```

If the inputs of the filter are modified directly after the filter is attached to a layer, the behavior is undefined.

Special Considerations

While the `CALayer` class exposes this property, Core Image is not available in iOS. Currently the filters available for this property are undefined.

Availability

Available in iOS 2.0 and later.

See Also

[@property backgroundFilters](#) (page 15)

Declared In

`CALayer.h`

contents

An object that provides the contents of the layer. Animatable.

```
@property(retain) id contents
```

Discussion

A layer can set this property to a `CGImageRef` to display the image as its contents. The default value is `nil`.

Availability

Available in iOS 2.0 and later.

See Also

[@property contentsRect](#) (page 18)

Declared In

`CALayer.h`

contentsCenter

Specifies the area of the content image that should be scaled. Animatable.

```
@property CGRect contentsCenter
```

Discussion

The rectangle is interpreted after the effects of the `contentsRect` property have been applied to the image.

Defaults to the unit rectangle (0.0,0.0) (1.0,1.0) resulting in the entire image being scaled. If the rectangle extends outside the unit rectangle the result is undefined.

When an image is resized due to its [contentsGravity](#) (page 18) property, its center part implicitly defines the 3x3 grid that controls how the image is scaled to its drawn size. The center part is stretched in both dimensions; the top and bottom parts are only stretched horizontally; the left and right parts are only stretched vertically; the four corner parts are not stretched at all.

Note: If the width or height of `contentsCenter` is 0, it is implicitly adjusted to the width or height of a single source pixel centered at that position.

Availability

Available in iOS 3.0 and later.

See Also

[@property contentsRect](#) (page 18)

[@property contentsGravity](#) (page 18)

[@property contents](#) (page 17)

Declared In

CALayer.h

contentsGravity

Determines how the receiver's contents are positioned within its bounds.

```
@property(copy) NSString *contentsGravity
```

Discussion

The possible values for `contentsGravity` are shown in “[Contents Gravity Values](#)” (page 52). The default value is `kCAGravityResize` (page 53).

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

contentsRect

A rectangle, in the unit coordinate space, defining the subrectangle of [contents](#) (page 17) that the receiver should draw. Animatable.

```
@property CGRect contentsRect
```

Discussion

Defaults to the unit rectangle (0.0, 0.0, 1.0, 1.0).

If pixels outside the unit rectangles are requested, the edge pixels of the contents image will be extended outwards.

If an empty rectangle is provided, the results are undefined.

Availability

Available in iOS 2.0 and later.

See Also

[@property contents](#) (page 17)

Declared In

CALayer.h

contentsScale

The scale factor applied to the layer.

```
@property CGFloat contentsScale
```

Discussion

This value defines the mapping between the logical coordinate space of the layer (measured in points) and the physical coordinate space (measured in pixels). Higher scale factors indicate that each point in the layer is represented by more than one pixel at render time. For example, if the scale factor is 2.0 and the layer's bounds are 50 x 50 points, the size of the bitmap used to present the layer's content is 100 x 100 pixels.

The `contentsScale` default value is 1.0. In certain restricted cases, the value may set the value to 2.0 on hi-dpi devices.

You can change this value as needed to indicate to Core Animation that the bitmap of the backing layer needs to be bigger or smaller. For example, if you set the contents of the view directly, you can change the value to ensure that layer's bitmap matches the size of the image you are using.

Availability

Available in iOS 4.0 and later.

Declared In

CALayer.h

cornerRadius

Specifies a radius used to draw the rounded corners of the receiver's background. Animatable.

```
@property CGFloat cornerRadius
```

Discussion

If the radius is greater than 0 the background is drawn with rounded corners. The default value is 0.0.

Availability

Available in iOS 3.0 and later.

Declared In

CALayer.h

delegate

Specifies the receiver's delegate object.

```
@property(assign) id delegate
```

Discussion

In iOS, if you want to assign a `UIView` object to this property, you *must* assign the view whose layer this is. Assigning a a superview of the layer's view will cause your application to crash during drawing.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

doubleSided

Determines whether the receiver is displayed when facing away from the viewer. Animatable.

```
@property(getter=isDoubleSided) BOOL doubleSided
```

Discussion

If NO, the layer is hidden when facing away from the viewer. Defaults to YES.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

edgeAntialiasingMask

A bitmask defining how the edges of the receiver are rasterized.

```
@property unsigned int edgeAntialiasingMask
```

Discussion

For each of the four edges (left, right, bottom, top) if the corresponding bit is set the edge will be antialiased.

Typically, this property is used to disable antialiasing for edges that abut edges of other layers, to eliminate the seams that would otherwise occur.

The mask values are defined in [“Edge Antialiasing Mask”](#) (page 51).

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

filters

An array of CoreImage filters that are applied to the contents of the receiver and its sublayers. Animatable.

```
@property(copy) NSArray *filters
```

Discussion

Defaults to nil. Filter properties should be modified by calling `setValue:forKeyPath:` on each layer that the filter is attached to. If the inputs of the filter are modified directly after the filter is attached to a layer, the behavior is undefined.

Special Considerations

While the `CALayer` class exposes this property, Core Image is not available in iOS. Currently the filters available for this property are undefined.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

frame

Specifies receiver's frame rectangle in the super-layer's coordinate space.

```
@property CGRect frame
```

Discussion

The value of `frame` is derived from the `bounds` (page 16), `anchorPoint` (page 14) and `position` (page 25) properties. When the `frame` is set, the receiver's `position` (page 25) and the size of the receiver's `bounds` (page 16) are changed to match the new frame rectangle. The value of this property is specified in points.

See “Layer Geometry and Transforms” in *Core Animation Programming Guide* for more information on the relationship between the `bounds` (page 16), `anchorPoint` (page 14) and `position` (page 25) properties.

Note: The `frame` property is not directly animatable. Instead you should animate the appropriate combination of the `bounds` (page 16), `anchorPoint` (page 14) and `position` (page 25) properties to achieve the desired result.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

geometryFlipped

Determines if the geometry of the layer and its sublayers are flipped vertically.

```
@property(getter=isGeometryFlipped) BOOL geometryFlipped
```

Discussion

The value of this property does not effect the rendering of the layer's content, the image specified by contents will display the same regardless of the value of `geometryFlipped`.

Defaults to NO.

Availability

Available in iOS 3.0 and later.

Declared In

`CALayer.h`

hidden

Determines whether the receiver is displayed. Animatable.

```
@property(getter=isHidden) BOOL hidden
```

Discussion

The default is NO.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

magnificationFilter

The filter used when increasing the size of the content.

```
@property(copy) NSString *magnificationFilter
```

Discussion

The possible values for `magnificationFilter` are shown in “[Scaling Filters](#)” (page 54). The default value is `kCAFilterLinear` (page 54).

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

mask

An optional layer whose alpha channel is used as a mask to select between the layer's background and the result of compositing the layer's contents with its filtered background.

```
@property(retain) CALayer *mask
```

Discussion

Defaults to nil.

Special Considerations

When setting the `mask` to a new layer, the new layer's `superlayer` must first be set to nil, otherwise the behavior is undefined.

Availability

Available in iOS 3.0 and later.

Declared In

CALayer.h

masksToBounds

Determines if the sublayers are clipped to the receiver’s bounds. Animatable.

@property BOOL masksToBounds

Discussion

If YES, an implicit mask matching the layer bounds is applied to the layer, including the effects of the [cornerRadius](#) (page 19) property. If YES and a [mask](#) (page 22) property is specified, the two masks are multiplied to get the actual mask values. Defaults to NO.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

minificationFilter

The filter used when reducing the size of the content.

@property(copy) NSString *minificationFilter

Discussion

The possible values for `minificationFilter` are shown in “[Scaling Filters](#)” (page 54). The default value is [kCAFilterLinear](#) (page 54).

Availability

Available in iOS 2.0 and later.

See Also

[@property minificationFilterBias](#) (page 23)

Declared In

CALayer.h

minificationFilterBias

The bias factor used by the minification filter to determine the levels of detail.

@property float minificationFilterBias

Discussion

This value is used by the [minificationFilter](#) (page 23) when it is set to [kCAFilterTrilinear](#) (page 54).

Defaults to 0.

Availability

Available in iOS 3.0 and later.

Declared In

CALayer.h

name

The name of the receiver.

```
@property(copy) NSString *name
```

Discussion

The layer name is used by some layout managers to identify a layer. Defaults to `nil`.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

needsDisplayOnBoundsChange

Returns whether the receiver must be redisplayed when the bounds rectangle is updated.

```
@property BOOL needsDisplayOnBoundsChange
```

Discussion

When YES, `setNeedsDisplay` (page 47) is automatically invoked when the receiver's `bounds` (page 16) is changed. Default value is NO.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

opacity

Determines the opacity of the receiver. Animatable.

```
@property float opacity
```

Discussion

Possible values are between 0.0 (transparent) and 1.0 (opaque). The default is 1.0.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

opaque

This property affects how the layer creates the content directly.


```
@property(getter=isOpaque) BOOL opaque
```

Discussion

Defaults to NO.

Note that this property has no effect for images provided directly by the developer.

This property only affects what happens if `setNeedsDisplay` is called, and then `display` creates a bitmap for the `drawInContext:` method to draw into. In that case whether the generated bitmap has an alpha channel is defined by the value of this property.

This value has no effect for images provided directly by the developer in the `contents` property.

Availability

Available in iOS 2.0 and later.

Related Sample Code

[aurioTouch](#)

[GLSprite](#)

[SpeakHere](#)

Declared In

`CALayer.h`

position

Specifies the receiver's position in the superlayer's coordinate system. Animatable.

```
@property CGPoint position
```

Discussion

The position is relative to [anchorPoint](#) (page 14). The value of this property is specified in points. The default is (0.0, 0.0).

See “Layer Geometry and Transforms” in *Core Animation Programming Guide* for more information on the relationship between the [bounds](#) (page 16), [anchorPoint](#) (page 14) and [position](#) (page 25) properties.

Availability

Available in iOS 2.0 and later.

See Also

[@property anchorPoint](#) (page 14)

Declared In

`CALayer.h`

rasterizationScale

The scale at which to rasterize content, relative to the coordinate space of the layer. Animatable

@property CGFloat rasterizationScale

Discussion

When the value in the [shouldRasterize](#) (page 27) property is YES, the layer uses this property to determine whether to scale the rasterized content (and by how much). The default value of this property is 1.0, which indicates that the layer should be rasterized at its current size. Larger values magnify the content and smaller values shrink it.

Availability

Available in iOS 3.2 and later.

Declared In

CALayer.h

shadowColor

Specifies the color of the receiver's shadow. Animatable.

@property CGColorRef shadowColor

Discussion

The default is opaque black.

Availability

Available in iOS 3.2 and later.

Declared In

CALayer.h

shadowOffset

Specifies the offset of the receiver's shadow. Animatable.

@property CGSize shadowOffset

Discussion

The default is (0.0,-3.0).

Availability

Available in iOS 3.2 and later.

Declared In

CALayer.h

shadowOpacity

Specifies the opacity of the receiver's shadow. Animatable.

@property float shadowOpacity

Discussion

The default is 0.0.

Availability

Available in iOS 3.2 and later.

Declared In

CALayer.h

shadowPath

Defines the shape of the shadow.

@property CGPathRef shadowPath

Discussion

If the value in this property is non-`nil`, the shadow is created using the specified path instead of the layer's composited alpha channel. The path defines the outline of the shadow. It is filled using the non-zero winding rule and the current shadow color, opacity, and blur radius.

Specifying an explicit path usually improves rendering performance. The default value of this property is `NULL`.

Availability

Available in iOS 3.2 and later.

Declared In

CALayer.h

shadowRadius

Specifies the blur radius used to render the receiver's shadow. Animatable.

@property CGFloat shadowRadius

Discussion

The default value is 3.0.

Availability

Available in iOS 3.2 and later.

Declared In

CALayer.h

shouldRasterize

A Boolean that indicates whether the layer is rendered as a bitmap before compositing. Animatable

@property BOOL shouldRasterize

Discussion

When the value of this property is YES, the layer is rendered as a bitmap in its local coordinate space and then composited to the destination with any other content. Shadow effects and any filters in the [filters](#) (page 20) property are rasterized and included in the bitmap. However, the current opacity of the layer is not rasterized. If the rasterized bitmap requires scaling during compositing, the filters in the [minificationFilter](#) (page 23) and [magnificationFilter](#) (page 22) properties are applied as needed.

When the value of this property is NO, the layer is composited directly into the destination whenever possible. The layer may still be rasterized prior to compositing if certain features of the compositing model (such as the inclusion of filters) require it.

The default value of this property is NO.

Availability

Available in iOS 3.2 and later.

Declared In

CALayer.h

style

An optional dictionary referenced to find property values that aren't explicitly defined by the receiver.

@property(copy) NSDictionary *style

Discussion

This dictionary may in turn have a `style` key, forming a hierarchy of default values. In the case of hierarchical style dictionaries the shallowest value for a property is used. For example, the value for “`style.someValue`” takes precedence over “`style.style.someValue`”.

If the style dictionary doesn't define a value for an attribute, the receiver's [defaultValueForKey:](#) (page 31) method is called. Defaults to `nil`.

The style dictionary is not consulted for the following keys: `bounds`, `frame`.



Warning: If the style dictionary or any of its ancestors are modified, the values of the layer's properties are undefined until the `style` property is reset.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

sublayers

An array containing the receiver's sublayers.

```
@property(copy) NSArray *sublayers
```

Discussion

The layers are listed in back to front order. Defaults to `nil`.

Special Considerations

When setting the `sublayers` property to an array populated with layer objects you must ensure that the layers have had their [superlayer](#) (page 29) set to `nil`.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

sublayerTransform

Specifies a transform applied to each sublayer when rendering. Animatable.

```
@property CATransform3D sublayerTransform
```

Discussion

This property is typically used as the projection matrix to add perspective and other viewing effects to the receiver. Defaults to the identity transform.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

superlayer

Specifies receiver's superlayer. (read-only)

```
@property(readonly) CALayer *superlayer
```

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

transform

Specifies the transform applied to the receiver, relative to the center of its bounds. Animatable.

```
@property CATransform3D transform
```

Discussion

Defaults to the identity transform.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

visibleRect

Returns the visible region of the receiver, in its own coordinate space. (read-only)

```
@property(readonly) CGRect visibleRect
```

Discussion

The visible region is the area not clipped by the containing scroll layer.

Availability

Available in iOS 2.0 and later.

Declared In

CAScrollViewLayer.h

zPosition

Specifies the receiver's position on the z axis. Animatable.

```
@property CGFloat zPosition
```

Discussion

Defaults to 0.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

Class Methods

defaultActionForKey:

Returns an object that implements the default action for the specified identifier.

```
+ (id < CAAction >)defaultActionForKey:(NSString *)aKey
```

Parameters

aKey

The identifier of the action.

Return Value

Returns the object that provides the action for *aKey*.

Discussion

See [actionForKey:](#) (page 32) for a description of the action search pattern.

Availability

Available in iOS 2.0 and later.

See Also

- [actionForKey:](#) (page 32)
- [actionForLayer:forKey:](#) (page 49)
- [@property actions](#) (page 13)
- [@property style](#) (page 28)

Declared In

CALayer.h

defaultValueForKey:

Specifies the default value of the property with the specified key.

```
+ (id)defaultValueForKey:(NSString *)key
```

Parameters

key

The name of one of the receiver's properties.

Return Value

The default value for the named property. Returns `nil` if no default value has been set.

Discussion

If this method returns `nil` a suitable “zero” default value for the property is provided, based on the declared type of the key. For example, if *key* is a `CGSize` object, a size of (0.0,0.0) is returned. For a `CGRect` an empty rectangle is returned. For `CGAffineTransform` and `CATransform3D`, the appropriate identity matrix is returned.

Special Considerations

If *key* is not a known for property of the class, the result of the method is undefined.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

layer

Creates and returns an instance of `CALayer`.

```
+ (id)layer
```

Return Value

The initialized `CALayer` object or `nil` if initialization is not successful.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

needsDisplayForKey:

Returns whether changes to the specified key requires the layer to be redisplayed.

```
+ (BOOL)needsDisplayForKey:(NSString *)key
```

Parameters

key

A string that specifies an attribute of the layer.

Return Value

YES if the layer requires display.

Discussion

Subclasses should override this method and return YES if the layer should be redisplayed when the value of the specified attribute changes. Animations changing the value of the attribute will also trigger redisplay.

The default implementation returns NO.

Availability

Available in iOS 3.0 and later.

See Also

+ [defaultActionForKey:](#) (page 30)

+ [defaultValueForKey:](#) (page 31)

Declared In

CALayer.h

Instance Methods

actionForKey:

Returns an object that implements the action for the specified identifier.

```
- (id < CAAction >)actionForKey:(NSString *)aKey
```

Parameters

aKey

The identifier of the action.

Return Value

Returns the object that provides the action for *aKey*. The object must implement the `CAAction` protocol.

Discussion

There are three types of actions: property changes, externally-defined events, and layer-defined events. Whenever a layer property is modified, the event with the same name as the property is triggered. External events are defined by the owner of the layer calling `actionForKey:` to lookup the action associated with the identifier and directly messaging the returned object (if non-`nil`.)

The default implementation searches for an action object as follows:

- Return the value `NULL` if the search should not continue.
- If defined, return the object provided by the receiver's delegate method `actionForLayer:forKey:` (page 49).
- Return the object that corresponds to the identifier in the receiver's `actions` (page 13) dictionary property.
- If `nil` is returned there is no action specified for requested `aKey`.
- Search the `style` (page 28) dictionary recursively for an actions dictionary that contains the identifier.
- Call the receiver's `defaultActionForKey:` (page 30) method and return the result.

When an action object is invoked it receives three parameters: the name of the event, the object on which the event happened (the layer), and a dictionary of named arguments specific to each event kind.

Availability

Available in iOS 2.0 and later.

See Also

- `actionForLayer:forKey:` (page 49)
- `@property actions` (page 13)
- + `defaultActionForKey:` (page 30)
- `@property style` (page 28)

Declared In

`CALayer.h`

addAnimation:forKey:

Add an animation object to the receiver's render tree for the specified key.

```
- (void)addAnimation:(CAAnimation *)anim forKey:(NSString *)key
```

Parameters

anim

The animation to be added to the render tree. Note that the object is copied by the render tree, not referenced. Any subsequent modifications to the object will not be propagated into the render tree.

key

A string that specifies an identifier for the animation. Only one animation per unique key is added to the layer. The special key `kCATransition` (page 51) is automatically used for transition animations. The `nil` pointer is also a valid key.

Discussion

Typically this is implicitly invoked through an action that is an `CAAnimation` object. If the `duration` property of the animation is zero or negative it is given the default duration, either the current value of the `kCATransactionAnimationDuration` transaction property, otherwise .25 seconds

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

addSublayer:

Appends the layer to the receiver's `sublayers` (page 28) array.

```
- (void)addSublayer:(CALayer *)aLayer
```

Parameters

aLayer

The layer to be added to the receiver's `sublayers` (page 28) array.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

affineTransform

Convenience method for getting the `transform` (page 29) property as an affine transform.

```
- (CGAffineTransform)affineTransform
```

Return Value

A `CGAffineTransform` instance that best represents the receiver's `transform` (page 29) property.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

animationForKey:

Returns the animation added to the receiver with the specified identifier.

```
- (CAAnimation *)animationForKey:(NSString *)key
```

Parameters

key

A string that specifies the identifier of the animation.

Return Value

The animation object matching the identifier, or `nil` if no such animation exists.

Discussion

Attempting to modify any properties of the returned object will result in undefined behavior.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

animationKeys

Returns an array containing the keys of all animations currently attached to the receiver.

- (NSArray *)animationKeys

Return Value

An array of `NSString` objects representing the layer's animations.

Discussion

The order of the array matches the order in which animations will be applied.

Availability

Available in iOS 3.0 and later.

Declared In

`CALayer.h`

containsPoint:

Returns whether the receiver contains a specified point.

- (BOOL)containsPoint:(CGPoint)thePoint

Parameters

thePoint

A point in the receiver's coordinate system.

Return Value

YES if the bounds of the layer contains the point.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

contentsAreFlipped

Returns whether the layer content is implicitly flipped when rendered.

- (BOOL)contentsAreFlipped

Return Value

YES if the layer contents are implicitly flipped when rendered.

Discussion

When this method returns YES the CGContextRef object passed to [drawInContext:](#) (page 39) by the default [display](#) (page 38) method will have been y- flipped and rectangles passed to [setNeedsDisplayInRect:](#) (page 48) will be similarly flipped.

Defaults to NO.

Subclasses should not attempt to redefine this method.

Availability

Available in iOS 3.0 and later.

Declared In

CALayer.h

convertPoint:fromLayer:

Converts the point from the specified layer's coordinate system to the receiver's coordinate system.

```
- (CGPoint)convertPoint:(CGPoint)aPoint fromLayer:(CALayer *)layer
```

Parameters

aPoint

A point specifying a location in the coordinate system of *layer*.

layer

The layer with *aPoint* in its coordinate system. The receiver and *layer* and must share a common parent layer.

Return Value

The point converted to the receiver's coordinate system.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

convertPoint:toLayer:

Converts the point from the receiver's coordinate system to the specified layer's coordinate system.

```
- (CGPoint)convertPoint:(CGPoint)aPoint toLayer:(CALayer *)layer
```

Parameters

aPoint

A point specifying a location in the coordinate system of *layer*.

layer

The layer into whose coordinate system *aPoint* is to be converted. The receiver and *layer* must share a common parent layer.

Return Value

The point converted to the coordinate system of *layer*.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

convertRect:fromLayer:

Converts the rectangle from the specified layer's coordinate system to the receiver's coordinate system.

```
- (CGRect)convertRect:(CGRect)aRect fromLayer:(CALayer *)layer
```

Parameters

aRect

A point specifying a location in the coordinate system of *layer*.

layer

The layer with *aRect* in its coordinate system. The receiver and *layer* and must share a common parent layer.

Return Value

The rectangle converted to the receiver's coordinate system.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

convertRect:toLayer:

Converts the rectangle from the receiver's coordinate system to the specified layer's coordinate system.

```
- (CGRect)convertRect:(CGRect)aRect toLayer:(CALayer *)layer
```

Parameters

aRect

A point specifying a location in the coordinate system of *layer*.

layer

The layer into whose coordinate system *aRect* is to be converted. The receiver and *layer* and must share a common parent layer.

Return Value

The rectangle converted to the coordinate system of *layer*.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

convertTime:fromLayer:

Converts the time interval from the specified layer's time space to the receiver's time space.

```
- (CFTimeInterval)convertTime:(CFTimeInterval)timeInterval fromLayer:(CALayer *)layer
```

Parameters*timeInterval*

A point specifying a location in the coordinate system of *layer*.

layer

The layer with *timeInterval* in its time space. The receiver and *layer* and must share a common parent layer.

Return Value

The time interval converted to the receiver's time space.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

convertTime:toLayer:

Converts the time interval from the receiver's time space to the specified layer's time space

```
- (CFTimeInterval)convertTime:(CFTimeInterval)timeInterval toLayer:(CALayer *)layer
```

Parameters*timeInterval*

A point specifying a location in the coordinate system of *layer*.

layer

The layer into whose time space *timeInterval* is to be converted. The receiver and *layer* and must share a common parent layer.

Return Value

The time interval converted to the time space of *layer*.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

display

Reload the content of this layer.

- (void)display

Discussion

Calls the [drawInContext:](#) (page 39) method, then updates the receiver’s [contents](#) (page 17) property. You should not call this method directly.

Subclasses can override this method to set the [contents](#) (page 17) property to an appropriate `CGImageRef`.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

displayIfNeeded

Displays the layer if it has been marked as needing display.

- (void)displayIfNeeded

Discussion

When this message is received the layer will invoke [display](#) (page 38) if it has been marked as requiring display.

Availability

Available in iOS 3.0 and later.

See Also

- [needsDisplay](#) (page 43)

Declared In

`CALayer.h`

drawInContext:

Draws the receiver’s content in the specified graphics context.

- (void)drawInContext:(CGContextRef)ctx

Parameters

ctx

The graphics context in which to draw the content.

Discussion

Default implementation does nothing. The context may be clipped to protect valid layer content. Subclasses that wish to find the actual region to draw can call `CGContextGetClipBoundingBox`. Called by the [display](#) (page 38) method when the [contents](#) (page 17) property is being updated.

Subclasses can override this method to draw the receiver’s content. When drawing, all coordinates should be specified in the logical coordinate space—that is, measured in points.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

hitTest:

Returns the farthest descendant of the receiver in the layer hierarchy (including itself) that contains a specified point.

```
- (CALayer *)hitTest:(CGPoint)thePoint
```

Parameters*thePoint*

A point in the coordinate system of the receiver's superlayer.

Return Value

The layer that contains *thePoint*, or *nil* if the point lies outside the receiver's bounds rectangle.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

init

Returns an initialized CALayer object.

```
- (id)init
```

Return Value

An initialized CALayer object.

Discussion

This is the designated initializer for CALayer.

Availability

Available in iOS 2.0 and later.

See Also

+ [layer](#) (page 31)

Declared In

CALayer.h

initWithLayer:

Override to copy or initialize custom fields of the specified layer.

```
- (id)initWithLayer:(id)layer
```


Parameters*layer*

The layer from which custom fields should be copied.

Return Value

A layer instance with any custom instance variables copied from *layer*.

Discussion

This initializer is used to create shadow copies of layers, for example, for the `presentationLayer` method.

Subclasses can optionally copy their instance variables into the new object.

Subclasses should always invoke the superclass implementation

Note: Invoking this method in any other situation will produce undefined behavior. Do not use this method to initialize a new layer with an existing layer's content.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

insertSublayer:above:

Inserts the layer into the receiver's sublayers array, above the specified sublayer.

```
- (void)insertSublayer:(CALayer *)aLayer above:(CALayer *)siblingLayer
```

Parameters*aLayer*

The layer to be inserted to the receiver's sublayer array.

sublayer

An existing sublayer in the receiver to insert *aLayer* above.

Special Considerations

If *sublayer* is not in the receiver's [sublayers](#) (page 28) array, an exception is raised.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

insertSublayer:atIndex:

Inserts the layer as a sublayer of the receiver at the specified index.

```
- (void)insertSublayer:(CALayer *)aLayer atIndex:(unsigned)index
```

Parameters*aLayer*

The layer to be inserted to the receiver's sublayer array.

*index*The index in the receiver at which to insert *aLayer*. This value must not be greater than the count of elements in the sublayer array.**Availability**

Available in iOS 2.0 and later.

Declared In

CALayer.h

insertSublayer:below:

Inserts the layer into the receiver's sublayers array, below the specified sublayer.

- (void)insertSublayer:(CALayer *)*aLayer* below:(CALayer *)*sublayer***Parameters***aLayer*

The layer to be inserted to the receiver's sublayer array.

*sublayer*An existing sublayer in the receiver to insert *aLayer* after.**Discussion**If *sublayer* is not in the receiver's [sublayers](#) (page 28) array, an exception is raised.**Availability**

Available in iOS 2.0 and later.

Declared In

CALayer.h

layoutIfNeeded

Recalculate the receiver's layout, if required.

- (void)layoutIfNeeded

Discussion

When this message is received, the layer's superlayers are traversed until an ancestor layer is found that does not require layout. Then layout is performed on the entire layer-tree beneath that ancestor.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

layoutSublayers

Called when the layer requires layout.

```
- (void)layoutSublayers
```

Discussion

The default implementation invokes the layout manager method `layoutSublayersOfLayer:`, if a layout manager is specified and it implements that method. Subclasses can override this method to provide their own layout algorithm, which must set the frame of each sublayer.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

modelLayer

Returns the model layer of the receiver, if it represents a current presentation layer.

```
- (id)modelLayer
```

Return Value

A layer instance representing the underlying model layer.

Discussion

The result of calling this method after the transaction that produced the presentation layer has completed is undefined.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

needsDisplay

Returns whether the layer has been marked as requiring display.

```
- (BOOL)needsDisplay
```

Return Value

YES if the layer has been marked as requiring display.

Availability

Available in iOS 3.0 and later.

Declared In

`CALayer.h`

needsLayout

Returns whether the layer has been marked as requiring layout.

- (BOOL)needsLayout

Return Value

YES if the layer has been marked as requiring layout.

Availability

Available in iOS 3.0 and later.

See Also

- [setNeedsLayout](#) (page 48)

Declared In

CALayer.h

preferredFrameSize

Returns the preferred frame size of the layer in the coordinate space of the superlayer.

- (CGSize)preferredFrameSize

Return Value

Returns the receiver's preferred frame size.

Discussion

The default implementation calls the layout manager, if one exists and it implements the `preferredSizeOfLayer:` method. Otherwise, it returns the size of the receiver's [bounds](#) (page 16) rectangle mapped into coordinate space of the receiver's [superlayer](#) (page 29).

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

presentationLayer

Returns a copy of the layer containing all properties as they were at the start of the current transaction, with any active animations applied.

- (id)presentationLayer

Return Value

A layer instance representing the current presentation layer.

Discussion

This method provides a close approximation to the version of the layer that is currently being displayed. The [sublayers](#) (page 28), [mask](#) (page 22), and [superlayer](#) (page 29) properties of the returned layer return the presentation versions of these properties. This pattern carries through to the read-only layer methods. For example, sending a [hitTest:](#) (page 40) message to the `presentationLayer` will query the presentation values of the layer tree.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

removeAllAnimations

Remove all animations attached to the receiver.

```
- (void)removeAllAnimations
```

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

removeAnimationForKey:

Remove the animation attached to the receiver with the specified key.

```
- (void)removeAnimationForKey:(NSString *)key
```

Parameters

key

The identifier of the animation to remove.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

removeFromSuperlayer

Removes the layer from the [sublayers](#) (page 28) array or [mask](#) (page 22) property of the receiver's [superlayer](#) (page 29).

```
- (void)removeFromSuperlayer
```

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

renderInContext:

Renders the receiver and its sublayers into the specified context.

```
- (void)renderInContext:(CGContextRef)ctx
```

Parameters

ctx

The graphics context that the content is rendered in to.

Discussion

This method renders directly from the layer tree, ignoring any animations added to the render tree. Renders in the coordinate space of the layer.

Important: The Mac OS X v10.5 implementation of this method does not support the entire Core Animation composition model. `QCCompositionLayer`, `CAOpenGLLayer`, and `QTMovieLayer` layers are not rendered. Additionally, layers that use 3D transforms are not rendered, nor are layers that specify `backgroundFilters` (page 15), `filters` (page 20), `compositingFilter` (page 16), or a `mask` (page 22) values. Future versions of Mac OS X may add support for rendering these layers and properties.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

replaceSublayer:with:

Replaces the layer in the receiver's sublayers array with the specified new layer.

```
- (void)replaceSublayer:(CALayer *)oldLayer with:(CALayer *)newLayer
```

Parameters

oldLayer

The layer to be replaced to the receiver's sublayer array.

newLayer

The layer with which to replace *oldLayer* in the receiver's sublayer array.

Discussion

If the receiver is not the superlayer of *oldLayer* the behavior is undefined.

Availability

Available in iOS 2.0 and later.

Declared In

`CALayer.h`

scrollPoint:

Scrolls the receiver's closest ancestor `CAScrollLayer` so that the specified point lies at the origin of the layer.

```
- (void)scrollPoint:(CGPoint)thePoint
```

Parameters*thePoint*

The point in the receiver to scroll to.

Availability

Available in iOS 2.0 and later.

Declared In

CAscrollLayer.h

scrollRectToVisible:Scrolls the receiver's closest ancestor `CAscrollLayer` the minimum distance needed so that the specified rectangle becomes visible.- (void)scrollRectToVisible:(CGRect)*theRect***Parameters***theRect*

The rectangle to be made visible.

Availability

Available in iOS 2.0 and later.

Declared In

CAscrollLayer.h

setAffineTransform:Convenience method for setting the [transform](#) (page 29) property as an affine transform.- (void)setAffineTransform:(CGAffineTransform)*m***Parameters***m*The affine transform to set as the [transform](#) (page 29) property.**Availability**

Available in iOS 2.0 and later.

Declared In

CALayer.h

setNeedsDisplay

Marks the receiver as needing display before the content is next committed.

- (void)setNeedsDisplay

DiscussionCalling this method will cause the receiver to recache its content. This will result in the layer receiving a [drawInContext:](#) (page 39) which may result in the delegate receiving either a [displayLayer:](#) (page 49) or [drawLayer:inContext:](#) (page 50) message.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

setNeedsDisplayInRect:

Marks the region of the receiver within the specified rectangle as needing display.

```
- (void)setNeedsDisplayInRect:(CGRect)theRect
```

Parameters

theRect

The rectangular region of the receiver to mark as invalid; it should be specified in the coordinate system of the receiver.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

setNeedsLayout

Called when the preferred size of the receiver may have changed.

```
- (void)setNeedsLayout
```

Discussion

This method is typically called when the receiver's sublayers have changed. It marks that the receiver sublayers must update their layout (by invoking [layoutSublayers](#) (page 43) on the receiver and all its superlayers). If the receiver's layout manager implements the `invalidateLayoutOfLayer:` method it is called.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

shouldArchiveValueForKey:

Specifies whether the value of the property for a given key is archived.

```
- (BOOL)shouldArchiveValueForKey:(NSString *)key
```

Parameters

key

The name of one of the receiver's properties.

Return Value

YES if the specified property should be archived, otherwise NO.

Discussion

The default implementation returns YES. Called by the object's implementation of `encodeWithCoder:`.

Availability

Available in iOS 4.0 and later.

Declared In

CALayer.h

Delegate Methods

actionForLayer:forKey:

Allows the delegate to customize the action for a layer.

```
- (id < CAAction >)actionForLayer:(CALayer *)layer forKey:(NSString *)event
```

Parameters

layer

The layer that is the target of the action.

key

The identifier of the action.

Return Value

Returns an object implementing the `CAAction` protocol. May return `nil` if the delegate doesn't specify a behavior for `key`.

Discussion

See [actionForKey:](#) (page 32) for a description of the action search pattern.

Availability

Available in iOS 2.0 and later.

See Also

- [actionForLayer:forKey:](#) (page 49)

[@property actions](#) (page 13)

+ [defaultActionForKey:](#) (page 30)

[@property style](#) (page 28)

Declared In

CALayer.h

displayLayer:

Allows the delegate to override the `display` (page 38) implementation.

```
- (void)displayLayer:(CALayer *)layer
```

Parameters*layer*

The layer to display.

Discussion

If defined, called by the default implementation of `display`, in which case it should set the layer's contents property.

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

drawLayer:inContext:

Allows the delegate to override the layer's `drawInContext:` implementation.

```
- (void)drawLayer:(CALayer *)layer inContext:(CGContextRef)ctx
```

Parameters*layer*

The layer to draw the content of.

ctx

The graphics context to draw in to.

Discussion

If defined, called by the default implementation of [drawInContext:](#) (page 39).

Availability

Available in iOS 2.0 and later.

Declared In

CALayer.h

Constants

Action Identifiers

These constants are the predefined action identifiers used by [actionForKey:](#) (page 32), [addAnimation:forKey:](#) (page 33), [defaultActionForKey:](#) (page 30), [removeAnimationForKey:](#) (page 45), [actionForLayer:forKey:](#) (page 49), and the `CAAction` protocol method `runActionForKey:object:arguments:.`

```
NSString * const kCAOnOrderIn;
NSString * const kCAOnOrderOut;
NSString * const kCATransition;
```

Constants

kCAOnOrderIn

The identifier that represents the action taken when a layer becomes visible, either as a result being inserted into the visible layer hierarchy or the layer is no longer set as hidden.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

kCAOnOrderOut

The identifier that represents the action taken when the layer is removed from the layer hierarchy or is hidden.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

kCATransition

The identifier that represents a transition animation.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

Declared In

`CALayer.h`

Edge Antialiasing Mask

This mask is used by the [edgeAntialiasingMask](#) (page 20) property.

```
enum CAEdgeAntialiasingMask
{
    kCALayerLeftEdge    = 1U << 0,
    kCALayerRightEdge   = 1U << 1,
    kCALayerBottomEdge  = 1U << 2,
    kCALayerTopEdge     = 1U << 3,
};
```

Constants

kCALayerLeftEdge

Specifies that the left edge of the receiver's content should be antialiased.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

kCALayerRightEdge

Specifies that the right edge of the receiver's content should be antialiased.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

kCALayerBottomEdge

Specifies that the bottom edge of the receiver's content should be antialiased.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCALayerTopEdge`

Specifies that the top edge of the receiver's content should be antialiased.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

Declared In

`CALayer.h`

Contents Gravity Values

The contents gravity constants specify the position of the content object when the layer bounds is larger than the bounds of the content object. The are used by the `contentsGravity` (page 18) property.

```
NSString * const kCAGravityCenter;
NSString * const kCAGravityTop;
NSString * const kCAGravityBottom;
NSString * const kCAGravityLeft;
NSString * const kCAGravityRight;
NSString * const kCAGravityTopLeft;
NSString * const kCAGravityTopRight;
NSString * const kCAGravityBottomLeft;
NSString * const kCAGravityBottomRight;
NSString * const kCAGravityResize;
NSString * const kCAGravityResizeAspect;
NSString * const kCAGravityResizeAspectFill;
```

Constants

`kCAGravityCenter`

The content is horizontally and vertically centered in the bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityTop`

The content is horizontally centered at the top-edge of the bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityBottom`

The content is horizontally centered at the bottom-edge of the bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityLeft`

The content is vertically centered at the left-edge of the bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityRight`

The content is vertically centered at the right-edge of the bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityTopLeft`

The content is positioned in the top-left corner of the bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityTopRight`

The content is positioned in the top-right corner of the bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityBottomLeft`

The content is positioned in the bottom-left corner of the bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityBottomRight`

The content is positioned in the bottom-right corner of the bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityResize`

The content is resized to fit the entire bounds rectangle.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityResizeAspect`

The content is resized to fit the bounds rectangle, preserving the aspect of the content. If the content does not completely fill the bounds rectangle, the content is centered in the partial axis.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

`kCAGravityResizeAspectFill`

The content is resized to completely fill the bounds rectangle, while still preserving the aspect of the content. The content is centered in the axis it exceeds.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

Declared In

`CALayer.h`

Identity Transform

Defines the identity transform matrix used by Core Animation.

```
const CATransform3D CATransform3DIdentity
```

Constants

`CATransform3DIdentity`

The identity transform: `[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1]`.

Available in iOS 2.0 and later.

Declared in `CATransform3D.h`.

Declared In

CATransform3D.h

Scaling Filters

These constants specify the scaling filters used by [magnificationFilter](#) (page 22) and [minificationFilter](#) (page 23).

```
NSString * const kCAFilterLinear;
NSString * const kCAFilterNearest;
NSString * const kCAFilterTrilinear;
```

Constants

kCAFilterLinear

Linear interpolation filter.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

kCAFilterNearest

Nearest neighbor interpolation filter.

Available in iOS 2.0 and later.

Declared in `CALayer.h`.

kCAFilterTrilinear

Trilinear minification filter. Enables mipmap generation. Some renderers may ignore this, or impose additional restrictions, such as source images requiring power-of-two dimensions..

Available in iOS 3.0 and later.

Declared in `CALayer.h`.**Declared In**

CALayer.h

Transform

Defines the standard transform matrix used throughout Core Animation.

```
struct CATransform3D
{
    CGFloat m11, m12, m13, m14;
    CGFloat m21, m22, m23, m24;
    CGFloat m31, m32, m33, m34;
    CGFloat m41, m42, m43, m44;
};
typedef struct CATransform3D CATransform3D;
```

Fields

m11

The entry at position 1,1 in the matrix.

m12

The entry at position 1,2 in the matrix.

m13

The entry at position 1,3 in the matrix.

m14	The entry at position 1,4 in the matrix.
m21	The entry at position 2,1 in the matrix.
m22	The entry at position 2,2 in the matrix.
m23	The entry at position 2,3 in the matrix.
m24	The entry at position 2,4 in the matrix.
m31	The entry at position 3,1 in the matrix.
m32	The entry at position 3,2 in the matrix.
m33	The entry at position 3,3 in the matrix.
m34	The entry at position 3,4 in the matrix.
m41	The entry at position 4,1 in the matrix.
m42	The entry at position 4,2 in the matrix.
m43	The entry at position 4,3 in the matrix.
m44	The entry at position 4,4 in the matrix.

Discussion

The transform matrix is used to rotate, scale, translate, skew, and project the layer content. Functions are provided for creating, concatenating, and modifying CATransform3D data.

Availability

Available in iOS 2.0 and later.

Declared In

CATransform3D.h

Document Revision History

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

Date	Notes
2010-06-11	Added symbols introduced in iOS 4.0.
2010-02-25	Added the shadowPath property, which is new in iOS 3.2.
2009-11-17	Updated forKey: method description
2009-04-20	Updated for iOS 3.0. Added new properties and methods. It is now possible to animate your own properties.
2009-02-04	Added -init method description.
2008-10-15	Updated frame property description.
2008-09-09	Corrected containsPoint: coordinate system reference.
2008-05-27	Updated for iOS.
2008-03-10	Corrected method signature for removeAnimationForKey:.
2007-12-11	Added discussion of renderInContext: limitations.
2007-07-24	New document that describes the class that defines the basic drawing, timespace, and animation management for Core Animation.

REVISION HISTORY

Document Revision History