

---

# CGGeometry Reference

[Graphics & Imaging](#) > Quartz



2009-01-06



Apple Inc.  
© 2003, 2009 Apple Inc.  
All rights reserved.

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

The Apple logo is a trademark of Apple Inc.

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

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

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

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

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

iPhone is a trademark of Apple 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

## CGGeometry Reference 5

---

Overview	5
Functions by Task	5
Creating a Geometric Primitive From a Dictionary Representation	5
Creating a Dictionary Representation From a Geometric Primitive	5
Creating a Geometric Primitive From Values	6
Modifying Rectangles	6
Comparing Values	6
Checking for Membership	6
Getting Min, Mid, and Max Values	7
Getting Height and Width	7
Checking Rectangle Characteristics	7
Functions	7
CGPointCreateDictionaryRepresentation	7
CGPointEqualToPoint	8
CGPointMake	8
CGPointMakeWithDictionaryRepresentation	9
CGRectContainsPoint	9
CGRectContainsRect	10
CGRectCreateDictionaryRepresentation	10
CGRectDivide	11
CGRectEqualToRect	11
CGRectGetHeight	12
CGRectGetMaxX	12
CGRectGetMaxY	13
CGRectGetMidX	13
CGRectGetMidY	14
CGRectGetMinX	14
CGRectGetMinY	15
CGRectGetWidth	15
CGRectInset	16
CGRectIntegral	17
CGRectIntersection	17
CGRectIntersectsRect	18
CGRectIsEmpty	18
CGRectIsInfinite	19
CGRectIsIntegral	19
CGRectIsNull	20
CGRectMake	20
CGRectMakeWithDictionaryRepresentation	21
CGRectOffset	21

- CGRectStandardize 22
- CGRectUnion 23
- CGSizeCreateDictionaryRepresentation 23
- CGSizeEqualToSize 23
- CGSizeMake 24
- CGSizeMakeWithDictionaryRepresentation 24
- Data Types 25
  - CGPoint 25
  - CGRect 25
  - CGSize 26
- Constants 26
  - CGRectInfinite 26
  - Geometric Zeroes 27
  - Geometrical Null 27
  - CGRectEdge 28
  - CGFloat Informational Macros 29

---

**Document Revision History 31**

---

**Index 33**

---

# CGGeometry Reference

---

<b>Framework:</b>	ApplicationServices/ApplicationServices.h
<b>Companion guide</b>	Quartz 2D Programming Guide
<b>Declared in</b>	CABase.h CGGeometry.h

## Overview

*CGGeometry Reference* defines structures for geometric primitives and functions that operate on them. The data structure `CGPoint` represents a point in a two-dimensional coordinate system. The data structure `CGRect` represents the location and dimensions of a rectangle. The data structure `CGSize` represents the dimensions of width and height.

## Functions by Task

### Creating a Geometric Primitive From a Dictionary Representation

- [CGPointCreateDictionaryRepresentation](#) (page 7)  
Returns a dictionary representation of the provided point.
- [CGSizeCreateDictionaryRepresentation](#) (page 23)  
Returns a dictionary representation of the provided size.
- [CGRectCreateDictionaryRepresentation](#) (page 10)  
Returns a dictionary representation of the provided rectangle.

### Creating a Dictionary Representation From a Geometric Primitive

- [CGPointMakeWithDictionaryRepresentation](#) (page 9)  
Fills in a `CGPoint` structure using the contents of the provided dictionary.
- [CGSizeMakeWithDictionaryRepresentation](#) (page 24)  
Fills in a `CGSize` structure using the contents of the provided dictionary.
- [CGRectMakeWithDictionaryRepresentation](#) (page 21)  
Fills in a `CGRect` structure using the contents of the provided dictionary.

## Creating a Geometric Primitive From Values

[CGPointMake](#) (page 8)

Returns a `CGPoint` structure filled in with the coordinate values you provide.

[CGRectMake](#) (page 20)

Returns a `CGRect` structure filled in with the coordinate and dimension values you provide.

[CGSizeMake](#) (page 24)

Returns a `CGSize` structure filled in with dimension values you provide.

## Modifying Rectangles

[CGRectDivide](#) (page 11)

Divides a source rectangle into two component rectangles.

[CGRectInset](#) (page 16)

Returns a rectangle that is smaller or larger than the source rectangle, with the same center point.

[CGRectIntegral](#) (page 17)

Returns the smallest rectangle that results from converting the source rectangle values to integers.

[CGRectIntersection](#) (page 17)

Returns the intersection of two rectangles.

[CGRectOffset](#) (page 21)

Returns a rectangle with an origin that is offset from that of the source rectangle.

[CGRectStandardize](#) (page 22)

Returns a rectangle with a positive width and height.

[CGRectUnion](#) (page 23)

Returns the smallest rectangle that contains the two provided rectangles.

## Comparing Values

[CGPointEqualToPoint](#) (page 8)

Returns whether two points are equal.

[CGSizeEqualToSize](#) (page 23)

Returns whether two sizes are equal.

[CGRectEqualToRect](#) (page 11)

Returns whether two rectangles are equal in size and position.

[CGRectIntersectsRect](#) (page 18)

Returns whether two rectangles intersect.

## Checking for Membership

[CGRectContainsPoint](#) (page 9)

Returns whether a rectangle contains a specified point.

[CGRectContainsRect](#) (page 10)

Returns whether the first rectangle contains the second rectangle.

## Getting Min, Mid, and Max Values

[CGRectGetMinX](#) (page 14)

Returns the x-coordinate that establishes the left edge of a rectangle.

[CGRectGetMinY](#) (page 15)

Returns the y-coordinate that establishes the bottom edge of a rectangle.

[CGRectGetMidX](#) (page 13)

Returns the x-coordinate that establishes the center of a rectangle.

[CGRectGetMidY](#) (page 14)

Returns the y-coordinate that establishes the center of a rectangle.

[CGRectGetMaxX](#) (page 12)

Returns the x-coordinate that establishes the right edge of a rectangle.

[CGRectGetMaxY](#) (page 13)

Returns the y-coordinate that establishes the top edge of a rectangle.

## Getting Height and Width

[CGRectGetHeight](#) (page 12)

Returns the height of a rectangle.

[CGRectGetWidth](#) (page 15)

Returns the width of a rectangle.

## Checking Rectangle Characteristics

[CGRectIsEmpty](#) (page 18)

Returns whether a rectangle has zero width or height, or is a null rectangle.

[CGRectIsNull](#) (page 20)

Returns whether a rectangle is invalid.

[CGRectIsInfinite](#) (page 19)

Returns whether a rectangle is infinite.

[CGRectIsIntegral](#) (page 19)

Returns whether the origin and size of the rectangle can be represented exactly as integers.

# Functions

## **CGPointCreateDictionaryRepresentation**

Returns a dictionary representation of the provided point.

```
CFDictionaryRef CGPointCreateDictionaryRepresentation(
    CGPoint point
);
```

**Parameters***point*

A point.

**Return Value**

The dictionary representation of the point.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CGGeometry.h

**CGPointEqualToPoint**

Returns whether two points are equal.

```
bool CGPointEqualToPoint (
    CGPoint point1,
    CGPoint point2
);
```

**Parameters***point1*

The first point to examine.

*point2*

The second point to examine.

**Return Value**

Returns 1 if the two specified points are the same; otherwise, 0.

**Availability**

Available in Mac OS X version 10.0 and later.

**Declared In**

CGGeometry.h

**CGPointMake**Returns a `CGPoint` structure filled in with the coordinate values you provide.

```
CGPoint CGPointMake (
    CGFloat x,
    CGFloat y
);
```

**Parameters***x*

The x-coordinate of the point to construct.



*y*

The y-coordinate of the point to construct.

**Return Value**

Returns a `CGPoint` structure, representing a single (x,y) coordinate pair.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CALayerEssentials

CarbonSketch

**Declared In**

CGGeometry.h

**CGPointMakeWithDictionaryRepresentation**

Fills in a `CGPoint` structure using the contents of the provided dictionary.

```
bool CGPointMakeWithDictionaryRepresentation(
    NSDictionaryRef dict,
    CGPoint *point
);
```

**Parameters***dict*

A dictionary that was previously returned from the function [CGPointCreateDictionaryRepresentation](#) (page 7).

*point*

On return, the point created from the provided dictionary.

**Return Value**

true if successful; false otherwise.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CGGeometry.h

**CGRectContainsPoint**

Returns whether a rectangle contains a specified point.

```
bool CGRectContainsPoint (
    CGRect rect,
    CGPoint point
);
```

**Parameters***rect*

The rectangle to examine.

*point*

The point to examine.

**Return Value**

Returns 1 if the specified point is located within the specified rectangle; otherwise, 0.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CarbonSketch

**Declared In**

CGGeometry.h

## CGRectContainsRect

Returns whether the first rectangle contains the second rectangle.

```
bool CGRectContainsRect (
    CGRect rect1,
    CGRect rect2
);
```

**Parameters**

*rect1*

The rectangle to examine for containment of the rectangle passed in *rect2*.

*rect2*

The rectangle to examine for being contained in the rectangle passed in *rect1*.

**Return Value**

Returns 1 if the rectangle specified by *rect2* is contained in the rectangle passed in *rect1*; otherwise, 0. The first rectangle contains the second if the union of the two rectangles is equal to the first rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CarbonSketch

**Declared In**

CGGeometry.h

## CGRectCreateDictionaryRepresentation

Returns a dictionary representation of the provided rectangle.

```
CFDictionaryRef CGRectCreateDictionaryRepresentation(
    CGRect rect
);
```

**Parameters**

*rect*

A rectangle.

**Return Value**

The dictionary representation of the rectangle.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CGGeometry.h

**CGRectDivide**

Divides a source rectangle into two component rectangles.

```
void CGRectDivide (
    CGRect rect,
    CGRect *slice,
    CGRect *remainder,
    CGFloat amount,
    CGRectEdge edge
);
```

**Parameters**

*rect*

The source CGRect structure.

*slice*

On input, a pointer to an uninitialized CGRect structure. On return, a CGRect structure filled in with the specified edge and values that extends the distance beyond the edge specified by the *amount* parameter.

*remainder*

On input, a pointer to an uninitialized rectangle CGRect structure. On return, the CGRect structure contains the portion of the source CGRect structure that remains after CGRectEdge produces the “slice” rectangle.

*amount*

A distance from the rectangle side that is specified in the *edge* parameter. This distance defines the line, parallel to the specified side, that Quartz uses to divide the source CGRect structure.

*edge*

A CGRectEdge value (CGRectMinXEdge (page 28), CGRectMinYEdge (page 28), CGRectMaxXEdge (page 28), or CGRectMaxYEdge (page 28)) that specifies the side of the rectangle from which the distance passed in the *amount* parameter is measured. CGRectDivide produces a “slice” rectangle that contains the specified edge and extends *amount* distance beyond it.

**Availability**

Available in Mac OS X version 10.0 and later.

**Declared In**

CGGeometry.h

**CGRectEqualToRect**

Returns whether two rectangles are equal in size and position.

```
bool CGRectEqualToRect (
    CGRect rect1,
    CGRect rect2
);
```

**Parameters***rect1*

The first rectangle to examine.

*rect2*

The second rectangle to examine.

**Return Value**Returns 1 if the two specified rectangles have equal size and origin values, or are both `NULL`. Otherwise, returns 0.**Availability**

Available in Mac OS X version 10.0 and later.

**Declared In**

CGGeometry.h

**CGRectGetHeight**

Returns the height of a rectangle.

```
CGFloat CGRectGetHeight (
    CGRect rect
);
```

**Parameters***rect*

The rectangle to examine.

**Return Value**

The height of the specified rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CarbonSketch

HID Calibrator

HID Config Save

HID Explorer

WhackedTV

**Declared In**

CGGeometry.h

**CGRectGetMaxX**

Returns the x-coordinate that establishes the right edge of a rectangle.

```
CGFloat CGRectGetMaxX (  
    CGRect rect  
);
```

**Parameters**

*rect*  
The rectangle to examine.

**Return Value**

The x-coordinate of the top-right corner of the specified rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

HID Calibrator  
HID Explorer

**Declared In**

CGGeometry.h

## CGRectGetMaxY

Returns the y-coordinate that establishes the top edge of a rectangle.

```
CGFloat CGRectGetMaxY (  
    CGRect rect  
);
```

**Parameters**

*rect*  
The rectangle to examine.

**Return Value**

The y-coordinate of the top-right corner of the specified rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

HID Explorer

**Declared In**

CGGeometry.h

## CGRectGetMidX

Returns the x- coordinate that establishes the center of a rectangle.

```
CGFloat CGRectGetMidX (
    CGRect rect
);
```

**Parameters***rect*

The rectangle to examine.

**Return Value**

The x-coordinate of the center of the specified rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

HID Calibrator

**Declared In**

CGGeometry.h

**CGRectGetMidY**

Returns the y-coordinate that establishes the center of a rectangle.

```
CGFloat CGRectGetMidY (
    CGRect rect
);
```

**Parameters***rect*

The rectangle to examine.

**Return Value**

The y-coordinate of the center of the specified rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

HID Calibrator

HID Explorer

**Declared In**

CGGeometry.h

**CGRectGetMinX**

Returns the x-coordinate that establishes the left edge of a rectangle.

```
CGFloat CGRectGetMinX (  
    CGRect rect  
);
```

**Parameters**

*rect*  
The rectangle to examine.

**Return Value**

The x-coordinate of the bottom-left corner of the specified rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CarbonSketch  
HID Config Save  
HID Explorer

**Declared In**

CGGeometry.h

## CGRectGetMinY

Returns the y-coordinate that establishes the bottom edge of a rectangle.

```
CGFloat CGRectGetMinY (  
    CGRect rect  
);
```

**Parameters**

*rect*  
The rectangle to examine.

**Return Value**

The y-coordinate of the bottom-left corner of the specified rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CarbonSketch  
HID Config Save  
HID Explorer

**Declared In**

CGGeometry.h

## CGRectGetWidth

Returns the width of a rectangle.

```
CGFloat CGRectGetWidth (
    CGRect rect
);
```

**Parameters***rect*

The rectangle to examine.

**Return Value**

The width of the specified rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CarbonSketch

HID Calibrator

HID Config Save

HID Explorer

WhackedTV

**Declared In**

CGGeometry.h

**CGRectInset**

Returns a rectangle that is smaller or larger than the source rectangle, with the same center point.

```
CGRect CGRectInset (
    CGRect rect,
    CGFloat dx,
    CGFloat dy
);
```

**Parameters***rect*

The source CGRect structure.

*dx*

The x-coordinate value to use for adjusting the source rectangle. To create an inset rectangle, specify a positive value. To create a larger, encompassing rectangle, specify a negative value.

*dy*

The y-coordinate value to use for adjusting the source rectangle. To create an inset rectangle, specify a positive value. To create a larger, encompassing rectangle, specify a negative value.

**Return Value**

A filled-in CGRect structure. The origin value is offset in the x-axis by the distance specified by the *dx* parameter and in the y-axis by the distance specified by the *dy* parameter, and its size adjusted by  $(2*dx, 2*dy)$ , relative to the source rectangle. If *dx* and *dy* are positive values, then the rectangle's size is decreased. If *dx* and *dy* are negative values, the rectangle's size is increased.

**Availability**

Available in Mac OS X version 10.0 and later.



**Related Sample Code**

CarbonSketch

**Declared In**

CGGeometry.h

**CGRectIntegral**

Returns the smallest rectangle that results from converting the source rectangle values to integers.

```
CGRect CGRectIntegral (
    CGRect rect
);
```

**Parameters***rect*

The source rectangle.

**Return Value**

A filled-in `CGRect` structure whose values represent the rectangle with the smallest integer values for its origin and size that contains the source rectangle. That is, given a rectangle with fractional origin or size values, `CGRectIntegral` rounds the rectangle's origin downward and its size upward to the nearest whole integers, such that the result contains the original rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**See Also**[CGRectIsIntegral](#) (page 19)**Related Sample Code**

WhackedTV

**Declared In**

CGGeometry.h

**CGRectIntersection**

Returns the intersection of two rectangles.

```
CGRect CGRectIntersection (
    CGRect r1,
    CGRect r2
);
```

**Parameters***rect1*

The first source rectangle.

*rect2*

The second source rectangle.

**Return Value**

A filled-in `CGRect` structure that represents the intersection of the two specified rectangles. If the two rectangles do not intersect, returns the null rectangle. To check for this condition, use `CGRectIsNull` (page 20).

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

WhackedTV

**Declared In**

`CGGeometry.h`

**CGRectIntersectsRect**

Returns whether two rectangles intersect.

```
bool CGRectIntersectsRect (
    CGRect rect1,
    CGRect rect2
);
```

**Parameters**

*rect1*

The first rectangle to examine.

*rect2*

The second rectangle to examine.

**Return Value**

Returns 1 if the two specified rectangles intersect; otherwise, 0. The first rectangle intersects the second if the intersection of the rectangles is not equal to the null rectangle.

**Availability**

Available in Mac OS X version 10.0 and later.

**Declared In**

`CGGeometry.h`

**CGRectIsEmpty**

Returns whether a rectangle has zero width or height, or is a null rectangle.

```
bool CGRectIsEmpty (
    CGRect rect
);
```

**Parameters**

*rect*

The rectangle to examine.

**Return Value**

Returns 1 if the specified rectangle is empty; otherwise, 0.

**Discussion**

An empty rectangle is either a null rectangle or a valid rectangle with zero height or width. See also [CGRectIsNull](#) (page 20).

**Availability**

Available in Mac OS X version 10.0 and later.

**Declared In**

CGGeometry.h

**CGRectIsInfinite**

Returns whether a rectangle is infinite.

```
bool CGRectIsInfinite (
    CGRect rect
);
```

**Parameters**

*rect*

The rectangle to examine.

**Return Value**

Returns `true` if the specified rectangle is infinite, `false` otherwise.

**Discussion**

An infinite rectangle is one that has no defined bounds. Infinite rectangles can be created as output from a tiling filter. For example, the Core Image framework perspective tile filter creates an image whose extent is described by an infinite rectangle.

**Availability**

Available in Mac OS X v10.4 and later.

**Related Sample Code**

WhackedTV

**Declared In**

CGGeometry.h

**CGRectIsIntegral**

Returns whether the origin and size of the rectangle can be represented exactly as integers.

```
bool CGRectIsIntegral (
    CGRect rect
);
```

**Parameters**

*rect*

The rectangle to examine.

**Return Value**

Returns `true` if the origin and size of the rectangle can be represented exactly as integers; `false` otherwise.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CGGeometry.h

**CGRectIsNull**

Returns whether a rectangle is invalid.

```
bool CGRectIsNull (
    CGRect rect
);
```

**Parameters**

*rect*

The rectangle to examine.

**Return Value**

Returns 1 if the specified rectangle is null; otherwise, 0.

**Discussion**

A null rectangle is one that is not valid (you cannot draw a null rectangle). For example, the result of intersecting two disjoint rectangles is a null rectangle. See also [CGRectIsEmpty](#) (page 18).

**Availability**

Available in Mac OS X version 10.0 and later.

**Declared In**

CGGeometry.h

**CGRectMake**

Returns a CGRect structure filled in with the coordinate and dimension values you provide.

```
CGRect CGRectMake (
    CGFloat x,
    CGFloat y,
    CGFloat width,
    CGFloat height
);
```

**Parameters**

*x*

The x-coordinate of the rectangle's origin point.

*y*

The y-coordinate of the rectangle's origin point.

*width*

The width of the rectangle.

*height*

The height of the rectangle.

**Return Value**

Returns a rectangle with the specified location and dimensions.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CALayerEssentials

CarbonSketch

HID Calibrator

HID Explorer

QTCarbonShell

**Declared In**

CGGeometry.h

**CGRectMakeWithDictionaryRepresentation**

Fills in a `CGRect` structure using the contents of the provided dictionary.

```
bool CGRectMakeWithDictionaryRepresentation(
    CFDictionaryRef dict,
    CGRect *rect
);
```

**Parameters**

*dict*

A dictionary that was previously returned from the function [CGRectCreateDictionaryRepresentation](#) (page 10).

*rect*

On return, the rectangle created from the provided dictionary.

**Return Value**

true if successful; false otherwise.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CGGeometry.h

**CGRectOffset**

Returns a rectangle with an origin that is offset from that of the source rectangle.

```
CGRect CGRectOffset (
    CGRect rect,
    CGFloat dx,
    CGFloat dy
);
```

**Parameters***rect*

The source rectangle.

*dx*

The offset value for the x-coordinate.

*dy*

The offset value for the y-coordinate.

**Return Value**

A filled-in `CGRect` structure that is the same size as the source, but with its origin offset by `dx` units along the x-axis and `dy` units along the y-axis with respect to the source.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CarbonSketch

**Declared In**

CGGeometry.h

**CGRectStandardize**

Returns a rectangle with a positive width and height.

```
CGRect CGRectStandardize (
    CGRect rect
);
```

**Parameters***rect*

The source rectangle.

**Return Value**

A filled-in `CGRect` structure that represents the source rectangle, but with positive width and height values.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CarbonSketch

**Declared In**

CGGeometry.h

## CGRectUnion

Returns the smallest rectangle that contains the two provided rectangles.

```
CGRect CGRectUnion (
    CGRect r1,
    CGRect r2
);
```

### Parameters

*r1*  
The first source rectangle.

*r2*  
The second source rectangle.

### Return Value

A filled-in `CGRect` structure that represents the smallest rectangle that completely contains both of the source rectangles.

### Discussion

If one of the rectangles has 0 (or negative) width or height, a copy of the other rectangle is returned; but if both have 0 (or negative) width or height, the returned rectangle has its origin at (0.0, 0.0) and has 0 width and height.

### Availability

Available in Mac OS X version 10.0 and later.

### Declared In

`CGGeometry.h`

## CGSizeCreateDictionaryRepresentation

Returns a dictionary representation of the provided size.

```
CFDictionaryRef CGSizeCreateDictionaryRepresentation(
    CGSize size
);
```

### Parameters

*size*  
A size.

### Return Value

The dictionary representation of the size.

### Availability

Available in Mac OS X v10.5 and later.

### Declared In

`CGGeometry.h`

## CGSizeEqualToSize

Returns whether two sizes are equal.

```
bool CGSizeEqualToSize (
    CGSize size1,
    CGSize size2
);
```

**Parameters***size1*

The first size to examine.

*size2*

The second size to examine.

**Return Value**

Returns 1 if the two specified sizes are equal; otherwise, 0.

**Availability**

Available in Mac OS X version 10.0 and later.

**Declared In**

CGGeometry.h

**CGSizeMake**

Returns a CGSize structure filled in with dimension values you provide.

```
CGSize CGSizeMake (
    CGFloat width,
    CGFloat height
);
```

**Parameters***width*

A width value.

*height*

A height value.

**Return Value**

Returns a CGSize structure with the specified width and height.

**Availability**

Available in Mac OS X version 10.0 and later.

**Related Sample Code**

CarbonSketch

**Declared In**

CGGeometry.h

**CGSizeMakeWithDictionaryRepresentation**

Fills in a CGSize structure using the contents of the provided dictionary.



```
bool CGSizeMakeWithDictionaryRepresentation(
    CFDictionaryRef dict,
    CGSize *size
);
```

**Parameters***dict*

A dictionary that was previously returned from the function [CGSizeCreateDictionaryRepresentation](#) (page 23).

*size*

On return, the size created from the provided dictionary.

**Return Value**

true if successful; false otherwise.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CGGeometry.h

## Data Types

**CGPoint**

A structure that contains a point in a two-dimensional coordinate system.

```
struct CGPoint {
    CGFloat x;
    CGFloat y;
};
typedef struct CGPoint CGPoint;
```

**Fields***x*

The x-coordinate of the point.

*y*

The y-coordinate of the point.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CGGeometry.h

**CGRect**

A structure that contains the location and dimensions of a rectangle.

```

struct CGRect {
    CGPoint origin;
    CGSize size;
};
typedef struct CGRect CGRect;

```

**Fields**

origin

A [CGPoint](#) (page 25) structure that specifies the coordinates of the rectangle's origin. The origin is located in the lower-left of the rectangle.

size

A [CGSize](#) (page 26) structure that specifies the height and width of the rectangle.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CGGeometry.h

**CGSize**

A structure that contains width and height values.

```

struct CGSize {
    CGFloat width;
    CGFloat height;
};
typedef struct CGSize CGSize;

```

**Fields**

width

A width value.

height

A height value.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CGGeometry.h

## Constants

**CGRectInfinite**

A rectangle that has infinite extent.

```
const CGRect CGRectInfinite;
```

**Constants**

`CGRectInfinite`

A rectangle that has infinite extent.

Available in Mac OS X v10.4 and later.

Declared in `CGGeometry.h`.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

`CGGeometry.h`

## Geometric Zeroes

A zero point, zero rectangle, or zero size.

```
const CGPoint CGPointZero;
const CGRect CGRectZero;
const CGSize CGSizeZero;
```

**Constants**

`CGPointZero`

A point constant with location (0, 0). The zero point is equivalent to `CGPointMake(0,0)`.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

`CGRectZero`

A rectangle constant with location (0,0), and width and height of 0. The zero rectangle is equivalent to `CGRectMake(0,0,0,0)`.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

`CGSizeZero`

A size constant with width and height of 0. The zero size is equivalent to `CGSizeMake(0,0)`.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

**Declared In**

`CGGeometry.h`

## Geometrical Null

The null or empty rectangle.

```
const CGRect CGRectNull;
```

**Constants**

CGRectNull

The null rectangle. This is the rectangle returned when, for example, you intersect two disjoint rectangles. Note that the null rectangle is not the same as the zero rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

**Declared In**

`CGGeometry.h`

## CGRectEdge

Coordinates that establish the edges of a rectangle.

```
enum CGRectEdge {
    CGRectMinXEdge,
    CGRectMinYEdge,
    CGRectMaxXEdge,
    CGRectMaxYEdge
};
typedef enum CGRectEdge CGRectEdge;
```

**Constants**

CGRectMinXEdge

The x-coordinate that establishes the left edge of a rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

CGRectMinYEdge

The y-coordinate that establishes the minimum edge of a rectangle. In Mac OS X, this is typically the bottom edge of the rectangle. If the coordinate system is flipped (or if you are using the default coordinate system in iPhone OS), this constant refers to the top edge of the rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

CGRectMaxXEdge

The x-coordinate that establishes the right edge of a rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

CGRectMaxYEdge

The y-coordinate that establishes the maximum edge of a rectangle. In Mac OS X, this is typically the top edge of the rectangle. If the coordinate system is flipped (or if you are using the default coordinate system in iPhone OS), this constant refers to the bottom edge of the rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

**Declared In**

`CGGeometry.h`

## CGFloat Informational Macros

Informational macros for the `CGFloat` type.

```
#define CGFLOAT_MIN FLT_MIN // 32-bit
#define CGFLOAT_MAX FLT_MAX
#define CGFLOAT_IS_DOUBLE 0

#define CGFLOAT_MIN DBL_MIN // 64-bit
#define CGFLOAT_MAX DBL_MAX
#define CGFLOAT_IS_DOUBLE 1
```

### Constants

`CGFLOAT_MIN`

The minimum allowable value for a `CGFloat` type. For 32-bit code, this value is  $1.17549435e-38F$ . For 64-bit code, it is  $2.2250738585072014e-308$ .

Available in Mac OS X v10.5 and later.

Declared in `CABase.h`.

`CGFLOAT_MAX`

The maximum allowable value for a `CGFloat` type. For 32-bit code, this value is  $3.40282347e+38F$ . For 64-bit code, it is  $1.7976931348623157e+308$ .

Available in Mac OS X v10.5 and later.

Declared in `CABase.h`.

`CGFLOAT_IS_DOUBLE`

Indicates whether `CGFloat` is defined as a `float` or `double` type.

Available in Mac OS X v10.5 and later.

Declared in `CABase.h`.



# Document Revision History

This table describes the changes to *CGGeometry Reference*.

Date	Notes
2009-01-06	Updated the descriptions of the <code>CGRectMinYEdge</code> and <code>CGRectMaxYEdge</code> constants to reflect the different coordinate system possibilities.
2008-10-15	Added the definition for the <code>CGFloat</code> data type.
2008-04-08	Made minor technical corrections.
2006-12-22	Updated for Mac OS X v10.5.
	All instances of the <code>float</code> data type were changed to the <code>CGFloat</code> data type.
	Added <code>CGRectIsIntegral</code> (page 19), <code>CGPointCreateDictionaryRepresentation</code> (page 7), <code>CGSizeCreateDictionaryRepresentation</code> (page 23), <code>CGRectCreateDictionaryRepresentation</code> (page 10), <code>CGPointMakeWithDictionaryRepresentation</code> (page 9), <code>CGSizeMakeWithDictionaryRepresentation</code> (page 24), and <code>CGRectMakeWithDictionaryRepresentation</code> (page 21).
2005-04-29	Updated for Mac OS X v10.4.
	Added the function <code>CGRectIsIntegral</code> (page 19) and the constant “ <code>CGRectInfinite</code> ” (page 26).
2004-08-31	Added introductory material.
2004-02-26	First version of this document. An earlier version of this information appeared in <i>Quartz 2D Reference</i> .

## REVISION HISTORY

### Document Revision History



# Index

---

## C

---

**CGFloat** Informational Macros [29](#)  
**CGFLOAT\_IS\_DOUBLE** constant [29](#)  
**CGFLOAT\_MAX** constant [29](#)  
**CGFLOAT\_MIN** constant [29](#)  
**CGPoint** structure [25](#)  
**CGPointCreateDictionaryRepresentation** function [7](#)  
**CGPointEqualToPoint** function [8](#)  
**CGPointMake** function [8](#)  
**CGPointMakeWithDictionaryRepresentation** function [9](#)  
**CGPointZero** constant [27](#)  
**CGRect** structure [25](#)  
**CGRectContainsPoint** function [9](#)  
**CGRectContainsRect** function [10](#)  
**CGRectCreateDictionaryRepresentation** function [10](#)  
**CGRectDivide** function [11](#)  
**CGRectEdge** [28](#)  
**CGRectEqualToRect** function [11](#)  
**CGRectGetHeight** function [12](#)  
**CGRectGetMaxX** function [12](#)  
**CGRectGetMaxY** function [13](#)  
**CGRectGetMidX** function [13](#)  
**CGRectGetMidY** function [14](#)  
**CGRectGetMinX** function [14](#)  
**CGRectGetMinY** function [15](#)  
**CGRectGetWidth** function [15](#)  
**CGRectInfinite** [26](#)  
**CGRectInfinite** constant [27](#)  
**CGRectInset** function [16](#)  
**CGRectIntegral** function [17](#)  
**CGRectIntersection** function [17](#)  
**CGRectIntersectsRect** function [18](#)  
**CGRectIsEmpty** function [18](#)  
**CGRectIsInfinite** function [19](#)  
**CGRectIsIntegral** function [19](#)  
**CGRectIsNull** function [20](#)  
**CGRectMake** function [20](#)

**CGRectMakeWithDictionaryRepresentation** function [21](#)  
**CGRectMaxXEdge** constant [28](#)  
**CGRectMaxYEdge** constant [28](#)  
**CGRectMinXEdge** constant [28](#)  
**CGRectMinYEdge** constant [28](#)  
**CGRectNull** constant [28](#)  
**CGRectOffset** function [21](#)  
**CGRectStandardize** function [22](#)  
**CGRectUnion** function [23](#)  
**CGRectZero** constant [27](#)  
**CGSize** structure [26](#)  
**CGSizeCreateDictionaryRepresentation** function [23](#)  
**CGSizeEqualToSize** function [23](#)  
**CGSizeMake** function [24](#)  
**CGSizeMakeWithDictionaryRepresentation** function [24](#)  
**CGSizeZero** constant [27](#)

## G

---

**Geometric Zeroes** [27](#)  
**Geometrical Null** [27](#)