
Cursor Management

[Cocoa](#) > [User Experience](#)



2008-03-11



Apple Inc.
© 2008 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, Cocoa, and Objective-C are trademarks of Apple Inc., registered in the United States and other countries.

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Simultaneously published in the United States and Canada.

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

ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

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

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

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

Contents

Introduction to Cursor Management 5

Organization of This Document 5

About Cursors 7

Setting the Current Cursor 9

Document Revision History 11

Introduction to Cursor Management

Instances of the `NSCursor` class manage the appearance of the cursor.

Organization of This Document

The following articles provide an introduction to cursors and their usage:

- [“About Cursors”](#) (page 7) describes how cursors work.
- [“Setting the Current Cursor”](#) (page 9) describes how to change the current cursor being displayed.

About Cursors

Instances of the `NSCursor` class manage the appearance of the cursor. When you initialize a cursor—the designated initializer is `initWithImage:hotSpot:`—you assign it an `NSImage` object and a point to be the hot spot. The image is usually a small, opaque icon—for example, a pair of cross-hairs—surrounded by transparent pixels. The pixels in the cursor image are mapped on a flipped coordinate system with the upper left pixel being (0,0).

To determine exactly when the mouse is inside a particular cursor rectangle, the Application Kit tracks a single pixel in the cursor image. This pixel is known as the hot spot, and you can reference it using the `hotSpot` method. By definition, the location of the current cursor’s hot spot is the location of the mouse; when the hot spot is inside a cursor rectangle, so is the mouse. The hot spot is useful not only for determining which cursor is current, but for determining where a mouse click should have its effect.

An `NSCursor` object is immutable: you cannot change its hot spot or image after it’s created. Instead, use `initWithImage:hotSpot:` to create a new one with the new settings.

Setting the Current Cursor

An application may use several cursor instances—for example, one that looks like an arrow and one that looks like an I-beam. The instance that currently appears on the screen is called the “current cursor,” and is referenced by the `currentCursor` class method. You can set the current cursor in several ways:

- You can send a set message to the cursor.
- You can manage cursors in a stack, using the `push` and `pop` methods of `NSCursor`. The stack’s top cursor is the current cursor.
- You can tell a cursor to become current when the mouse enters a region in a view known as the cursor rectangle. The `NSView` class provides methods to support using cursor rectangles to change the cursor image. For more information, see *Cocoa Event-Handling Guide*.
- You can tell a cursor to set itself when the mouse exits a view’s cursor rectangle.

The cursor rectangle is a region inside an `NSView` that triggers a change in the current cursor. To create a cursor rectangle, use the `addCursorRect:cursor:` method of `NSView` to associate a region of the view with the cursor, as shown in the Objective-C example that follows. To make the association persistent, you can call `addCursorRect:cursor:` from within an override of `resetCursorRects` method of `NSView`, as described in *Cocoa Event-Handling Guide*.

```
[aView addCursorRect:aRect cursor:aCursor];  
[aCursor setOnMouseEntered:YES];
```

Here is a Java version of the same operation:

```
aView.addCursorRect(aRect, aCursor);  
aCursor.setOnMouseEntered(true);
```

This assignment means that when the mouse enters `aRect`, `aCursor` receives a `mouseEntered:` event message, which the cursor uses to make itself the current cursor. However, before the cursor can acknowledge the `mouseEntered:` message, you must invoke the cursor’s `setOnMouseEntered:` method. Alternatively, you can set the cursor when the mouse leaves the cursor rectangle by invoking the `setOnMouseExited:` method instead of `setOnMouseEntered:`. A cursor that sets itself upon leaving the cursor rectangle receives a `mouseExited:` event message to instigate the change.

The Application Kit provides two ready-made cursors for commonly used cursor images. You can retrieve these cursors by using the `arrowCursor` and `IBeamCursor` class methods. There is no `NSCursor` instance for the wait cursor, because the system automatically displays it at the appropriate times.

Document Revision History

This table describes the changes to *Cursor Management*.

Date	Notes
2008-03-11	Updated information about cursor size and types. Replaced one article with a more relevant article.
2006-03-08	Updated the table of contents to reflect all of the available articles.
2002-11-12	Revision history was added to existing topic. It will be used to record changes to the content of the topic.

