NSCursor

Inherits From:	NSObject
Conforms To:	NSCoding NSObject (NSObject)
Declared In:	AppKit/NSCursor.h

Class Description

Instances of the NSCursor class manage the appearance of the cursor. When you initialize an cursor—the designated initializer is **initWithImage:**—you assign it a 16-by-16 pixel NSImage. Later, if you want to change the image, you can use **setImage:**. The image is usually an 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: the upper left pixel is (0,0); the lower right is (15,15).

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 make a cursor current in several ways:

- Most simply, you can send a cursor a **set** message. This message makes the receiver current.
- You can manage cursors in a stack, using the **push** and **pop** methods. The stack's top cursor is current.
- You can tell a cursor to become current when the mouse enters a part of the screen known as the "cursor rectangle." To do this, you first assign the cursor to an NSRect, using NSView's **addCursorRect:cursor:** method:

  ```
  [aView addCursorRect:&aRect cursor:aCursor];
  ```

  This assignment means that when the mouse enters *aRect*, *aCursor* will receive a **mouseEntered:** event message. Next, using NSCursor's **setOnMouseEntered:** method, you tell *aCursor* to respond to the **mouseEntered:** event by setting itself—that is, by making itself current:

  ```
  [aCursor setOnMouseEntered:YES];
  ```

- Conversely, you can tell a cursor to set itself when the mouse exits the cursor's rectangle, using the **setOnMouseExited:** method.

To determine when exactly 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. You should usually set the hot spot, using **setHotSpot:**, to the point where users will focus their attention. In a cursor that looks like cross-hairs, for example, you should set the hot spot at the crossing point.

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's no NSCursor instance for the wait cursor, because the system automatically displays it at the appropriate times.

Adopted Protocols

NSCoding	– encodeWithCoder: – initWithCoder:

Method Types

Initializing a new cursor	– initWithImage:
Setting cursor attributes	– image – setImage: – hotSpot – setHotSpot: + hide + unhide + setHiddenUntilMouseMoves:
Controlling which cursor is current	+ pop – pop – push – set – mouseEntered: – setOnMouseEntered: – isSetOnMouseEntered – mouseExited: – setOnMouseExited: – isSetOnMouseExited
Retrieving cursor instances	+ arrowCursor + currentCursor + IBeamCursor

Class Methods

arrowCursor

+ (NSCursor *)**arrowCursor**

Returns the default cursor, a slanted arrow with its hot spot at the tip. The arrow cursor is the one you're used to seeing over buttons, cursors and many other objects in the window system.

See also: + **IBeamCursor**, + **currentCursor**, – **hotSpot**

currentCursor

+ (NSCursor *)**currentCursor**

Returns the cursor that's currently displayed on the screen.

See also: – **set**, – **push**, + **pop**, – **mouseEntered:**, – **mouseExited:**

hide

+ (void)**hide**

Makes the current cursor invisible. If another cursor becomes current, that cursor will be invisible, too. It will remain invisible until you invoke the **unhide** method.

hide overrides **setHiddenUntilMouseMoves:**.

IBeamCursor

+ (NSCursor *)**IBeamCursor**

Returns a cursor that looks like a capital I with a tiny crossbeam at its middle. This is the cursor that you're used to seeing over editable or selectable text. The I-beam cursor's default hot spot is where the crossbeam intersects the I.

See also: + **arrowCursor,** + **currentCursor**

pop

+ (void)**pop**

Sends a **pop** instance message to the cursor on top of the stack.

setHiddenUntilMouseMoves:

+ (void)**setHiddenUntilMouseMoves:**(BOOL)*flag*

If *flag* is YES, hides the cursor. The cursor remains invisible until either:

- the mouse moves, or
- you invoke the method again, with *flag* set to NO.

Don't try to counter this method with **unhide**. The results are undefined.

See also: + **hide**

unhide

+ (void)**unhide**

Negates an earlier call to **hide**.

See also: + **setHiddenUntilMouseMoves:**

Instance Methods

hotSpot

– (NSPoint)**hotSpot**

Returns the position of the hot spot, specified according to the cursor's flipped 16-by-16 coordinate system. For a fuller explanation, see the class description.

image

– (NSImage *)**image**

Returns the image that determines the appearance of the receiving cursor; if no image has been set, returns **nil**.

See also: – **initWithImage:**

initWithImage:

– (id)**initWithImage:**(NSImage *)*anImage*

This method is the designated initializer for the class. It initializes the receiver, assigns it *anImage* (which must be 16-by-16 pixels) and sets its hot spot to (0,0), the upper left corner of the image.

Later, you can change the image and hotSpot, using **setImage:** and **setHotSpot:**.

Returns **self**.

See also: – **hotSpot**, – **image**

isSetOnMouseEntered

– (BOOL)**isSetOnMouseEntered**

Returns YES if the receiving cursor will become current when it receives a **mouseEntered:** message; otherwise, returns NO.

To receive such a message, the receiver must first be assigned a cursor rectangle. This assignment can be made using NSView's **addCursorRect:cursor:** method. For a fuller explanation, see the class description.

See also: – **setOnMouseEntered:**, – **isSetOnMouseExited:**

isSetOnMouseExited

– (BOOL)**isSetOnMouseExited**

Returns YES if the receiving cursor will become current when it receives a **mouseExited:** message; otherwise, returns NO. The preconditions parallel those for **isSetOnMouseEntered:**.

See also: – **setOnMouseExited:**

mouseEntered:

– (void)**mouseEntered:**(NSEvent *)*anEvent*

This message is automatically sent to the receiver when the mouse enters the receiver's cursor rectangle. If used after **setOnMouseEntered:**YES, **mouseEntered:** can make the receiver the the current cursor.

In your programs, you won't invoke **mouseEntered:** explicity. It's only included in the class interface so that you can override it.

For a fuller explanation, see the class description.

See also: – **isSetOnMouseEntered, – mouseExited:**

mouseExited:

– (void)**mouseExited:**(NSEvent *)*theEvent*

This message is automatically sent to the receiver when the mouse exits the receiver's cursor rectangle. Like **mouseEntered:**, it is part of the class interface only so that you can override it.

See also: – **setOnMouseExited:**, – **isSetOnMouseExited**

pop

– (void)**pop**

Removes the receiver from the top of the cursor stack. Automatically, the next cursor down in the stack becomes the current cursor.

When the last remaining cursor is popped from the stack, the current cursor defaults to the arrow cursor.

See also: – **push**

push

– (void)**push**

Puts the receiver on top of the cursor stack. Automatically, the receiver becomes the current cursor.

See also: – **pop**

set

– (void)**set**

Sets the receiver to be the current cursor.

See also: + **currentCursor**

setHotSpot:

– (void)**setHotSpot:**(NSPoint)*hotSpot*

Sets the location of the cursor's hot spot. The hot spot should be specified in terms of the cursor's flipped 16-by-16 coordinate system.

See also: – **hotSpot**

setImage:

– (void)**setImage:**(NSImage *)*anImage*

Assigns *anImage* to the receiving cursor. *anImage* must be 16 pixels wide by 16 pixels high.

Don’t reset the image of the current cursor. The results may be unpredictable.

See also: + **currentCursor**, – **image**, – **initWithImage:**

setOnMouseEntered:

– (void)**setOnMouseEntered:**(BOOL)*flag*

Determines, based on *flag*, whether the NSCursor instance will set itself to be the current cursor when it receives a **mouseEntered:** event message.

setOnMouseExited:

– (void)**setOnMouseExited:**(BOOL)*flag*

Determines, based on *flag*, whether NSCursor will set itself to be the **currentCursor** when it receives a **mouseExited:** event message.