NSColorPanel

Inherits From:	NSPanel : NSWindow : NSResponder : NSObject
Conforms To:	NSCoding (NSResponder) NSObject (NSObject)
Declared In:	AppKit/NSColorPanel.h

Class Description

NSColorPanel provides a standard user interface for selecting color in an application. It provides a number of standard color selection modes, and, with the NSColorPickingDefault and NSColorPickingCustom protocols, allows an application to add its own color selection modes. It allows the user to save swatches containing frequently used colors. Once set, these swatches are displayed by NSColorPanel in any application where it is used, giving the user color consistency between applications. NSColorPanel enables users to capture a color anywhere on the screen for use in the active application, and allows dragging colors from itself into views in an application. NSColorPanel's action message is sent to the target object when the user changes the current color.

An application has only one instance of NSColorPanel, the shared instance. Invoking the **sharedColorPanel:** method returns the shared instance of NSColorPanel, instantiating it if necessary.

You can put NSColorPanel in any application created with Interface Builder by adding the "Colors..." item from the Menu palette to the application's menu.

Color Mask and Color Modes

The color mask determines which of the color modes are enabled for NSColorPanel. This mask is set before you initialize a new instance of NSColorPanel. NSColorPanelAllModesMask represents the logical OR of the other color mask constants: It causes the NSColorPanel to display all standard color pickers. When initializing a new instance of NSColorPanel, you can logically OR any combination of color mask constants to restrict the available color modes.

Mode	Color Mask Constant
Grayscale-Alpha	NSColorPanelGrayModeMask
Red-Green-Blue	NSColorPanelRGBModeMask
Cyan-Yellow-Magenta-Black	NSColorPanelCMYKModeMask
Hue-Saturation-Brightness	NSColorPanelHSBModeMask
Custom palette	NSColorPanelCustomPaletteModeMask
Custom color list	NSColorPanelColorListModeMask
Color wheel	NSColorPanelWheelModeMask
All of the above	NSColorPanelAllModesMask

The NSColorPanel's color mode mask is set using the class method **setPickerMask:**. The mask must be set before creating an application's instance of NSColorPanel.

When an application's instance of NSColorPanel is masked for more than one color mode, your program can set its active mode by invoking the **setMode:** method with a color mode constant as its argument; the user can set the mode by clicking buttons on the panel. Here are the standard color modes and mode constants:

Mode	Color Mode Constant
Grayscale-Alpha	NSGrayModeColorPanel
Red-Green-Blue	NSRGBModeColorPanel
Cyan-Yellow-Magenta-Black	NSCMYKModeColorPanel
Hue-Saturation-Brightness	NSHSBModeColorPanel
Custom palette	NSCustomPaletteModeColorPanel
Custom color list	NSColorListModeColorPanel
Color wheel	NSWheelModeColorPanel

In grayscale-alpha, red-green-blue, cyan-magenta-yellow-black, and hue-saturation-brightness modes, the user adjusts colors by manipulating sliders. In the custom palette mode, the user can load an NSImage file (TIFF or EPS) into the NSColorPanel, then select colors from the image. In custom color list mode, the user can create and load lists of named colors. The two custom modes provide NSPopUpButtons for loading and saving files. Finally, color wheel mode provides a simplified control for selecting colors.

If a color panel has been used, it uses whatever mode it was in last as the default mode when NSColorPanelAllModesMask is used to initialize the NSColorPanel. Otherwise, it uses color wheel mode.

Associated Classes and Protocols

The NSColorList class provides an API for managing custom color lists. The NSColorPanel methods **attachColorList:** and **detachColorList:** let your application add and remove custom lists from the NSColorPanel's user interface.

The protocols NSColorPickingDefault and NSColorPickingCustom provide an API for adding custom color selection to the user interface. The NSColorPicker class implements the NSColorPickingDefault protocol; you can subclass NSColorPicker and implement the NSColorPickingCustom protocol in your subclass to create your own user interface for color selection.

See also: NSColorList, NSColorPickingDefault, NSColorPicker, NSColorPickingDefault protocol, NSColorPickingCustom protocol, NSColorWell

Method Types

Creating the NSColorPanel	+ sharedColorPanel + sharedColorPanelExists
Setting color picker modes	+ setPickerMask: + setPickerMode:
Setting the NSColorPanel	– accessoryView – isContinuous – mode – setAccessoryView: – setAction: – setContinuous: – setMode: – setShowsAlpha: – setTarget: – showsAlpha
Attaching a color list	– attachColorList: – detachColorList:
Setting color	+ dragColor:withEvent:fromView: – setColor:
Getting color information	– alpha – color

Class Methods

dragColor:withEvent:fromView:

+ (BOOL)**dragColor:**(NSColor **)*color*
withEvent:(NSEvent *)*anEvent*
fromView:(NSView *)*sourceView*

Drags *color* into a destination view from *sourceView*. This method is usually invoked by the **mouseDown:** method of *sourceView*. The dragging mechanism handles all subsequent events.

Because it is a class method, **dragColor:withEvent:fromView:** can be invoked whether or not the instance of NSColorPanel exists. Returns YES.

setPickerMask:

+ (void)**setPickerMask:**(int)*mask*

Accepts as a parameter one or more logically OR'd color mode masks (defined in the header file **AppKit/NSColorPanel.h**):

- NSColorPanelGrayModeMask
- NSColorPanelRGBModeMask
- NSColorPanelCMYKModeMask
- NSColorPanelHSBModeMask
- NSColorPanelCustomPaletteModeMask
- NSColorPanelColorListModeMask
- NSColorPanelWheelModeMask
- NSColorPanelAllModesMask

This determines which color selection modes will be available in an application's NSColorPanel. This method only has an effect before NSColorPanel is instantiated.

If you create a class that implements the color picking protocols (NSColorPickingDefault and NSColorPickingCustom), you may want to give it a unique mask—one different from those defined for the standard color pickers. To display your color picker, your application will need to logically OR that unique mask with the standard color mask constants when invoking this method.

See also: **+ setPickerMode:**, NSColorPicker class, NSColorPickingDefault protocol, NSColorPickingCustom protocol

setPickerMode:

+ (void)**setPickerMode:**(int)*mode*

Sets the color panel's initial picker to *mode*, which may be one of the symbolic constants described in the class description (declared in the header file **AppKit/NSColorPanel.h**). The mode determines which picker will initially be visible. This method may be called at any time, whether or not an application's NSColorPanel has been instantiated.

See also: **+ setPickerMask:**, **– setMode:**

sharedColorPanel

+ (NSColorPanel *)**sharedColorPanel**

Creates if necessary and returns the shared NSColorPanel.

sharedColorPanelExists

+ (BOOL)**sharedColorPanelExists**

Returns YES if the NSColorPanel has been created already.

See also: + **sharedColorPanel**

Instance Methods

accessoryView

– (NSView *)**accessoryView**

Returns the accessory view, or **nil** if there is none.

See also: – **setAccessoryView:**

alpha

– (float)**alpha**

Returns the NSColorPanel’s current alpha value based on its opacity slider. Returns 1.0 (opaque) if the panel has no opacity slider.

See also: **– setShowsAlpha**, **– showsAlpha**

attachColorList:

– (void)**attachColorList:**(NSColorList *)*colorList*

Adds the specified list of NSColors to all the color pickers in the color panel that display color lists by invoking **attachColorList:** on all color pickers in the application.

An application should use this method to add an NSColorList saved with a document in its file package or in a directory other than NSColorList’s standard search directories.

See also: – **detachColorList**

color

– (NSColor *)**color**

Returns the currently selected color in the NSColorPanel.

See also: – **setColor**

detachColorList:

– (void)**detachColorList:**(NSColorList *)*colorList*

Removes the specified list of NSColors from all the color pickers in the color panel that display color lists by invoking **detachColorList:** on all color pickers in the application.

Your application should use this method to remove an NSColorList saved with a document in its file package or in a directory other than NSColorList's standard search directories.

See also: – **attachColorList**

isContinuous

– (BOOL)**isContinuous**

Returns whether or not the NSColorPanel continuously sends the action message to the target as the user manipulates the color picker.

See also: – **setContinuous**

mode

– (int)**mode**

Returns the color picker mode of the NSColorPanel. The mode constants for the standard color pickers are listed in the class description.

See also: **+ setPickerMode**, **– setMode**

setAccessoryView:

– (void)**setAccessoryView:**(NSView *)*aView*

Sets the accessory view displayed in the NSColorPanel to *aView*. The accessory view can be any custom view that you want to display with NSColorPanel, such as a view offering color blends in a drawing program. The accessory view is displayed below the color picker and above the color swatches in the

NSColorPanel. The NSColorPanel automatically resizes to accommodate the accessory view. Returns the previous accessory view, if there was one; otherwise, returns **nil**.

See also: – **accessoryView:**

setAction:

– (void)**setAction:**(SEL)*aSelector*

Sets the action message sent to *aSelector*. The NSColorPanel's action is the message sent to the target whenever the color is set in the color panel.

See also: – **setTarget**

setColor:

– (void)**setColor:**(NSColor *)*color*

Sets the color of the NSColorPanel to *color*. This method posts the NSColorPanelChangedNotification notification with the receiving object to the default notification center.

See also: – **color**

setContinuous:

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

Sets the NSColorPanel to send the action message to its target continuously as the color of the NSColorPanel is set by the user. Send this message with flag YES if, for example, you want to continuously update the color of the target.

See also: – **isContinuous**

setMode:

– (void)**setMode:**(int)*mode*

Sets the mode of the NSColorPanel if *mode* is one of the modes allowed by the color mask. The color mask is set when you first create the shared instance of NSColorPanel for an application. *mode* may be one of these symbolic constants described in the class description (and declared in the header file **AppKit/NSColorPanel.h**):

- NSGrayModeColorPanel
- NSRGBModeColorPanel
- NSCMYKModeColorPanel

- NSHSBModeColorPanel
- NSCustomPaletteModeColorPanel
- NSColorListModeColorPanel
- NSWheelModeColorPanel

See also: **+ setPickerMode**, **– mode**

setShowsAlpha:

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

Tells the NSColorPanel whether or not to show alpha values and an opacity slider.

See also: – **alpha**, – **showsAlpha:**

setTarget:

– (void)**setTarget:**(id)*anObject*

Sets the target of the NSColorPanel to *anObject*. The NSColorPanel's target is the object to which the action message is sent when the user selects a color.

See also: **– setAction:**, **– setContinuous:**

showsAlpha

– (BOOL)**showsAlpha**

Returns whether or not the NSColorPanel shows alpha values and an opacity slider.

See also: **– alpha**, **– setShowsAlpha:**

Notifications

NSColorPanelColorChangedNotification

Notification Object	The NSColorPanel

Posted when the NSColorPanel's color is set, as when **setColor:** is invoked.