NSColorPickingDefault

Adopted By: NSColorPicker

Declared In: AppKit/NSColorPicking.h

Protocol Description

The NSColorPickingDefault protocol, together with the NSColorPickingCustom protocol, provides an interface for adding color pickers—custom user interfaces for color selection—to an application's NSColorPanel. The NSColorPickingDefault protocol provides basic behavior for a color picker. The NSColorPickingCustom protocol provides implementation-specific behavior.

The NSColorPicker class implements the NSColorPickingDefault protocol. The simplest way to implement your own color picker is to create a subclass of NSColorPicker, implementing the NSColorPickingCustom protocol for that subclass. However, it's possible to create a subclass of another class, such as NSView, and use it as a base upon which to add the methods of both NSColorPickingDefault and NSColorPickingCustom.

Color Picker Bundles

A class that implements the NSColorPickingDefault and NSColorPickingCustom protocols needs to be compiled and linked in an application's object file. However, your application need not explicitly create an instance of this class. Instead, your application's file package should include a directory named **ColorPickers**; within this directory you should place a directory *MyPickerClass***.bundle** for each custom color picker your application implements. This bundle should contain all resources required for your color picker: nib files, TIFF files, and so on.

NSColorPanel will allocate and initialize an instance of each class for which a bundle is found in the **ColorPickers** directory. The class name is assumed to be the bundle directory name minus the **.bundle** extension.

Color Picker Buttons

NSColorPanel lets the user select a color picker from an NSMatrix of NSButtonCells. This protocol includes methods for providing and manipulating the image that gets displayed on the button.

Color Mask and Color Modes

The color mask determines which color mode is 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. The predefined color mask constants are:

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

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 NSPopUpLists for loading and saving files. Finally, color wheel mode provides a simplified control for selecting colors.

These constants are defined in **AppKit/NSColorPanel.h**.

See also: NSColorPickingCustom, NSColorPicker (class), NSColorPanel (class)

Method Types

Initializing a Color Picker	– initWithPickerMask:colorPanel:
Setting the Mode	– setMode:
Using Color Lists	– attachColorList: – detachColorList:

Adding Button Images	– insertNewButtonImage:in: – provideNewButtonImage
Showing Opacity Controls	– alphaControlAddedOrRemoved:
Responding to a Resized View	– viewSizeChanged:

Instance Methods

alphaControlAddedOrRemoved:

– (void)**alphaControlAddedOrRemoved:**(id)*sender*

Sent by the color panel when the opacity controls have been hidden or displayed. Invoked automatically when the NSColorPanel's opacity slider is added or removed; you never invoke this method directly.

If the color picker has its own opacity controls, it should hide or display them, depending on whether the sender's **showsAlpha** method returns NO or YES.

attachColorList:

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

Tells the color picker to attach the given *colorList*, if it isn't already displaying the list. You never invoke this method; it's invoked automatically by the NSColorPanel when its **attachColorList:** method is invoked. Since NSColorPanel's list mode manages NSColorLists, this method need only be implemented by a custom color picker that manages NSColorLists itself. This method ordinarily doesn't do anything, since NSColorPanel's list mode manages NSColorLists.

See also: – detachColorList

detachColorList:

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

Tells the color picker to detach the given *colorList*, unless the receiver isn't displaying the list. You never invoke this method; it's invoked automatically by the NSColorPanel when its **detachColorList:** method is invoked. Since NSColorPanel's list mode manages NSColorLists, this method need only be implemented by a custom color picker that manages NSColorLists itself. This method ordinarily doesn't do anything, since NSColorPanel's list mode manages NSColorLists.

See also: – attachColorList

initWithPickerMask:colorPanel:

– (id)**initWithPickerMask:**(int)*mask*
colorPanel:(NSColorPanel *)*owningColorPanel*

Notifies the color picker of the color panel's mask and initializes the color picker. This method is sent by the NSColorPanel to all implementors of the color picking protocols when the application's color panel is first initialized. In order for your color picker to receive this message, it must have a bundle in your application's "ColorPickers" directory (described in "Color Picker Bundles" in the Protocol Description).

mask is determined by the argument to the NSColorPanel method **setPickerMask:**. If no mask has been set, *mask* is NSColorPanelAllModesMask. If your color picker supports any additional modes, you should invoke the **setPickerMask:** method when your application initializes to notify the NSColorPanel class. The standard mask constants are:

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

This method should examine the mask and determine whether it supports any of the modes included there. You may also check the value in *mask* to enable or disable any subpickers or optional controls implemented by your color picker. Your color picker may also retain *owningColorPanel* in an instance variable for future communication with the color panel.

This method is provided to initialize your color picker; however, much of a color picker's initialization may be done lazily through the NSColorPickingCustom protocol's **provideNewView:** method. If your color picker responds to any of the modes represented in *mask*, it should perform its initialization and return **self**. Color pickers that do so have their buttons inserted in the color panel and continue to receive messages from the panel as the user manipulates it. If the color picker doesn't respond to any of the modes represented in mask, it should do nothing and return **nil**.

See also: **+ setPickerMask:** (NSColorPanel class)

insertNewButtonImage:in:

– (void)**insertNewButtonImage:**(NSImage *)*newButtonImage*
in:(NSButtonCell *)*buttonCell*

Sets *newButtonImage* as *buttonCell*'s image. *buttonCell* is the NSButtonCell object that lets the user choose the picker from the color panel—the color picker's representation in the NSColorPanel's picker NSMatrix.

This method should perform application-specific manipulation of the image before it's inserted and displayed by the button cell.

See also: – provideNewButtonImage

provideNewButtonImage

– (NSImage *)**provideNewButtonImage**

Returns the image for the mode button that the user uses to select this picker in the color panel, that is, the color picker's representation in the NSColorPanel's picker NSMatrix. (This is the same image that the color panel uses as an argument when sending the **insertNewButtonImage:in:** message.)

setMode:

– (void)**setMode:**(int)*mode*

Sets the color picker's mode. This method is invoked by NSColorPanel's **setMode:** method to ensure that the color picker reflects the current mode. For example, invoke this method during color picker initialization to ensure that all color pickers are restored to the mode the user left them in the last time an NSColorPanel was used.

Most color pickers have only one mode, and thus don't need to do any work in this method. An example of a color picker that uses this method is the slider picker, which can choose from one of several submodes depending on the value of *mode*. The available modes are:

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

viewSizeChanged:

– (void)**viewSizeChanged:**(id)*sender*

Tells the color picker when the NSColorPanel's view size changes in a way that might affect the color picker. *sender* is the NSColorPanel that contains the color picker. Use this method to perform special preparation when resizing the color picker's view. Since this method is invoked only as appropriate, it's

better to implement this method than to override the method **superviewSizeChanged:** for the NSView in which the color picker's user interface is contained.

See also: **– provideNewView:** (NSColorPickingCustom protocol)