MiscColor

Inherits From: Object

Conforms To: NXTransport

Declared In: misckit/MiscColor.h

Class Description

This documentation is not yet completed; many of the methods are not yet detailed. Sorry.

The MiscColor class is primarily an object oriented wrapper around the NeXT-supplied functions which deal with color and the NXColor structure. It does, however, add a few useful twists. You create a MiscColor using any of the +**newColor:**, +**newFromPasteboard:**, or +**alloc** methods. If you use +**alloc**, then you have the choice of initializing the MiscColor with any of the ± **init**, ± **initColor:**, ± **initFromPasteBoard:**, or ±**initColorFromPixel:** methods. You can check to see if there is a color on a particular pasteboard with the +**canInitFromPasteboard:** method.

To change the color of a MiscColor object, use ±**setColor:**, ±**setColorFromPixel:**, or ±**setColorFromObject:**. The MiscColor object considers itself to represent a ªbaseº color, set by the above methods, which may actually be rendered differently depending upon environmental conditions. There are two mechanisms by which the environment can alter a MiscColor. First, a *delegate* object can alter a MiscColor before it is sent to the window server. This allows the delegate to alter the color to handle things such as spot or process color separations. (For example, when doing a spot color

separation of the color green, the delegate can turn all greens into shades of gray and all other colors to white.) Once the delegate has altered the color, the MiscColor instance runs the color through its own filter function, implemented in the ±**filterColor:** method, to further modify the color. This allows, for example, the MiscHalftoneColor class to track a base color and a halftone value, halftoning the color returned by the delegate in an appropriate manner. Because of this system, the base color stored in the MiscColor is accessed directly via the ±**actualColor** method while the true color it represents (the base color run through the filter) is returned by the ±**color** method. In other words, there is an internal representation which is different from the default displayed/printed color, converted from the internal representation to an actual Postscript color by the ±**filterColor:** method. All this confusion allows the MiscColor to handle color separations in a flexible and elegant manner. By not implementing the ±**filterColor:** method and not connecting up a delegate, a developer can disable both mechanisms and ignore them completely, if the application does not require color separations. (Note that by designing an application with the MiscColor, color separations may be easily added at a later date with much less effort expended.)

The delegate object can only affect a MiscColor when it is about to be sent to the window server, via the ±**setColor** method. The ±**setColor** method sends the ±**willSendColorToPS:shouldPrint:** message before sending anything to the server. The delegate can return a new color, which supercedes the base color and may or may not be derived from the base color, or the delegate can return the color sent to it unchanged. This allows the delegate to make whatever changes are necessary to support color separations. After the modified and filtered color has been sent to the window server, the actual color that was sent to the window server is forwarded to the delegate via the ±**didSendColorToPS:** method. The delegate may be accessed via the ±**delegate** and ±**setDelegate:** methods.

MiscColors can be passed around via the Pasteboard class, archiving to a typed stream, or **bycopy** in distributed objects. On the pasteboard, MiscColors currently place an NXColor, and not the whole object. In future implementations, both MiscColors and their subclasses will be able to place various archived, richer, versions of themselves on a Pasteboard for more flexibility. (This will also facilitate communication between applications which use the MiscColor.) Pasteboard support is provided via the ±**readFromPasteBoard:**, ±**writeToPasteBoard:**, and +**newFromPasteboard:** methods. Use +**canInitFromPasteboard:** to see if the Pasteboard contains a color representation which can be converted into a MiscColor.

There are three different methods which may be used to compare MiscColors: ±**isEqualColor:**, ±**isEqualToColor:**, and ±**isEqual:**. The ±**isEqualColor:** method returns YES when the object passed to it has the same base color as the receiver. (It first checks ±**actualColor**, if it can. If that method isn't implemented by the passed object, then it checks ±**color** against the filtered base color.) The ±**isEqualToColor:** method returns YES when an NXColor is the same as the filtered base color, and the ±**isEqual:** method returns YES only when the passed object is both the same class as the receiver and would

return YES for ±**isEqualColor:**.

There is a plethora of NX*name*() functions which access color components (red, green, blue, hue, saturation, brightness, cyan, magenta, yellow, black, gray, and alpha). The MiscColor class has counterparts to all of them which follow similar naming conventions. All these methods operate upon the base color and *not* the filtered color of the MiscColor. First, you can obtain the value of any color component via the appropriate ±***name*Component** method, where ***name*** is the name of a color component. You can get all the components of a color space at once via ±**getRed:green:blue:alpha:**, ±**getCyan:magenta:yellow:black:alpha:**, ±**getHue:saturation:brightness:alpha:**, ±**getGray:alpha:**, or the couterparts without the **alpha:** on the end. By changing **get** to **set** you can convert from a colorspace to a MiscColor. You can also set a single component via the ±**set*Name*Component** series of methods.

There is rudimentary support of NeXT's ColorList object in the MiscColor object. The name of the color list from which the base color stored in the MiscColor object was obtained is available through the ±**colorListName** method, and the name in that list is returned by ±**colorName**. These methods are really only useful for colors with persistent names, such as PANTONE^{TM} colors. You can create a new MiscColor instance from a named color in a ColorList via the +**findColorNamed:inList:** method.

Instance Variables

NXColor **theColor**;
id **delegate**;

theColor	The color stored in a MiscColor.
delegate	The MiscColor's delegate.

Method Types

Initializing and freeing a MiscColor	+ newColor: ± init ± initColor: ± initFromPasteBoard: ± initColorFromPixel:
Changing color components	± setRedComponent: ± setGreenComponent: ± setBlueComponent: ± setCyanComponent: ± setMagentaComponent: ± setYellowComponent: ± setBlackComponent: ± setHueComponent: ± setSaturationComponent: ± setBrightnessComponent: ± setGrayComponent: ± setAlphaComponent: ± setGray: ± setGray:alpha: ± setRed:green:blue: ± setRed:green:blue:alpha: ± setHue:saturation:brightness: ± setHue:saturation:brightness:alpha: ± setCyan:magenta:yellow:black: ± setCyan:magenta:yellow:black:alpha:
Changing the color	± setColor: ± setColorFromObject: ± setColorFromPixel:
Comparing colors	± isEqualColor:

	± isEqualToColor: ± isEqual:
Delegate	± delegate ± setDelegate:
NXColorList support	+ findColorNamed:inList: ± colorName ± colorListName
Obtaining color	± color ± actualColor ± filterColor:
Obtaining color components	± redComponent ± greenComponent ± blueComponent ± cyanComponent ± magentaComponent ± yellowComponent ± blackComponent ± hueComponent ± saturationComponent ± brightnessComponent ± grayComponent ± alphaComponent ± getGray: ± getGray:alpha: ± getRed:green:blue: ± getRed:green:blue:alpha: ± getHue:saturation:brightness: ± getHue:saturation:brightness:alpha: ± getCyan:magenta:yellow:black:

	± getCyan:magenta:yellow:black:alpha:
Pasteboard support	+ canInitFromPasteboard: ± readFromPasteBoard: ± setFromPasteBoard: ± writeToPasteBoard:
Setting current drawing color	± display ± setColor
Archiving	± read: ± write:
Sent to delegate	± willSendColorToPS:shouldPrint: ± didSendColorToPS:

Class Methods

canInitFromPasteboard:
+ (BOOL)**canInitFromPasteboard:**(Pasteboard *)*pasteboard*

Returns YES is *pasteboard* has available an NXColor data type, a MiscColor object, or a subclass of MiscColor. Returns NO otherwise.

See also: **+newFromPasteboard:** and ± **initFromPasteboard:**

findColorNamed:inList:
+ findColorNamed:(const char *)*aName* **inList:**(const char *)*aListName*

A cover for the NXFindColorNamed() function, which searches for a specific color in a list of named colors. Returns a new instance of the class this message was sent to unless the color wasn't found, in which case *nil* is returned.

See also: **-colorName**, **-colorListName**

newColor:
+ newColor:(NXColor)*color*

Returns a new instance of the receiving class initialized to *color.*

See also: **-initColor:**

newFromPasteBoard:
+ newFromPasteBoard:(Pasteboard *)*pasteboard*

Returns a new MiscColor instance initialized from pasteboard. If there is not a color on *pasteboard*, as determined by the **+canInitFromPasteboard:** method, then nil is returned.

See also: **+canInitFromPasteboard:** and ± **initFromPasteboard:**

Instance Methods

delegate
- delegate

Returns the delegate of the receiving MiscColor instance.

See also: **-setDelegate:**, **-didSendColorToPS:** (delegate), and **-willSendColorToPS:shouldPrint:** (delegate)

init

- init

Initializes a new instance of MiscColor and set its color to white. Returns *self*.

See also: **-initColor:**, ±**initColorFromPixel:**, and ±**initFromPasteboard:**

initColor:
- initColor:(NXColor)*color*

The designated initializer for the MiscColor class. This method initializes a new MiscColor instance and sets it to be the color passed in *color.* Returns *self*.

See also: **-init**, ±**initColorFromPixel:**, and ±**initFromPasteboard:**

initColorFromPixel:
- initColorFromPixel:(NXPoint *)*location*

Initializes the receiver to be the same color as the screen pixel at location *location*. The pixel must reside in the currently focused view and *location* is in the focused View's coordinate system. Returns *self*.

See also: **-init**, ±**initColor:**, and ±**initFromPasteboard:**

initFromPasteboard
- initFromPasteboard:(Pasteoard *)*pasteboard*

Attempts to initialize a new instance from *pasteboard*. If a color of some sort is not found on *pasteboard*, then *nil* is returned; otherwise this method returns *self*.

See also: +**canInitFromPasteboard:**, **-init**, ±**initColor:**, and ±**initColorFromPixel:**

read:
- read:(NXTypedStream *)*stream*

Reads the instance variables for the receiving MiscColor from the typed stream *stream*. Returns *self*.

See also: **-write:**

setDelegate
- setDelegate:(x)*x*

Sets the delegate for the receiving MiscColor instance. Returns *self*.

See also: **-delegate**, **-didSendColorToPS:** (delegate), and **-willSendColorToPS:shouldPrint:** (delegate)

write
- write:(NXTypedStream *)*stream*

Writes the instance variables for the receiving MiscColor to the typed stream *stream*. Returns *self*.

See also: **-read:**

x
- x:(x)*x*

x. Returns *self*.

See also: **-x:**

x
- x:(x)*x*

x. Returns *self*.

See also: **-x:**

Methods to be Implemented by the Delegate

didSendColorToPS:
- didSendColorToPS:(NXColor)*color*

If a color ends up being sent to the Postscript server, then it is also forwarded to the delegate, which may then perform any additional tasks as necessary. The color sent to the Postscript interpreter is passed in the *color* argument. This method should return *self*.

See also: **-willSendColorToPS:shouldPrint:** (delegate)

willSendColorToPS:shouldPrint:
- (NXColor)**willSendColorToPS:**(NXColor)*color* **shouldPrint:**(BOOL *)*printOK*

This method allows a delegate to alter a color before it is sent to the Postscript interpreter. The color to be sent is passed to the delegate in *color* and the delegate should return that color or a new, filtered color. The *printOK* boolean is set to YES before it is passed to the delegate. If the delegate decides that this color should not be printed, then it can set this to NO and the MiscColor will not send the color. A properly written delegate can use this method to generate print separations; both spot color and process color are possilbilities.

See also: ±**filterColor:**, ±**setColor**, and **-didSendColorToPS:** (delegate)

- writeToPasteBoard:(Pasteboard *)pasteboard;
- readFromPasteBoard:(Pasteboard *)pasteboard;
- setFromPasteBoard:(Pasteboard *)pasteboard;
- setRedComponent:(float)red;
- setGreenComponent:(float)green;
- setBlueComponent:(float)blue;
- setCyanComponent:(float)cyan;
- setMagentaComponent:(float)magenta;
- setYellowComponent:(float)yellow;
- setBlackComponent:(float)black;
- setHueComponent:(float)hue;
- setSaturationComponent:(float)saturation;
- setBrightnessComponent:(float)brightness;
- setGrayComponent:(float)gray;
- setAlphaComponent:(float)alpha;
- setColor:(NXColor)color;
- setColorFromObject:object;
- setColorFromPixel:(NXPoint *)location;
- (NXColor)color;
- (NXColor)actualColor;
- (NXColor)filterColor:(NXColor)input;
- display;
- (BOOL)setColor;
- getGray:(float *)gray;
- getGray:(float *)gray alpha:(float *)alpha;
- getRed:(float *)red green:(float *)green blue:(float *)blue;
- getRed:(float *)red green:(float *)green blue:(float *)blue alpha:(float *)alpha;
- getHue:(float *)hue saturation:(float *)saturation brightness:(float *)brightness;
- getHue:(float *)hue saturation:(float *)saturation brightness:(float *)brightness alpha:(float *)alpha;
- getCyan:(float *)cyan magenta:(float *)magenta yellow:(float *)yellow black:(float *)black;
- getCyan:(float *)cyan magenta:(float *)magenta yellow:(float *)yellow black:(float *)black alpha:(float *)alpha;
- setGray:(float)gray;
- setGray:(float)gray alpha:(float)alpha;
- setRed:(float)red green:(float)green blue:(float)blue;
- setRed:(float)red green:(float)green blue:(float)blue alpha:(float)alpha;
- setHue:(float)hue saturation:(float)saturation brightness:(float)brightness;
- setHue:(float)hue saturation:(float)saturation brightness:(float)brightness alpha:(float)alpha;

- setCyan:(float)cyan magenta:(float)magenta yellow:(float)yellow black:(float)black;
- setCyan:(float)cyan magenta:(float)magenta yellow:(float)yellow black:(float)black alpha:(float)alpha;
- (BOOL)isEqualColor:anObject;
- (BOOL)isEqualToColor:(NXColor)color;
- (BOOL)isEqual:anObject;
- (float)redComponent;
- (float)greenComponent;
- (float)blueComponent;
- (float)cyanComponent;
- (float)magentaComponent;
- (float)yellowComponent;
- (float)blackComponent;
- (float)hueComponent;
- (float)saturationComponent;
- (float)brightnessComponent;
- (float)grayComponent;
- (float)alphaComponent;
- (const char *)colorName;
- (const char *)colorListName;

x
- x:(x)*x*

x. Returns *self*.

See also: **-x:**

x
- x:(x)*x*

x. Returns *self*.

See also: **-x:**