Technote 1100

Color Picker 2.1

CONTENTS

Color Picker 2.1 Overview

New Color Pickers

- CMYK
- Crayon
- HSV
- HTML

Determining If a Picker is Installed

Color Picker 2.1 APIs

- New API
- Maintained APIs
- Obsolete APIs

Eye Dropper Tool

- Overview
- Instructions and Warnings

New Component Manager Messages

Bugs Fixed

References

With the release of Mac OS 8.0, Apple has redesigned and expanded the Color Picker. In doing so, we have changed the Color Picker APIs for developers and expanded on the number and kind of pickers that are available to the end user. This Technote outlines these changes in detail.

This Note is primarily intended for developers who use or are interested in using the Color Picker Manager APIs.

Note:
This Technote is to be used in conjunction with the 'Color Picker Manager' chapter in Apple's technical reference book: *Advanced Color Imaging on the MacOS* .

You should also be familiar with the 'Dialog Manager' chapter of *Inside Macintosh:Macintosh Toolbox Essentials* and the 'Component Manager' chapter of *Inside Macintosh:More Macintosh Toolbox* .

Color Picker 2.1 Overview

The Color Picker Manager supplies your application with functions and a standard user interface for asking the user for color choices. Your application can use these APIs to interact with the Color Picker Extension and also to create your own custom color picker.

System Requirements

- Color Picker 2.1 will run on System 7.5 through MacOS 8.0.

Dependencies

- QuickTime Component Manager: Color Picker 2.1 uses the QuickTime Component Manager, however, if QuickTime is not installed, the Color Picker Manager loads its own Component Manager code.
- ColorSync: the Color Picker Manager uses ColorSync if installed (not required). Color Picker 2.1 adds support for ColorSync 2.x profiles.

Licensing

- Color Picker 2.1 is available for licensing from Apple's Software Licensing group as part of the ColorSync addendum. You need to license the Color Picker to ship it with your product. For more information on licensing, please contact Software Licensing at 512-919-2645 or sw.license@apple.com.

Download

- Color Picker 2.1 is available for download from Apple's SDK site.

Gestalt

- By checking `Gestalt()` you can determine if Color Picker 2.1 is present by using the `gestaltColorPickerVersion` selector. If `Gestalt()` returns an error, then only the 1.0 ROM-based Color Picker is there. If it returns 0x00000100 then you have Color Picker 2.0 (weird, but true). If it returns 0x00000210 then you have 2.1 available.

New Features

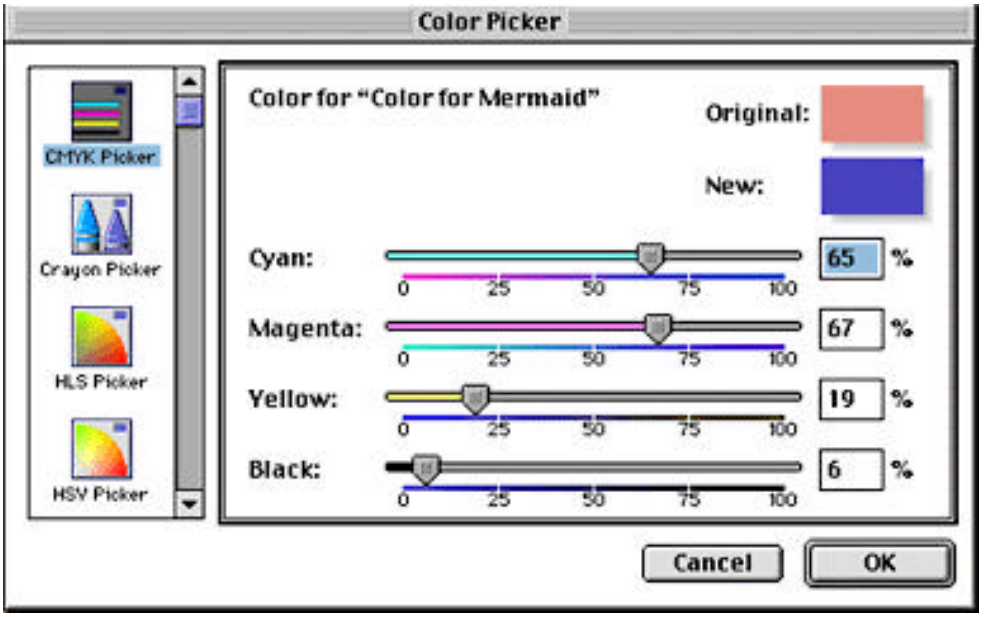
- New system pickers: CMYK Picker, Crayon Picker, HSV Picker, HTML Picker, See the 'New Color Pickers' section for more details.
- Color Sync 2.x profile support.
- Eye Dropper Tool. See the 'Eye Dropper Tool' section for more information.
- Smaller API. See the 'Color Picker 2.1 APIs' section for an overview.
- Native pickers on Power PC.

New Color Pickers

There are many new pickers with the 2.1 release: CMYK Picker, Crayon Picker, HSV Picker and an HTML Picker. This section overviews these new pickers and shows you what they look like.

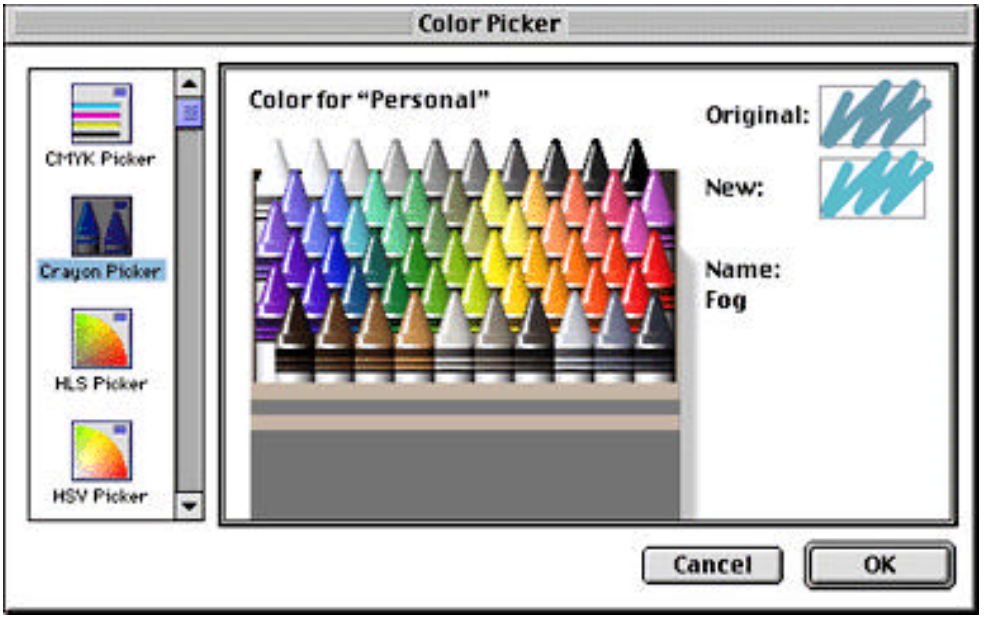
CMYK

This is a slider-based picker for selecting a color in CMYK color space. This picker is mainly intended for use in the color pre-press industry since a majority of printing devices out there use this color space.



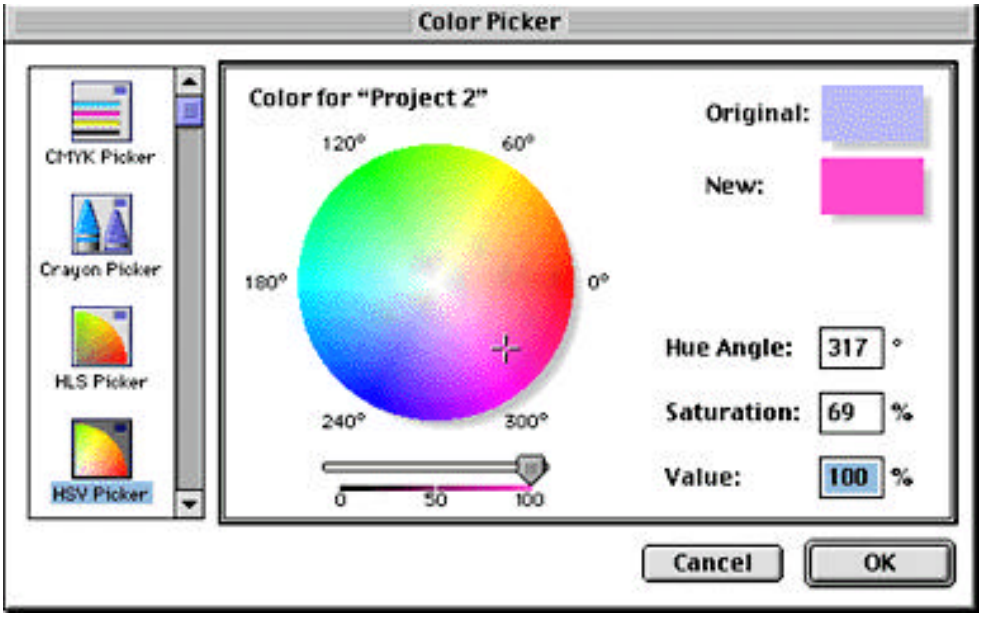
Crayon

The Crayon picker is targeted at the home and school market. It presents a box of 60 crayon colors from which the user can select.



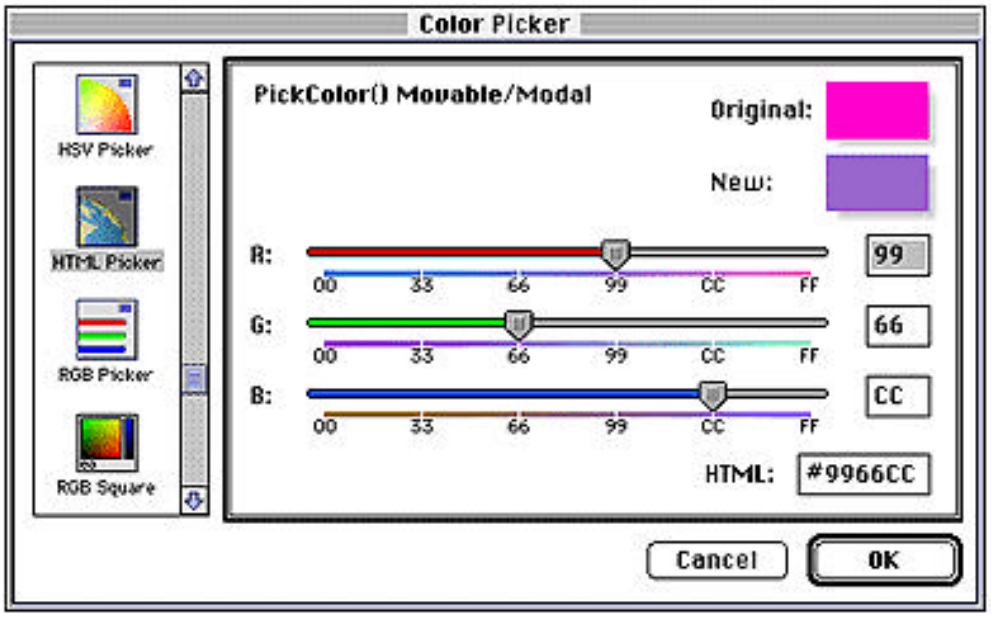
HSV

The original Color Picker 1.0 (the dartboard as it's affectionately called at Apple) was an HSV picker (although it called itself a Hue, Saturation, and "Brightness" picker). This picker was removed with the 2.0 release, however many users liked it and asked for its return. Therefore, for users who became comfortable working in that color space, we have added it back into Color Picker 2.1. It has the "wheel and slider" interface much like the original 1.0 picker.



HTML

This is basically another RGB slider-based color picker. The units though are in hexadecimal bytes and a combined HTML tag is maintained as a fourth text-edit field (so if the sliders read 99 for red, 66 for green and CC for blue, the combined HTML tag text field will contain #9966CC). Many HTML commands take this kind of composite "hash-mark-with-RGB-hex" format.



Determining If a Picker Is Installed

As a developer, you may wish to know whether a particular picker is installed on your user's machine. To accomplish this you need to complete two steps:

1. Check the Gestalt selector `gestaltColorPickerVersion`.

```
theErr = Gestalt(gestaltColorPickerVersion, &value);
```

2. Then use the Component Manager to query for a specific type of picker. All Apple color pickers use

a manufacturer type of 'appl', a type of 'cpkr' and a sub-type that reflects the color space of the picker (as defined in ColorSync's `CMApplication.h` header file).

The subtypes for the various color pickers are:

CMYK Picker = `cmCMYKData` = 'CMYK'
Crayon Picker =*none* = 'name' (for "named color space" like Pantone™ is a named color space).
HLS Picker = `cmHLSData` = 'HLS'
HSV Picker = `cmHSVData` = 'HSV'
HTML Picker = *none* = 'HTML'
RGB Picker = `cmRGBData` = 'RGB '

As an example, to see if Apple's CMYK picker exists on your user's system, you would set up your `ComponentDescription` structure like this:

```
theComponentDesc.componentType = 'cpkr';
theComponentDesc.componentSubType = cmCMYKData;
theComponentDesc.componentManufacturer = 'appl';
```

NOTE: Third party pickers and future color pickers may use some of the same subtypes or new ones. Therefore, you could potentially get back more than one picker during your Component Manager query. See the 'Component Manager' chapter of *Inside Macintosh: More Macintosh Toolbox* for further details.

Color Picker 2.1 APIs

To download new header files click here

In Apple's studies, we realized that a majority of the Color Picker APIs were not being used by developers. Therefore, with the Color Picker 2.1 release, a lot of the APIs have become obsolete. This section covers new, old, supported and unsupported/removed APIs. If you have a need for one of the obsolete APIs, please let Developer Support know at feedback and we will attempt to get the API reinstated or give you another way to implement the same functionality.

New API

Here is the one new Color Picker 2.1 call (ColorSync 2 savvy):

```
pascal OSErr NPickColor(NColorPickerInfo *theColorInfo);
```

`theColorInfo`:

A color picker parameter block, which is described on page 2-21 of the *Advanced Color Imaging Reference* . With Color Picker 2.1, however, where profile handles were used with Color Picker 2.0, profile references are nowused for this API.

DESCRIPTION

This call is identical to `PickColor()`, but it replaces the older ColorSync 1.0 data types with new ColorSync 2.x profile references. The color picker parameter block is described on page 2-21 of the *Advanced Color Imaging Reference* . When filling out the parameter block for a call to `NPickColor`, replace all profile handles with profile references. The optional color-changed proc you has also changed; a new data structure `NCMColor` replaces the `CMColor` data type and uses profile references. See the "New Component Manager Messages" section for more information on the color-changed proc.

Maintained APIs

Here are the original Color Picker 1.0 calls that we are keeping and maintaining:

```
pascal SmallFract Fix2SmallFract(Fixed f);

pascal Fixed SmallFract2Fix(SmallFract s);

pascal void CMY2RGB(const CMYColor *cColor, RGBColor *rColor);

pascal void RGB2CMY(const RGBColor *rColor, CMYColor *cColor);

pascal void HSL2RGB(const HSLColor *hColor, RGBColor *rColor);

pascal void RGB2HSL(const RGBColor *rColor, HSLColor *hColor);

pascal void HSV2RGB(const HSVColor *hColor, RGBColor *rColor);

pascal void RGB2HSV(const RGBColor *rColor, HSVColor *hColor);

pascal Boolean GetColor(Point where, ConstStr255Param prompt, const RGBColor *inColor,
     RGBColor *outColor);
```

Here is the one Color Picker 2.0 call we are keeping and maintaining:

```
pascal OSErr PickColor(ColorPickerInfo *theColorInfo);
```

NOTE: Please see the '*Advanced Color Imaging* ' book for complete details on these APIs.

Obsolete APIs

Here are the Color Picker 2.0 and 2.1 calls we have removed:

```
AddPickerToDialog
CreateColorDialog
CreatePickerDialog
DisposeColorPicker
GetPickerVisibility
SetPickerVisibility
SetPickerPrompt
DoPickerEvent
DoPickerEdit
DoPickerDraw
GetPickerColor
SetPickerColor
GetPickerOrigin
SetPickerOrigin
GetPickerProfile
SetPickerProfile
GetPickerEditMenuState
SetPickerColorChangedProc
NCreateColorDialog
NGetPickerColor
NSetPickerColor
NGetPickerProfile
NSetPickerProfile
NSetPickerColorChangedProc
ExtractPickerHelpItem
```

If you have a need for one of these APIs, please contact feedback.

Eye Dropper Tool

For a picker developer, the eye dropper tool is probably the single-most significant addition to Color

Picker 2.1. If the user holds down the Option key, the cursor will change to an eyedropper and the user will "pick up" the color under the cursor. This section describes how this feature is implemented and how your application can take advantage of it.

Overview

The way this tool was created is rather straightforward, but is a little tedious to explain. Recall that beginning with Color Picker 2.0, color pickers have been component based (that is, using the Component Manager). Individual color pickers are loaded and instantiated as the user browses through them in the Color Picker dialog. The way the Color Picker Manager interacts with the individual color picker components is through Component Manager-based messages. The Color Picker has defined many messages for things such as passing events to the picker, telling the color picker to redraw itself, telling the color picker to set the "new" color, etc. In terms of the eyedropper tool, two messages are particularly important: the `PickerEvent` message and the `PickerSetColor` message.

In Color Picker 2.0, the individual pickers were sent only two `PickerSetColor` messages after instantiating the picker. These were to tell the picker how to set both the original color and new color boxes. The original color was the color originally passed in when the application first called upon the color picker, the new (or user) color was the current color the user had selected. After a color picker had received these messages (and others as well) and displayed itself, the `PickerEvent`message came to the forefront. In fact, until the user clicked on another color picker or hit the okay or cancel button, a color picker basically remained in an event loop, receiving events from the Color Picker Manager via the `PickerEvent` message. Even null events were passed along to the color pickers (this was an ideal time for a color picker to update the cursor, etc.).

In Color Picker 2.1, things have changed a bit. First of all, you can no longer be assured that your picker will receive only the initial two `PickerSetColor` messages. When a user holds down the option key (invoking the eye-dropper tool) and clicks, the Color Picker Manager now suppresses that click from the current color picker and instead sends it a `PickerSetColor` message. The color picker sends this message with the user color flag set thus, the original color will never change. For a color picker to be savvy with 2.1, it needs to be able to handle the odd `PickerSetColor` message and handle the new color passed it. This includes updating the user color box in the upper right corner of the picker to display the new color, updating whatever sliders and text edit fields the color picker has, etc. Try it out, all the Apple color pickers do this correctly.

Instructions and Warnings

1. The Color Picker Manager may change the cursor on you (to the eyedropper cursor). It is the picker's responsibility to change it back. If your color picker is handling cursor updates during null events, you need do nothing more. One third party picker was doing no cursor updating at all this would leave the cursor as an eyedropper even though the user had let up on the option key. A well-written color picker will always handle cursor updating for example, when the user moves the cursor over a text edit field the cursor should turn into an I-beam cursor, etc.

2. Since color pickers should be handling cursor updating during null events, all null events are suspended while the option key is held down. If this were not the case, there would be a sort of tug-of-war between the Color Picker Manager and the individual color pickers the Color Picker Manager would be fighting to set the cursor to an eyedropper while the picker was trying to set it to an arrow or whatever.

3. Finally, recall also that the Color Picker can be called moveable modal (using the more mid-level API calls like `PickColor()`) as well as standard modal (using the higher-level API call `GetColor()`). There is an interesting side-effect then for these two cases. If the picker is called moveable-modal (think fast now) what would be the effect of option-clicking, say, in the Desktop while the picker is up? Well, the Event Manager snuffs the option-click and instead switches the user out of the current application, hides the current application, and switches the user into the Finder. In fact, option-clicking in any other process's windows will cause a similar behavior. Rather than breaking that "user experience", we preserve it. So, the only time that the user actually gets an eyedropper cursor is when 1) the option key is down and 2) the cursor is over one of the current process's windows! By and large, this is as you

would want it. If you were writing a paint program and wanted the user to be able to pick up a color using the Color Picker from their document, this document window would of course be a window owned by your application. Also, if the Color Picker were called moveable-modal, the user would of course be able to move the Color Picker's dialog around on the screen so as to get at that specific color that they want from their document.

The strictly modal case is quite another situation. Since it is a non-moveable modal dialog, there never was any context switching behavior to preserve. In fact, the only thing the user would expect from clicking in the Desktop from a modal dialog is a `SysBeep()` sound (indicating, "Don't do that!". Well, Color Picker 2.1 preserves the `SysBeep()` scenario except when the option key is down. When the option key is down, the user can click anywhere (except the menu bar) and pick up a color even from the Desktop or another process' window. The down side of course, is that the dialog is not moveable. So if the color the user wanted to pick up was underneath the Color Picker's dialog too bad.

By the way, the Color Picker handles all the hit-testing etc. by building the hit-test region so that when the cursor is within this region and the option key is down, they get an eyedropper; otherwise they don't. For the moveable-modal case, we walk the current layers window list accumulating window regions. In the strictly modal case, we use `GetGrayRgn()`.

Finally, make sure you update the cursor during null events passed your picker and also be able to handle `PickerSetColor` messages at any time. The eyedropper tool is otherwise free working across all color pickers and requires no additional work on your part; (most third-party color pickers work perfectly without modification).

New Component Manager Messages

For new header files click here

If you wish to write a Color Picker component or if you wish to revise a Color Picker component you have already written, there are some new Color Picker messages you may want to implement.

`PickerSetColorChangedProc`

```
pascal ComponentResult PickerSetColorChangedProc(long storage, ColorChangedUPP
colorProc, long colorProcData)
```

This message is sent once to your picker, after you are called to initialize yourself. It contains a procPtr to the calling applications `colorChangedProc`. This allows you, the picker, to call the applications `colorChangedProc` directly. This was found useful during times when the picker is stealing all events - for example, during a drag of a slider when the user has the mouse button continuously down. When the user releases the mouse button, control returns to the Color Picker Manager and the `colorChangedProc` is called at that time. But you may, in your picker, want to call the `colorChangedProc` yourself while the user is "live dragging".

Continually calling the `colorChangedProc` can slow you way down however, depending on how processor intensive the `colorChangedProc` is. Therefore, the new Color Picker defines a new flag in the flags field passed in by the calling application: `kColorPickerCallColorProcLive`. Your picker is passed a copy of the flags when the Color Picker Manager sends your picker a PickerInit message. Therefore, it is recommended that if you support the `PickerSetColorChangedProc` message you should:

1) Call it only during mouse drags when the Color Picker Manager is unable to get processor time to call the `colorChangedProc`.

2) Call it only if the `kColorPickerCallColorProcLive` flag is set.

`NPickerGetColor,`
`NPickerSetColor,`
`NPickerGetProfile,`

`NPickerSetProfile`, and `NPickerSetColorChangedProc`

```
pascal ComponentResult NPickerGetColor(long storage, ColorType whichColor,
NPMColor *color
pascal ComponentResult NPickerSetColor(long storage, ColorType whichColor,
NPMColor *color)
pascal ComponentResult NPickerGetProfile(long storage, CMProfileRef *profile)
pascal ComponentResult NPickerSetProfile(long storage, CMProfileRef profile)
pascal ComponentResult NPickerSetColorChangedProc(long storage, NColorChangedUPP
colorProc, long colorProcData)
```

These messages are simply parallel to the original Color Picker 2.0 messages. They however pass profile references where their original Color Picker 2.0 counterparts passed in profile handles. This reflects the ColorSync 2.x support. Please see the '*Advanced Color Imaging* ' book for more information on the 2.0 messages.

Color Picker Component Manager Message Warnings

Recall that a component has only a single "main" entry point and that the Component Manger sends your component messages via this entry point. Typically, you would have something like a big case statement in your "main" function. Based upon the message, you would either call out to your function or return badComponentSelector (0x80008002) if you receive a message value that you don't understand. Please see *Inside Macintosh: More Macintosh Toolbox* for more information on the Component Manager and badComponentSelector.

If you choose not to implement all the new Color Picker messages, you must return badComponentSelector for all the messages that you do not support. This allows the Color Picker Manager to handle the message in some other fashion.

Also, please note that there is a standard Component Manager message '`kComponentCanDoSelect`'. This also is an important place to respond to the Color Picker Manager that, "Yes, I know what how to handle that message," or "No, I don't.". It is important to do this so that the Color Picker does not try to work around your application.

Further References

- *Advanced Color Imaging on the MacOS*
- *Inside Macintosh:Macintosh Toolbox Essentials*
- *Inside Macintosh:More Macintosh Toolbox*
- To download the Color Picker 2.1 SDK and Sample code click here.

Downloadables

 Acrobat version of this Note (K)

To contact us, please use the Contact Us page.
Updated: 9-February-98