T E C H N O T E :
Palette Manager Tidbits

By Guillermo Ortiz
Revised by Michael Marinkovich
marink@applelink.apple.com
Apple Developer Technical Support (DTS)

This Technote describes the changes and enhancements, over time, to the Palette Manager. The biggest problem facing developers who work with the Palette Manager is making sure that they set the palette to a color window.

This Technote is of interest to developers designing general-purpose applications who want to use palettes for their windows but who don't want to deal with the hassles of setting up a new palette for each and every window.

Although the Palette Manager is documented in the Palette Manager chapter of *Inside Macintosh*: *Advanced Color Imaging*, this Technote sheds light on issues that may remain unclear after reading that chapter.

Enhancements to the Palette Manager

Several enhancements to the Palette Manager provide you with additional manageability for applications using the Palette Manager.

Application Default Palettes

Your application can define a default palette for the system to use when you need to define the color environment, that is, when your application creates a color window without an associated palette or display a dialog box.

The application palette feature is especially handy in cases where a color application uses old-style dialogs and alerts because without an application palette, the system will use its own default palette, which may not be the appropriate colors, to define the color environment. Since the system uses the default palette, the color environment may change (will change in 16-color mode) and cause some "cosmic" colors to appear in the active window. Defining a default application palette with two colors, black and white, solves this problem.

If the system needs a palette to define a color environment, it looks in the resource fork of the application for the `'pltt' ID = 0` resource and uses the palette contained therein. If the system cannot find this resource in the application's resource fork, it will use its own default palette (resource `'pltt' ID = 0` in the System file) if present, or, if necessary, it will use the Palette Manager's built-in palette.

Once an application has set its color environment (by calling `InitMenus`, or `InitPalettes`) it can find the default palette by calling

```
GetPalette (WindowRef) -1);
```

or change the default palette by calling

```
SetPalette ((WindowRef) -1, srcPalette, true);
```

Note
The initialization of the Palette Manager with a call to `InitMenus` is contrary to the way *Inside Macintosh*: *Advanced Color Imaging,* "The Palette Manager" documents it. ◆

One Palette, Many Ports

You can associate one palette with many `CGrafPort` and `CWindow` records, thus simplifying the use of a single palette with multiple ports and windows.

Although this ability to associate one palette with multiple ports and windows will allow the use of calls like `PmForeColor` and `PmBackColor`, calling

`ActivatePalette` with an off-screen port will associate the palette with the port but will not cause any change in the color environment.

An important implication of this feature is that `DisposeWindow` (`DisposWindow`) will no longer dispose of the associated palette automatically, since it may be allocated to other ports or windows. The only exception to this behavior is when an application has used `GetNewCWindow` to create the window, there is a `'pltt'` resource with the same ID as the window, and the application has not called `GetPalette` for the window.

Color Updates

`NSetPalette` has the same functionality as `SetPalette`, but the `CUpdates` parameter has been modified from a `Boolean` to an `Integer` as follows:

```
void NSetPalette (WindowPtr dstWindow, PaletteHandle srcPalette,
                  short nCUpdates)
```

`NSetPalette` changes the palette associated with `dstWindow` to `srcPalette`. It also records whether the window will receive updates as a result of a change to its color environment.

You have more flexibility in setting window updates for your application:

- If you want `dstWindow` to be updated whenever its color environment changes, set `nCUpdates` to `pmAllUpdates`.
- If you are only interested in updates when `dstWindow` is the active window, set `nCUpdates` to `pmFgUpdates`.
- If you are only interested in updates when `dstWindow` is not the active window, set `nCUpdates` to `pmBkUpdates`.

```
{ NSetPalette Update Constants }

pmNoUpdates = 0x8000 {no updates}
pmBkUpdates = 0xA000 {background updates only}
pmFgUpdates = 0xC000 {foreground updates only}
pmAllUpdates = 0xE000 {all updates}
```

`SetPalette` retains its syntax and function:

```
void SetPalette (WindowPtr dstWindow, PaletteHandle srcPalette,
                 Boolean CUpdates)
```

Note
The trap words for `NSetPalette` and `SetPalette` *are* identical. ◆

CopyPalette

```
void CopyPalette (PaletteHandle srcPalette, PaletteHandle dstPalette,
                  short srcEntry, short dstEntry, short dstLength);
```

`CopyPalette` is a utility procedure that copies `dstLength` entries from the source palette into the destination palette; the copy begins at `srcEntry` and `dstEntry`, respectively. `CopyPalette` will resize the destination palette when the number of entries after the copy is greater than the original.

`CopyPalette` does not call `ActivatePalette`, so the application is free to do a number of palette changes without causing a series of intermediate changes to the color environment; the application should call `ActivatePalette` after completing all palette changes.

If either of the palette handles are `NIL`, `CopyPalette` does nothing.

Summary

The Palette Manager will accomplish a lot for you with a few simple techniques, such as attaching palettes automatically to your windows. Most developers aren't aware of this. The Palette Manager will also perform automatic updates when your application switches to the foreground.

Further References

- *Inside Macintosh*: *Advanced Color Imaging*, Palette Manager chapter.

Change History

This Technote was originally written in October, 1988.