Animated Colors

Animated colors allow you to reserve device indices for color table animation.

One way to change the color of an object on the screen is to change the pixel values in the object's part of the pixel map-you draw it again in a different color. In certain situations, you can get the same effect at less cost in processing and memory by changing the colors in the video device's color table instead. All pixel values corresponding to the altered indices suddenly appear on the display device in a new color. By careful selection of index values and the corresponding colors, you can achieve a number of special animation effects.

To use an animated color, you must first draw with it using the **PmForeColor** or **PmBackColor** procedure. To create color table animation, you then change that entry's RGB color by using the **AnimateEntry** procedure. You can animate a contiguous set of colors by supplying RGB colors from a color table, using the **AnimatePalette** procedure.

The way the **Palette Manager** reserves indices for animated colors creates some side effects. The **Palette Manager** first checks each animated color to see if it already has a reserved index for the target device. If it does not, the **Palette Manager** checks all windows and reserves the least frequently used indices for your palette. (This reservation process is analogous to that used by the **Color Manager** procedure **ReserveEntry**.) The device's index and its corresponding color value are removed from the matching scheme used by **Color QuickDraw**; you cannot draw with the color by calling **RGBForeColor**. (However, when you call **PmForeColor**, the **Palette Manager** locates the reserved index and configures your window's port to draw with it.) On a multiscreen system the index reserved is likely to be different for each device, but this process is invisible to your application.

After reserving one or more device indices for each animated color it detects, the **Palette Manager** changes the color environment to match the RGB values specified in the palette.

The **Palette Manager** returns the indices used by your animated entries to each screen device in any of these situations:

- a window owning those animated entries moves off of that screen
- your application changes the usage of an animated color
- your application disposes of the palette owning those entries (merely hiding a window does not release its entries)

The **Palette Manager** replaces previously animated indices with the corresponding colors from the default color table for that device.

The **Palette Manager** receives notice when the screen depth changes, so that it can take appropriate action at that time, such as setting color tables to their defaults.

Displaying Animated Colors on Direct Devices

Color table animation does not work on a direct device-it has no color table. To present the best appearance, for example, on a window that spans an indexed device and a direct device, the **Palette Manager** records two colors in the ciRGB field of the color information record: the last color the entry was set to by **SetEntryColor**, and the last color the entry was set to by **AnimateEntry** or **AnimatePalette**. In the palette record, the high bytes of the components in the ciRGB field reflect the animated color, and the low bytes contain the color set by **SetEntryColor**. (**GetEntryColor** returns the last color the entry was animated to.) When you draw with an animated color on a direct device (or on any device on which the animated color was not allocated and reserved), then the color set by **SetEntryColor** is used. This allows successive updates of an animated image on a direct device to match correctly. A side effect is that **GetEntryColor** does not necessarily return an exact match of the color originally set (only the top 8 bits are an exact match).

> **Warning:** This internal usage of the color information fields may change. For maximum safety, use the procedures **SetEntryColor**, **SetEntryUsage**, **GetEntryColor**, and **GetEntryUsage**.