Creating Custom Color-Sampling Methods

Assembly-language programmers can create a custom method for sampling colors. You specify its use by placing the resource ID of your function in the colorPickMethod parameters of the **GetPixMapInfo**, **GetPictInfo**, and **NewPictInfo** functions. Your custom function should be in a resource of type 'cpmt'.

Your function is called with a routine selector in register D0.

D0 value	**Subroutine to call**
D0 = 0	MyInitPickMethod
D0 = 1	MyRecordColors
D0 = 2	MyCalcColorTable
D0 = 3	MyDisposeColorPickMethod

The MyInitPickMethod function is called first; it should allocate storage or perform any other initialization required by your function. The MyRecordColors function is called to record colors if you create your own custom color bank. The MyCalcColorTable function is called if there are more colors in the picture being examined than your application requested to see or if you are recording your own colors. The MyDisposeColorPickMethod function is called when the application is done with the color sampling method. It should release memory requested by the MyInitPickMethod function.

If your routines return an error, that error is passed back to the application.

pascal OSErr **MyInitPickMethod** (short *colorsRequested*, long **dataRef*, short **colorBankType*);

Your **MyInitPickMethod** function must allocate whatever data you need and store a handle to your data in the location pointed to by the *dataRef* parameter. It should also return the type of color bank your function uses for color storage in *colorBankType*.

There are three valid color bank types:

colorBankIsCustom	records colors you specify
colorBankIsExactAnd555	records exact colors
colorBankIs555	records colors in a 5-5-5 histogram

Return colorBankIsCustom if you want to record your own colors; this is the only case in which your RecordColors routine is called. Your CalcColorTable function will also be called.

Return colorBankIs555 if you want the colors stored in a 5-5-5 histogram. The **Picture Utilities Package** calls the CalcColorTable routine with a pointer to the color bank (the 5-5-5 histogram) when the application retrieves color information. The **Picture Utilities Package** does not call your MyRecordColors function.

ColorBankIsExactAnd555 tells the **Picture Utilities Package** to return exact colors. The MyCalcColorTable function should only be called if there are more colors in the picture or pixel map than the application requests. (This is

the same as ColorBankIs555 except that the
Picture Utilities Package returns the exact colors whenever possible.)

The format of the 5-5-5 histogram is like a reversed color table. It is an array of 32,768 integers, where the *index* into the array is the color: 5 bits of red, followed by 5 bits of green, followed by 5 bits of blue. Each *entry* in the array is the number of colors in the picture that are approximated by the color that indexes this entry.

For example, suppose there were three instances of the following color in the pixel map:

Color		**Binary value**
red	=	%1101101010101110
green	=	%0111101010110001
blue	=	%0101101101101010

This color would be represented by index % 0 11011 01111 01011 (in hex 0x6DEB) and the value in the histogram at this index would be 3 (because there were three instances of this color).

pascal OSErr **MyRecordColors** **(**long *dataRef*, RGBColor **colorsArray*, long *colorCount*, long **uniqueColors***)**;

The **Picture Utilities Package** calls your **MyRecordColors** function only if your **InitPickMethod** function returned colorBankIsCustom. **MyRecordColors** is called repeatedly for all the colors in the picture or pixel map. **MyRecordColors** should store the color information for the array of 48-bit colors that it was passed. The *colorCount* parameter indicates how many colors there are in the array *colorsArray*.

Your function must also calculate the number of unique colors (to the resolution of the color bank) that were added by this call to your **MyCalcColorTable** function. It should add this delta amount to the uniqueColors parameter that is passed to it.

pascal OSErr **MyCalcColorTable** **(**long *dataRef*, short *colorsRequested*, Ptr *colorBankPtr*, CSpecArray **resultPtr***)**;

Your **MyCalcColorTable** function should look either at your own internal color bank or at the 5-5-5 histogram passed in the *colorBankPtr* parameter, and it should fill the array of color specification records pointed to by resultPtr with as many colors as were requested. If more colors are requested than you can return, fill the remaining entries with black (0000 0000 0000). If your **MyInitPickMethod** function returned colorBankIsCustom, the *colorBankPtr* value will not be valid, since your color-sampling method is responsible for keeping a record of the colors.

The *colorBankPtr* parameter is of type Ptr because the data stored in the color bank is of the type specified by the **MyInitPickMethod** function. Thus, if you specified colorBankIs555 in the *colorBankType* parameter, the color bank would be an array of integers (see the description of **MyInitPickMethod**). However, if the **Picture Utilities Package** supports other data types in the future, the *colorBankPtr* parameter could point to completely different objects. For now, you should always coerce *colorBankPtr* to a pointer to an integer. In the future you may need to coerce *colorBankPtr* to a pointer of the type you specify in **MyInitPickMethod**.

pascal OSErr **MyDisposeColorPickMethod (**long *dataRef* **);**

Your **MyDisposeColorPickMethod** function should release any memory that you allocated in your **MyInitColorMethod** function.