Creating Palettes

Typically, you create a palette from the colors in a resource of type 'pltt' using the **GetNewPalette** function. You can also create a palette from the colors in a color table by using the **NewPalette** function.

Assigning Colors to a Palette

The inhibited usage categories are the best way to be sure that the right colors are available for screens of different depths, but in many situations you can achieve the same effect with a single set of colors if you sequence the colors in the palette, or arrange them according to the screen depth of the device that uses them, from least to greatest depth.

Color QuickDraw, to support standard **QuickDraw** features, puts white and black at the beginning and end, respectively, of each device's color table, and the **Palette Manager** never changes them. Thus the maximum number of indices available for animated or tolerant colors is really the maximum number of indices minus 2.

After white and black, you should assign the next two colors to the two you wish to have if the device is a 2-bit device. Likewise, the first 16 colors should be the optimal palette entries for a 4-bit device, and the first 256 colors should be the optimal palette entries for an 8-bit device.You should inhibit colors for gray-scale devices.

Creating a Palette in a Resource File

The format of a palette resource (type 'pltt') is an image of the palette structure itself. The private fields in both the header and in each color information record are reserved for future use. The Listing below shows a palette resource with 16 entries as it would appear within a resource file. Each entry has a tolerance value of 0, meaning that the color should be matched exactly.

A palette ('pltt') resource

```
#define zeroTolerance 0

resource 'pltt' (128, "Simple Palette") {
{   //array ColorInfo: 16 elements
    //[1]  white
    65535,    65535,        65535, pmTolerant, zeroTolerance,
    //[2]  black
    0,    0,     0,  pmTolerant, zeroTolerance,
    //[3]  yellow, for bit depths >= 2
    64512,    62333,        1327, pmTolerant, zeroTolerance,
    //[4]  orange
    65535,    25738,        652, pmTolerant, zeroTolerance,
    //[5]  blue-green used in bit depths >= 4
    881,      50943,    40649, pmTolerant, zeroTolerance,
    //[6]  green
    0,    22015,   0,   pmTolerant, zeroTolerance,
    //[7]  blue
    22015,    0,  0,  pmTolerant, zeroTolerance
    //[8]  red
    56683,    2242, 1698, pmTolerant, zeroTolerance,
    //[9]  light gray
    49152,    49152,       49152, pmTolerant, zeroTolerance,
    //[10] medium gray
    32768,    32768,       32768, pmTolerant, zeroTolerance,
    //[11] beige
    65535,    50140,       33120, pmTolerant, zeroTolerance,
    //[12] brown
    37887,    10266,       4812, pmTolerant, zeroTolerance,
    //[13] olive green
    25892,    49919,       0, pmTolerant, zeroTolerance,
    //[14] bright green
    0,    65535,   1265, pmTolerant, zeroTolerance,
    //[15] sky blue
    0,    0,     65535, pmTolerant, zeroTolerance,
    //[16] violet
    32768,    0,  65535,pmTolerant, zeroTolerance
 }
 };
```

Use **GetNewPalette** to obtain a 'pltt' resource; it initializes private fields in the palette data structure. (Do not use **GetResource**.)

The Listing below shows a palette with a variety of usage categories. The first two entries, for white and black, are the same as in the palette in the Listing above. The next three entries are animated and explicit, and inhibited on gray-scale devices. Entries 6 through 14 are tolerant with zero tolerance and inhibited on gray-scale devices. The last two entries in the palette are inhibited on color devices as well if they can only display 16 colors.

```
//A multi-use palette



#define zeroTolerance 0

#define pmInhibitG2 0x0100
#define pmInhibitC2 0x0200
#define pmInhibitG4 0x0400
#define pmInhibitC4 0x0800
#define pmInhibitG8 0x1000
#define pmInhibitC8 0x2000
// inhibit in all gray-scale devices
#define grayDevInhibit (pmInhibitG2  +  pmInhibitG4  +  pmInhibitG8)


#define pmTolerant       2
#define pmAnimated       4
#define pmExplicit       8

#define pmAnimDevInhibit (pmAnimated  +  grayDevInhibit)

// QD does the best job at displaying colors in gray-scale devices
   so it is a good idea to inhibit colors in all gray-scale devices. //
```

```
resource 'pltt' (129, "Inhibiting Palette") {
{ //array ColorInfo: 16 elements
 //[1]
65535,    65535,    65535, pmTolerant, zeroTolerance,
 //[2]
0,  0,       0,  pmTolerant, zeroTolerance,
 //request animated entries to go in known 'clut' slots
 //[3]
64512,    62333,    1327,           (pmAnimDevInhibit + pmExplicit),
 zeroTolerance,
 //[4]
 65535,   25738,    652,            (pmAnimDevInhibit + pmExplicit),
 zeroTolerance,
 //[5]
 881,      50943,   40649,          (pmAnimDevInhibit + pmExplicit),
 zeroTolerance,
 //now let the Palette Manager put the next colors where it wants
 //[6]
 0,        22015,   22015,          (pmTolerant + grayDevInhibit),
 zeroTolerance,
 //[7]
 22015,   0,  0,       (pmTolerant + grayDevInhibit), zeroTolerance,
 //[8]
 56683,   2242, 1698,  (pmTolerant + grayDevInhibit), zeroTolerance,
 //[9]
 49152,   49152,    49152,          (pmTolerant + grayDevInhibit),
 zeroTolerance,
 //[10]
 32768,   32768,    32768,          (pmTolerant + grayDevInhibit),
 zeroTolerance,
 //[11]
 65535,   50140,    33120,           (pmTolerant + grayDevInhibit),
 zeroTolerance,
 //[12]
37887,    10266,    4812,           (pmTolerant + grayDevInhibit),
 zeroTolerance,
 //[13]
 25892,   49919,    0,   (pmTolerant + grayDevInhibit), zeroTolerance,
 //[14]
 0,            65535,     1265,     (pmTolerant + grayDevInhibit),
 zeroTolerance,
 //inhibit the last two entries in 4-bit deep devices
 //[15]
 0,            65535,     0,        (pmTolerant + grayDevInhibit +
 pmInhibitC4), zeroTolerance,
 //[16]
 32768,   0, 65535,      (pmTolerant + grayDevInhibit + pmInhibitC4),
 zeroTolerance
 }
};
```