Technical Note QD04
Colorizing With CopyBits

CONTENTS

Inside Macintosh Volume V states that the foreground and background colors are applied to an image during a `CopyBits` or `CopyMask` call. Accidental use of this feature can create bizarre coloring effects. This note explains what happens, how to avoid problems, and how to use it.

[Nov 01 1987]

What Happens

Color QuickDraw has a feature that will allow you to convert a monochrome image to a color image. During a `CopyBits` or `CopyMask` call, if the foreground and background colors are not black and white, respectively, Color QuickDraw performs the following operation on every pixel being copied:

```
NOTE: color table index = pixel value

s = color table index of source pixel
fg = color table index of foreground color
bg = color table index of background color

ColoredPixelValue = (NOT(s) AND bg) OR (s AND fg)
```

If your source image contains only black and white pixels, then all black pixels would become the foreground color and all white pixels would become the background color. This is because the color table index for white is all zeros and the color table index for black is all ones.

For example, suppose your source image was a 4-bit deep color `PixMap`. Then the color table index for white (in binary) is 0000 and the index for black is 1111. And let's suppose that your foreground color is green with an index of 1101 while your background color is red with an index of 0011. Then for the black pixels, the above procedure produces:

```
ColoredPixelValue = (NOT(1111) AND 0011) OR (1111 AND 1101)
     1101         = (  0000    AND 0011) OR (1111 AND 1101)
```

And the operation on the white pixels yields:

```
ColoredPixelValue = (NOT(0000) AND 0011) OR (0000 AND 1101)
     0011         = (  1111    AND 0011) OR (0000 AND 1101)
```

Back to top

Possible Problems

This colorizing will only work on 2-color (i.e. black and white) images, and then only if those colors occupy the first and last entries in the color table. Trying to colorize colors that are not the first and last color table entries will yield unexpected results.

This is mainly due to the fact that the colorizing algorithm uses a pixel's color table index value rather than its actual RGB color. To illustrate this, let's assume that foreground and background colors are as above, and your image contains yellow with a color table index of 1000. The colorizing operation would give:

```
ColoredPixelValue = (NOT(1000) AND 0011) OR (1000 AND 1101)
     1011         = (  0111    AND 0011) OR (1000 AND 1101)
```

Since the color table may have any RGB color at the resulting index position, the final color may not even be close to the source, foreground, or background colors.

Similar things occur if you are trying to colorize a black and white image when white and black do not occupy the first and last positions in the color table.

The bottom line rules for CopyBitsing in a color environment are these:

- Thou shalt set thy background color to white and thy foreground color to black before calling CopyBits or CopyMask, unless thou art coloring a monochrome image.
- Thou shalt, when colorizing, make sure that the first color table entry is white and the last color table entry is black.

The second rule is easy to follow because the default color tables are constructed properly, and if you are using the Palette Manager (and you are, right?) then it will make sure that the color tables obey this rule.

Back to top

How To Colorize - An Example

This code fragment shows how to implement a color fill, like the paint bucket in MacPaint. It relies on three main things: SeedCFill for calculating the fill area, CopyMask for actually changing the bits, and QuickDraw colorizing.

```
PROCEDURE PaintBucket(where: Point; paintColor: RGBColor);

    VAR
        savedFG    : RGBColor;
        offBits    : BitMap;

    BEGIN
        {First, create an offscreen bitmap.}
        offBits.bounds := myWindow^.portRect;
        WITH offBits.bounds DO BEGIN
            offBits.rowBytes := ((right - left + 15) DIV 16) * 2;
            offBits.baseAddr := NewPtr((bottom-top) * offBits.rowBytes);
        END;

        {Check MemError here! Make sure NewPtr succeeded!}

        SeedCFill(myWindow^.portBits,offBits,myWindow^.portRect,
            myWindow^.portRect,where.h,where.v,NIL,0);
        GetForeColor(savedFG);
        RGBForeColor(paintColor);
        CopyMask(offBits,offBits,myWindow^.portBits,myWindow^.portRect,
            myWindow^.portRect,myWindow^.portRect);
        RGBForeColor(savedFG);

        DisposPtr(offBits.BaseAddr);
    END;
```

The variable `offBits` is an offscreen `BitMap` (not a `PixMap`) with `bounds` = `myWindow^.portRect`. `SeedCFill` effectively creates, in the offscreen `BitMap`, a monochrome image of the bits that we want to paint. Since `offBits` contains the exact bits that we want to paint, it is used as both the source image and the mask for `CopyMask`.

By setting the foreground color to the desired paint color, the result is a colorized version of the mask (the paint area) being copied onto the window's `PixMap` without affecting any other bits.

Back to top

References

Color QuickDraw

Back to top

Downloadables

Acrobat version of this Note (K). Download
