CHAPTER 16: THE PALETTE MANAGER - UNVEILING THE WORLD OF 256 COLORS

This chapter delves into the realm of the Windows Palette Manager, a tool born out of necessity due to certain hardware limitations.

A paletter lookup table on video boards works like:



While modern video adapters often support higher color depths like 24-bit or 16-bit, certain setups, particularly on laptops or in high-resolution modes, are constrained to 8 bits per pixel. This limitation translates to a palette of only 256 simultaneous colors.

The question arises: What can be accomplished with a palette of 256 colors? While 16 colors are insufficient for displaying realistic images, and thousands or millions of colors are more than ample for such tasks, the middle ground of 256 colors presents unique challenges.

To effectively showcase real-world images with this limited palette, colors must be carefully selected for each specific image. A one-size-fits-all "standard" set of 256 colors isn't feasible, as it won't cater to the diverse needs of every application.

Enter the Windows Palette Manager. This tool is designed for precisely specifying the colors required by a program when operating in an 8-bit video mode.

If your programs exclusively run in higher bit depths, you may not encounter the need for the Palette Manager. Nevertheless, this chapter holds valuable insights, particularly in tying up loose ends related to bitmap handling.

Key Points:

Hardware Limitations: Certain video adapters restrict color depth to 8 bits per pixel, allowing only 256 colors simultaneously.

Palette Manager's Purpose: Tailored for programs operating in 8-bit video modes, the Palette Manager enables the specification of essential colors.

Color Selection Challenge: Unlike higher color depths, where a standard set suffices, 256 colors require careful curation for each application's unique needs.

Understanding the Palette Manager is crucial for developers navigating the constraints of 8-bit video modes.

While it may not be applicable in all scenarios, its insights into color management are invaluable, especially when working with real-world images in resource-limited environments.

The 20 reserved colors in 256-color video modes:



The 20 reserved colors in 256-color video modes are part of the Windows Palette Manager, and they serve as standard colors that are predefined for system use. These colors are reserved to maintain consistency across applications running in an 8-bit video mode. Here is a description of each of the 20 reserved colors:

1. Black (Pixel Bits: 00000000, RGB Value: 00 00 00)
2. Dark Red (Pixel Bits: 00000001, RGB Value: 80 00 00)
3. Dark Green (Pixel Bits: 00000010, RGB Value: 00 80 00)
4. Dark Yellow (Pixel Bits: 00000011, RGB Value: 80 80 00)
5. Dark Blue (Pixel Bits: 00000100, RGB Value: 00 00 80)
6. Dark Magenta (Pixel Bits: 00000101, RGB Value: 80 00 80)
7. Dark Cyan (Pixel Bits: 00000110, RGB Value: 00 80 80)
8. Light Gray (Pixel Bits: 00000111, RGB Value: C0 C0 C0)
9. White (Pixel Bits: 11111111, RGB Value: FF FF FF)
10. Cyan (Pixel Bits: 11111110, RGB Value: 00 FF FF)
11. Magenta (Pixel Bits: 11111101, RGB Value: FF 00 FF)
12. Blue (Pixel Bits: 11111100, RGB Value: 00 00 FF)
13. Dark Gray (Pixel Bits: 11111000, RGB Value: 80 80 80)
14. Medium Gray (Pixel Bits: 11110111, RGB Value: A0 A0 A4)
15. Cream (Pixel Bits: 11110110, RGB Value: FF FB F0)
16. Sky Blue (Pixel Bits: 11110101, RGB Value: A6 CA F0)
17. Money Green (Pixel Bits: 11110100, RGB Value: C0 DC C0)
18. Reserved
19. Reserved
20. Reserved

In 256-color display modes, Windows manages a system palette that mirrors the video card's hardware color lookup table. This system palette controls the available colors for display.

By default, Windows provides a specific set of colors as the system palette, which is depicted in Figure above.

It holds 256 colors, 20 of which are fixed for system elements, while applications can customize the remaining 236.

Applications can adjust these colors using logical palettes. If multiple applications use logical palettes, Windows prioritizes the active window (the one in the foreground with a highlighted title bar), ensuring its color choices take precedence.

In scenarios where multiple applications are using logical palettes simultaneously, Windows gives the highest priority to the active window.

The active window refers to the window that currently has the highlighted title bar and appears in the foreground of all other windows.

This ensures that the active application's color choices take precedence over other applications, providing a consistent and coherent visual experience.

To explore this concept practically, we'll examine a sample program later in this chapter. To align with these examples, consider switching your display to 256-color mode.

Access display settings by right-clicking on your desktop, selecting "Properties," and choosing the "Settings" tab.

Key points:

* System palette: Master color table for Windows in 256-color modes.
* Logical palettes: Application-specific color customizations.
* Active window priority: Windows prioritizes the active application's colors.
* Switching to 256-color mode: Recommended for compatibility with examples.

Additional notes:

* While modern systems often use higher color depths, understanding system palettes is still relevant for legacy applications and specific development scenarios.
* The specific process for adjusting display settings may vary slightly depending on your Windows version.

GRAYS1 PROGRAM



The GRAYS1 program is designed to display 65 shades of gray as a "fountain" of color, ranging from black to white. It does not use the Windows Palette Manager but instead directly creates and fills rectangles with varying shades of gray.

The program starts by defining the necessary headers and function prototypes. It then defines the WinMain function, which is the entry point of the program. Inside WinMain, the program registers a window class, creates a window, and enters the message loop.

The WndProc function is the window procedure for handling messages related to the program's window. It handles messages such as WM\_SIZE, WM\_PAINT, and WM\_DESTROY.

In the WM\_SIZE message case, the cxClient and cyClient variables are updated with the width and height of the client area of the window, respectively.

In the WM\_PAINT message case, the program prepares to paint the window. It begins by obtaining a device context (hdc) and a paint structure (ps) using the BeginPaint function. Then, a loop is executed 65 times to create and fill rectangles with varying shades of gray.

Inside the loop, the rect structure is defined to represent the dimensions of each rectangle. The left and right coordinates of the rectangle are calculated based on the current iteration and the total number of shades. The top and bottom coordinates are set to cover the entire height of the client area.

A brush (hBrush) is created using the CreateSolidBrush function, specifying the RGB values for the gray color. The RGB values are derived from the current iteration to create a gradient effect.

The FillRect function is then used to fill the current rectangle with the gray color represented by the brush. After filling the rectangle, the brush is deleted to release the associated resources.

Finally, the program calls EndPaint to signal the end of the painting process and returns 0 to indicate that the message has been handled.

In the WM\_DESTROY message case, the program posts a quit message to exit the message loop and terminate the program.

If any other messages are received or not handled in the WndProc function, the program calls DefWindowProc to perform the default window procedure for those messages.



*Let's dive deeper into the GRAYS1 program and explore the section where color palettes are mentioned.*

In the given code, the GRAYS1 program does not use the Windows Palette Manager. Instead, it directly creates and fills rectangles with varying shades of gray using the RGB color model.

The concept of color palettes in computer graphics refers to a limited set of colors that are available for use in a particular system or application. In the Windows operating system, a palette is a data structure that holds a fixed number of colors, typically 256 colors. The Windows Palette Manager is responsible for managing and mapping colors from the system palette to the colors used by an application.

However, in the GRAYS1 program, the focus is on displaying shades of gray rather than utilizing a predefined color palette. The program achieves this by dynamically calculating and creating shades of gray using the CreateSolidBrush function and RGB values.

Within the WM\_PAINT message case, a loop is executed 65 times to create 65 rectangles, each representing a different shade of gray. The RGB values for each shade are calculated as min(255, 4 \* i), where i is the current iteration of the loop.

By multiplying i by 4 and clamping the result to a maximum of 255, the program ensures that the RGB values stay within the valid range for a grayscale color. This calculation creates a gradient effect, where the shades of gray become progressively lighter as i increases.

The CreateSolidBrush function is then used to create a brush with the calculated RGB values, representing the current shade of gray. The FillRect function fills the current rectangle with the gray color represented by the brush.

It's important to note that by creating and using brushes directly, the GRAYS1 program bypasses the Windows Palette Manager and the limitations of a fixed color palette. Instead, it dynamically generates and displays the shades of gray as a "fountain" of color in the client area of the window.

This approach allows for greater flexibility in displaying a wider range of shades and gradients, as it leverages the full RGB color space rather than being constrained by a predefined palette.