Indeo™ Video's Palette under Video for Windows™

A popular frog likes to sing "It's not easy being green". If the frog knew anything about 256-color palettes, he would be tempted to sing "It's not easy staying green". This article talks about the good and bad attributes of palettes and shows how to avoid some potentially ugly attributes. It shows how to adjust Indeo video clips to make them look good with an application's palette. Or, if so desired, it shows how to adjust an application's palette to make it look good with Indeo video clips. All of this is done to make sure the reds stay red, the blues stay blue, and as far as frogs are concerned, the greens stay green.

Background on Palettes

A palette refers to the current set of 256 colors used by a graphics card when the card is in 8-bit color mode. Although, only one palette is active at a time, it is possible to change palettes. When an application's window comes to the foreground, its palette is asserted.

Palettes are maintained in a Color Lookup Table (CLUT). Each entry of the table represents a color. Although 256 colors are allowed, when a new palette is created, it usually specifies no more than 236 colors. The first 10 and last 10 entries are reserved by the system. This prevents system colors from being disrupted when palettes are switched.

Indeo Video's Fixed Palette

Indeo video has always used a native fixed palette. The Indeo video palette contains the same 236 colors regardless of the video content. It was designed to produce good results across a broad range of image types and is optimized for the fastest possible playback in 8-bit color mode.

Using a fixed palette does have limitations. When a video clip is played, its palette comes to the foreground. Microsoft Windows* applies the foreground palette to the entire screen. Palette flash occurs at this time if the video's palette differs from the background palettes. The background images flash as Windows does a "best match" of the foreground colors to the background colors.

With Indeo video's fixed palette, background windows can avoid palette flash only if one of the following work arounds is used:

1. The backgrounds only use the 20 system colors.
2. The backgrounds only use Indeo video's fixed palette colors.
3. The backgrounds that use a different palette are not displayed at the same time as Indeo video clips.

Another limitation of a fixed palette is the static colors within the palette. The fixed palette may not provide the best mix of colors for an application. This can be especially noticeable if highly saturated colors are used. These limitations can be minimized if an application has the ability to select a single palette designed to produce a good video image with a specific background image.

Active Palette Support

The term "Active Palette" refers to the ability within Video for Windows to use an external palette. VfW 1.1d contains an updated Indeo video R3.2 driver that is capable of activating an external palette. When an Indeo video clip is played, an external palette can be activated rather than the internal fixed palette. Intel has always recommended using Indeo video's fixed palette throughout an application to avoid palette flash. This is still a good idea, whenever possible. But now, with active palette support, a custom palette can be created that closely matches the colors used by an application without worrying about palette flash. When a video clip and its background images use the same custom palette, no palette flash occurs.

Video for Windows has always been able to assert an external palette. Before active palette was implemented, Windows would dither the video image after Indeo video dithered the image. This resulted in poor playback performance and poor color quality. To solve this problem, Microsoft introduced active palette support in Video for Windows (VfW) 1.1d. With active palette support, the video image is only dithered by Indeo video.

Limitations of Active Palette

Active palette support does have the following drawbacks:

1. Playback performance is slightly degraded.
2. Initialization time is longer.
3. Uses more system memory. 160x120 clips use 64K bytes more memory. 240x180 and 320x240 clips use 128K bytes more memory.

Activating a Palette

Active palette is usually implemented at the Media Control Interface (MCI) level using the SETVIDEO command. The MCI REALIZE command can also be used but only for activating a background palette. MCI commands are documented in the Video for Windows Developer's Kit. The following line gives the SETVIDEO syntax for activating an external palette.

```
SETVIDEO <alias> PALETTE HANDLE TO <palette-handle>
```

The alias is a name that can be given a clip when it is opened. The palette handle points to the external palette.

Active palette can be implemented at a lower programming level using the DrawDibSetPalette() function call. However, a lot more coding is needed at this level.

It is also possible to activate a palette without entering any MCI commands. Indeo video clips will use whatever palette is forced to the foreground. Normally, a video clip stays in the foreground while it is playing. However, if another window is forced to the foreground while an Indeo video clip is playing, the clip will automatically switch to the foreground window's palette.

This last method can be tested using Media Player. Open up an Indeo video clip with Media Player. Start the clip playing and use the mouse to click on another window that uses a different palette. This brings the other window to the foreground along with its palette. The Indeo video clip will switch over to the new foreground palette and continue playing. The other window can contain any type of image that has its own palette, such as a bitmap or even another video clip.

MCITest Example

Provided below is a sequence of MCI command strings that implement active palette support. One way to test these commands is by using the MCITest utility provided as a sample with Microsoft Visual C++*. MCITest puts up a dialog box that allows MCI command strings to be entered and executed.

The Indeo video clip in this example is called indeo.avi. Rather than using it's internal fixed palette, the indeo.avi clip is forced to use an external palette that is stored in another AVI file called gray.avi. The gray.avi file is used to supply the external palette because MCI does not support opening .pal palette files. Gray.avi is a Microsoft Video 1 clip. It is included with this document. The indeo.avi file is not include with this article. Any Indeo video R3.2 clip may be used instead.

```
  open gray.avi alias pal
  status pal palette handle
  open indeo.avi alias movie
  setvideo movie palette handle to XXXX
  window movie state show
  play movie
  close movie
  close pal
```

Gray.avi contains one video frame and a 236 color palette. Each entry of its palette is a different shade of gray. The XXXX parameter used by the SETVIDEO command represents the integer handle returned earlier by the

STATUS command. If the external palette is very similar to Indeo video's native palette, it can be difficult to see if the external palette is really being used. The gray palette in this example makes it easy to detect when that palette is active. Some colors may still show up in the video because the 20 system colors are still available.

It is important to note the SETVIDEO command must be executed before the WINDOW command. Otherwise, the external palette is not activated properly. The following sequence of commands does not work:

```
window movie state show
setvideo movie palette handle to XXXX
play movie
```

Microsoft's Media Player is also able to execute MCI commands. Typing CTRL-F5 after a clip has been opened brings up a dialog box for entering MCI commands. Unfortunately, Media Player can not be used to test the SETVIDEO command. When Media Player opens an AVI file, it immediately shows the clip before the SETVIDEO command can be entered.

Implementing Active Palette

MCI commands are typically executed within a program. The following lines of code show how to execute the SETVIDEO command from within a Microsoft Visual C++ program.

```
HPALETTE hpal;               // palette handle
wsprintf(achCommand, "setvideo clip palette handle to %d", hpal);
mciSendString((LPSTR((achCommand, NULL, 0, NULL);
```

Two longer examples are provided at the end of this article. Both examples use source code from the Video for Windows Developer's Kit. They are based on the MOVPLAY2 movie player sample. The first example modifies movplay2.c to allocate a palette and activate it when playing an Indeo video clip. The second example plays clips using an external palette stored on Window's Clipboard. The executable for the second example is provided with this article. It was renamed to palplay.exe.

Tips on Creating an External Palette

Here are some tips on how to create a proper external palette:

1. Create a palette with 236 entries. A palette with more than 236 entries will just get truncated after the 236th entry. It is alright to create a palette with less than 236 entries. However, Windows will automatically fill in the missing entries with colors from Indeo video's fixed palette.
2. Don't bother including system palette entries with a 236 color palette. They get asserted automatically. Any duplicates of the system palette entries are dropped. At the end of this article is a chart listing the RGB values of the system palette entries.
3. Avoid duplicate entries with the same RGB value. For example, a palette that has 200 entries with the same RBG value, will end up with only one entry of that value. The other 199 entries will be automatically filled with colors from Indeo video's fixed palette.
4. Minimize or close Microsoft's PalEdit before asserting an external palette. External palettes are asserted incorrectly if PalEdit is maximized at the same time. Microsoft used to supply PalEdit with Video for Windows 1.0. Asymetrix now includes PalEdit with their Toolbook* application.
5. Display the current foreground palette during development. By looking at the current foreground palette, it is easy to verify when the correct palette is asserted. Microsoft supplies sample code with Visual C++ for a palette utility called palette.exe. The only thing this utility really does is display the current foreground palette. The previous tips were learned by looking at the foreground palette after an external palette was asserted. Palette.exe is included with this document.

Grabbing Indeo video's fixed palette

Since an external palettes is normally only activated by an MCI command, some applications can not use this feature. Using Indeo video's fixed palette throughout an application is still a good option. There are a couple of steps that need to be followed to properly grab Indeo video's fixed palette. Since the palette resides in the codec, the Indeo video driver has to be loaded before the palette can be obtained. The following steps use Microsoft's Media Player and PalEdit utility to grab the palette, edit it, and then save it to a .pal palette file.

1. Put a video frame onto Window's Clipboard.

Open an Indeo video clip with Microsoft's Media Player and copy a frame using the Edit/Copy Object menu command. This puts the current video frame on the Clipboard along with its palette. A palette with 256 entries is copied to the Clipboard. This is Indeo video's fixed palette plus the system entries. The Indeo video palette is preceded by the first 10 system color colors and followed by the last 10 system colors.

Do not use Microsoft's VidEdit to put a video frame on the Clipboard. VidEdit forces the system palette onto a video clip while it is stopped.

2. Paste the palette into PalEdit.

Use the menu item Edit/Paste to bring the palette into PalEdit.

3. Edit the palette and save it.

The fixed palette does not appear to paste correctly into PalEdit. By looking at the current foreground palette with the palette.exe file, it can be seen that the first 10 system entries are duplicated. This problem can be cleaned up by deleting the last 10 entries of the palette and then deleting the first 10 entries. A proper 236 entry palette, containing Indeo video's fixed palette, can then be saved.

Note: The video graphics card needs to be in 256 color mode for this procedure to work properly.

It is not possible to see the duplicate palette entries in step 3 unless the palette.exe utility is used. The Indeo video palette always begins with two gray entries and ends with two more gray entries. After the palette is edited, PalEdit shows the Indeo video palette using the first 236 entries.

Once the Indeo video palette has been saved, it can be used by other images within an application, such as bitmaps. Microsoft's BitEdit utility is capable of switching the palette used by a bitmap. Included with this article is a file called indeo.pal. It contains Indeo video's fixed 236 color palette.

Conclusion

There are now two ways to avoid palette flash within an application. Use Indeo video's fixed palette throughout an application or activate an external palette. Indeo video's fixed palette has the benefit of optimum performance. Active palette allows a custom palette to be used with colors that closely match an application's content.

Example #1 - Allocate and Activate an External Gray Palette

This example adds code to the movplay2.c file that comes with the Video for Windows 1.1 Developer's Kit. This application shows how to allocate and initialize an external gray palette prior to activating it.

```
// hpal is the palette that will be asserted.  It is initialized
// to 236 gray colors in a new function called initPalette().
// hpal is later asserted in fileOpenMovie().

HPALETTE hpal;  // new global variable

// initPalette - creates the desired palette for the movie clip

void initPalette() // new function
{
 LOGPALETTE *plgpl;
 int i;
 HANDLE hDC;

 // Allocate the logical palette.
 plgpl = (LOGPALETTE*) LocalAlloc(LPTR,
         sizeof(LOGPALETTE) + 236 * sizeof(PALETTEENTRY));
 plgpl->palVersion = 0x300;
 plgpl->palNumEntries = 236;

 // set each entry to gray color values
 for (i = 0; i < 236; ++i){
   plgpl->palPalEntry[i].peRed = i + 10;
   plgpl->palPalEntry[i].peGreen = i + 10;
   plgpl->palPalEntry[i].peBlue = i + 10;
   plgpl->palPalEntry[i].peFlags = PC_NOCOLLAPSE;
   }

 // Create the palette object and get a handle to it.
 hpal = CreatePalette(plgpl);
 LocalFree((HLOCAL) plgpl);
}
...
```

```
void fileOpenMovie(HWND hWnd)
{
 ...
 wsprintf((LPSTR)achCommand,"open %s alias mov style child parent %d",
          ofn.lpstrFile,hWnd);

 /* try to open the file */
 if (mciSendString((LPSTR)achCommand, NULL, 0, NULL) == 0){
    fMovieOpen = TRUE;

 //****** begin new code

    initPalette();
    wsprintf(achCommand, "setvideo mov palette handle to %d", hpal);
    if (mciSendString((LPSTR)achCommand, NULL, 0, NULL) != 0){
        MessageBox(hWnd, "Unable to set palette handle", NULL,
        MB_ICONEXCLAMATION|MB_OK); }

 //****** end new code

 /* we opened the file o.k., now set up to */
 /* play it.                           */
 mciSendString("window mov state show", NULL, 0, NULL);
 ...
}
```

Example #2 - Activate Palette on Window's Clipboard

This example adds a few lines of code to the movplay2.c file that comes with the Video for Windows Developer's Kit. Rather than allocate and initialize a palette as in the first example, this example gets a handle to an external palette on Window's Clipboard. It uses the Clipboard palette as the active palette. The executable for this file is supplied with this article. It was renamed to palplay.exe.

A quick way to test palplay.exe is by using the gray palette supplied with this article. Open up gray.avi with Media Player. Use Edit/Copy Object to put the palette on the Clipboard. Any Indeo video clips that are then played using palplay.exe will use the gray palette.

```
...
void fileOpenMovie(HWND hWnd)
{
 HPALETTE hClip_palette;  // add this new variable

 ...
 wsprintf((LPSTR)achCommand,"open %s alias mov style child parent %d",
          ofn.lpstrFile,hWnd);

 /* try to open the file */
 if (mciSendString((LPSTR)achCommand, NULL, 0, NULL) == 0){
   fMovieOpen = TRUE;

 //**** begin new code

 // get palette from Clipboard
 if (OpenClipboard(hWnd)){
   if (hClip_palette = GetClipboardData(CF_PALETTE)){
     wsprintf((LPSTR)achCommand,"setvideo mov palette handle to %d",
              hClip_palette);
     mciSendString((LPSTR)achCommand, NULL, 0, NULL);
     }
   CloseClipboard();
   }

 //**** end new code

 /* we opened the file o.k., now set up to */
 /* play it.                            */
 mciSendString("window mov state show", NULL, 0, NULL);
 ...
```

System Palette Entries

The following chart shows the RGB entries for a 256 color palette. The first 10 and last 10 entries are defined by the system. If the middle 236 entries belong to Indeo video's fixed palette, it begins and ends with 2 gray color entries.

Entry	Red	Green	Blue	Color	
0	0	0	0	black	
1	0x80	0	0	dark red	
2	0	0x80	0	dark green	
3	0x80	0x80	0	dark yellow	
4	0	0	0x80	dark blue	First 10 System colors
5	0x80	0	0x80	dark magenta	
6	0	0x80	0x80	dark cyan	
7	0xC0	0xC0	0xC0	light gray	
8	0xC0	0xDC	0xC0	green	
9	0xA6	0xCA	0xF0	sky blue	
10	0x81	0x81	0x81	gray	
11	0x81	0x81	0x82	gray	
...					Indeo video's fixed palette
244	0x81	0x81	0x83	gray	
245	0x81	0x81	0x84	gray	
246	0xFF	0xFB	0xF0	offwhite	
247	0xA0	0xA0	0xA4	light gray	
248	0x80	0x80	0x80	medium gray	
249	0xFF	0	0	bright red	
250	0	0xFF	0	bright green	Last 10 system colors
251	0xFF	0xFF	0	yellow	
252	0	0	0xFF	bright blue	
253	0xFF	0	0xFF	magenta	
254	0	0xFF	0xFF	cyan	
255	0xFF	0xFF	0xFF	white	