第十六章　调色盘管理器

如果硬体允许，本章就没有存在的必要。尽管许多现代的显示卡提供 24 位元颜色（也称「true color」或「数百万色」）或 16 位元颜色（「增强色」或「数万种颜色」），一些显示卡——尤其是在携带型电脑上或高解析度模式中——每个图素只允许 8 位元。这意味著仅有 256 种颜色。

我们用 256 种颜色能做什么呢？很明显，要显示真实世界的图像，仅 16 种颜色是不够的，至少要使用数千或数百万种颜色，256 种颜色位於中间状态。是的，用 256 种颜色来显示真实世界的图像足够了，但需要根据特定的图像来指定这些颜色。这意味著作业系统不能简单地选择「标准」系列的 256 种颜色，就希望它们对每个应用程式都是理想的颜色。

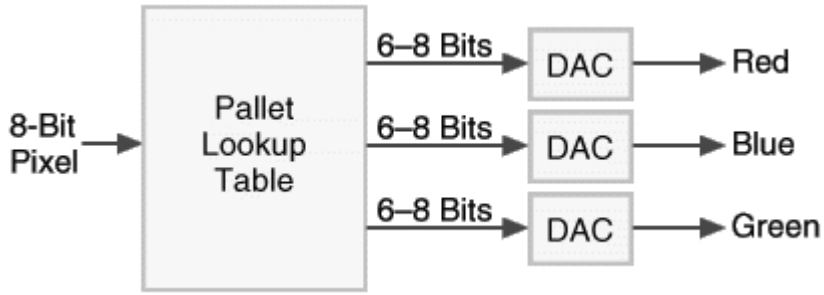
这就是 Windows 调色盘管理器所要涉及的全部内容。它用於指定程式在 8 位元显示模式下执行时所需要的颜色。如果知道程式肯定不会在 8 位元显示模式下执行，那么您也不需要使用调色盘管理器。不过，由於补充了点阵图的一些细节，所以本章还是包含重要资讯的。

使用调色盘

传统上讲，调色盘是画家用来混合颜色的板子。这个词也可以指画家在绘画过程中使用的所有颜色。在电脑图形中，调色盘是在图形输出设备（例如视讯显示器）上可用的颜色范围。这个名词也可以指支援 256 色模式的显示卡上的对照表。

视频硬体

显示卡上的调色盘对照表运作过程如下图所示：



在 8 位元显示模式中，每个图素占 8 位元。图素值查询包含 256RGB 值的对照表的位址。这些 RGB 值可以正好 24 位元宽，或者小一点，通常是 18 位元宽（即主要的红、绿和蓝各 6 位元）。每种颜色的值都输入到数位类比转换器，以得到发送给监视器的红、绿和蓝三个类比信号。

通常，软体可以用任意值来载入调色盘对照表，但这对装置无关的视窗介面，例如 Microsoft Windows，会有一些干扰。首先，Windows 必须提供软体介面，以便在不直接干扰硬体的情况下，应用程式就可以存取调色盘管理器。第二个问题更严重：因为所有的应用程式都共用同一个视讯显示器，而且同时执行，所以一个应用程式使用了调色盘对照表可能会影响其他程式的使用。

这时就需要使用 Windows 调色盘管理器(在 Windows 3.0 中提出)了。Windows 保留了 256 种颜色中的 20 种，而允许应用程式修改其余的 236 种。（在某些情况下，应用程式最多可以改变 256 种颜色中的 254 种——只有黑色和白色除外——但这有一点麻烦）。Windows 为系统保留的 20 种颜色（有时称为 20 种「静态」颜色）如表 16-1 所示。

表 16-1 256 种颜色显示模式中的 20 种保留的颜色

图素位元	RGB 值	颜色名称	图素位元	RGB 值	颜色名称
00000000	00 00 00	黑	11111111	FF FF FF	白
00000001	80 00 00	暗红	11111110	00 FF FF	青
00000010	00 80 00	暗绿	11111101	FF 00 FF	洋红
00000011	80 80 00	暗黄	11111100	00 00 FF	蓝
00000100	00 00 80	暗蓝	11111011	FF FF 00	黄
00000101	80 00 80	暗洋红	11111010	00 FF 00	绿
00000110	00 80 80	暗青	11111001	FF 00 00	红
00000111	C0 C0 C0	亮灰	11111000	80 80 80	暗灰
00001000	C0 DC C0	美元绿	11110111	A0 A0 A4	中性灰
00001001	A6 CA F0	天蓝	11110110	FF FB F0	乳白色

在 256 种颜色显示模式下执行时，由 Windows 维护系统调色盘，此调色盘与显示卡上的硬体调色盘对照表相同。内定的系统调色盘如表 16-1 所示。应用程式可以通过指定「逻辑调色盘（logical palettes）」来修改其余 236 种颜色。如果有多个应用程式使用逻辑调色盘，那么 Windows 就给活动视窗最高优先权（我们知道，活动视窗有高亮显示标题列，并且显示在其他所有视窗的前面）。我们将用一些简单的范例程式来检查它是如何工作的。

要执行本章其他部分的程式，您可能需要将显示卡切换成 256 色模式。在桌面上单擎滑鼠右键，从功能表中选择「属性」，然後选择「设定」页面标签。

显示灰阶

程式 16-1 所示的 GRAYS1 程式没有使用 Windows 调色盘管理器，而尝试用正常显示的 65 级种阶作为从黑到白的多种彩色的「来源」。

程式 16-1 GRAYS1

```
GRAYS1.C
/*-------------------------------------------------------------------------
-
      GRAYS1.C --      Gray Shades
                                              (c) Charles Petzold, 1998
-------------------------------------------------------------------------
-*/

#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
      static TCHAR     szAppName[] = TEXT ("Grays1") ;
      HWND                         hwnd ;
      MSG                    msg ;
      WNDCLASS                     wndclass ;

      wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc              = WndProc ;
      wndclass.cbClsExtra               = 0 ;
      wndclass.cbWndExtra               = 0 ;
      wndclass.hInstance                =     hInstance ;
      wndclass.hIcon                    =     LoadIcon                (NULL,
IDI_APPLICATION) ;
      wndclass.hCursor                  =     LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground       =     (HBRUSH)              GetStockObject
(WHITE_BRUSH) ;
      wndclass.lpszMenuName        =     NULL ;
      wndclass.lpszClassName       =     szAppName ;

      if (!RegisterClass (&wndclass))
      {
                    MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                szAppName, MB_ICONERROR) ;
                    return 0 ;
      }

      hwnd = CreateWindow (  szAppName, TEXT ("Shades of Gray #1"),
                          WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

      ShowWindow (hwnd, iCmdShow) ;
      UpdateWindow (hwnd) ;
```

```
     while (GetMessage (&msg, NULL, 0, 0))
     {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (  HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
     static int            cxClient, cyClient ;
     HBRUSH          hBrush ;
     HDC             hdc ;
     int             i ;
     PAINTSTRUCT           ps ;
     RECT            rect ;

     switch (message)
     {
     case  WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

     case  WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;

                                   // Draw the fountain of grays

                    for (i = 0 ; i < 65 ; i++)
                    {
                                   rect.left             = i * cxClient
/ 65 ;
                                   rect.top         = 0 ;
                                   rect.right            =  (i  +  1)  *
cxClient / 65 ;
                                   rect.bottom           = cyClient ;

                          hBrush = CreateSolidBrush (RGB(min (255, 4 * i),

                                        min (255, 4 * i),
                                        min (255, 4 * i))) ;
                                   FillRect (hdc, &rect, hBrush) ;
                                   DeleteObject (hBrush) ;
                    }
                    EndPaint (hwnd, &ps) ;
```

```
                    return 0 ;

     case  WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

在 WM_PAINT 讯息处理期间，程式呼叫了 65 次 FillRect 函式，每次都使用不同灰阶建立的画刷。灰阶值是 RGB 值（0,0,0）、（4,4,4）、（8,8,8）等等，直到最後一个值（255,255,255）。最後一个值来自 CreateSolidBrush 函式中的 min 巨集。

如果在 256 色显示模式下执行该程式，您将看到从黑到白的 65 种灰阶，而且它们几乎都用混色著色。纯颜色只有黑色、暗灰色（128,128,128）、亮灰色（192,192,192）和白色。其他颜色是混合了这些纯颜色的多位元模式。如果我们在显示行或文字，而不是用这 65 种灰阶填充区域，Windows 将不使用混色而只使用这四种纯色。如果我们正在显示点阵图，则图像将用 20 种标准 Windows 颜色近似。这时正如同您在执行最後一章中的程式的同时又载入了彩色或灰阶 DIB 所见到的一样。通常，Windows 在点阵图中不使用混色。

程式 16-2 所示的 GRAYS2 程式用较少的外部程式码验证了调色盘管理器中最重要的函式和讯息。

程式 16-2　GRAYS2

```
GRAYS2.C
/*-------------------------------------------------------------------------
-
     GRAYS2.C --      Gray Shades Using Palette Manager
                                          (c) Charles Petzold, 1998
-------------------------------------------------------------------------
-*/

#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                PSTR    szCmdLine,    int
iCmdShow)
{
     static TCHAR     szAppName[] = TEXT ("Grays2") ;
     HWND                         hwnd ;
     MSG                  msg ;
     WNDCLASS                     wndclass ;

     wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc              = WndProc ;
```

```
     wndclass.cbClsExtra              = 0 ;
     wndclass.cbWndExtra              = 0 ;
     wndclass.hInstance               = hInstance ;
     wndclass.hIcon                   = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor                 = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground      = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName       = NULL ;
     wndclass.lpszClassName      = szAppName ;

     if    (!    RegisterClass (&wndclass))
     {
                    MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                    szAppName, MB_ICONERROR) ;
                return 0 ;
     }

     hwnd = CreateWindow (  szAppName, TEXT ("Shades of Gray #2"),
                            WS_OVERLAPPEDWINDOW,
                            CW_USEDEFAULT, CW_USEDEFAULT,
                            CW_USEDEFAULT, CW_USEDEFAULT,
                            NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (   HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
     static HPALETTE        hPalette ;
     static int             cxClient, cyClient ;
     HBRUSH                 hBrush ;
     HDC                    hdc ;
     int                    i ;
     LOGPALETTE                  *     plp ;
     PAINTSTRUCT            ps ;
     RECT                   rect ;

     switch (message)
     {
     case  WM_CREATE:
```

```
                                    // Set up a LOGPALETTE structure and create
a palette

                         plp  =  malloc  (sizeof  (LOGPALETTE)  +  64  *  sizeof
(PALETTEENTRY)) ;

                         plp->palVersion  = 0x0300 ;
                         plp->palNumEntries     = 65 ;

                         for (i = 0 ; i < 65 ; i++)
        {
                         plp->palPalEntry[i].peRed    = (BYTE) min (255, 4 * i) ;
                         plp->palPalEntry[i].peGreen  = (BYTE) min (255, 4 * i) ;
                         plp->palPalEntry[i].peBlue   = (BYTE) min (255, 4 * i) ;
                         plp->palPalEntry[i].peFlags  = 0 ;
                         }
                         hPalette = CreatePalette (plp) ;
                         free (plp) ;
                         return 0 ;

      case  WM_SIZE:
                         cxClient = LOWORD (lParam) ;
                         cyClient = HIWORD (lParam) ;
                         return 0 ;
      case  WM_PAINT:
                         hdc = BeginPaint (hwnd, &ps) ;

                                    // Select and realize the palette in the device
context

                         SelectPalette (hdc, hPalette, FALSE) ;
                         RealizePalette (hdc) ;

                                    // Draw the fountain of grays

                         for (i = 0 ; i < 65 ; i++)
                         {
                                    rect.left              = i * cxClient / 64 ;
                                    rect.top         = 0 ;
                                    rect.right             = (i + 1) * cxClient
/ 64 ;
                                    rect.bottom            = cyClient ;

                               hBrush = CreateSolidBrush (PALETTERGB( min
      (255, 4 * i),
                                                  min      (255, 4 * i),
                                                  min      (255, 4 * i))) ;
                                    FillRect (hdc, &rect, hBrush) ;
```

```
                                DeleteObject (hBrush) ;
                        }
                        EndPaint (hwnd, &ps) ;
                        return 0 ;

     case  WM_QUERYNEWPALETTE:
                        if (!hPalette)
                                        return FALSE ;

                        hdc = GetDC (hwnd) ;
                        SelectPalette (hdc, hPalette, FALSE) ;
                        RealizePalette (hdc) ;
                        InvalidateRect (hwnd, NULL, TRUE) ;

                        ReleaseDC (hwnd, hdc) ;
                        return TRUE ;

     case  WM_PALETTECHANGED:
                        if (!hPalette || (HWND) wParam == hwnd)
                                        break ;

                        hdc = GetDC (hwnd) ;
                        SelectPalette (hdc, hPalette, FALSE) ;
                        RealizePalette (hdc) ;
                        UpdateColors (hdc) ;

                        ReleaseDC (hwnd, hdc) ;
                        break ;

     case  WM_DESTROY:
                        DeleteObject (hPalette) ;
                        PostQuitMessage (0) ;
                        return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

通常，使用调色盘管理器的第一步就是呼叫 CreatePalette 函式来建立逻辑调色盘。逻辑调色盘包含程式所需要的全部颜色——即 236 种颜色。GRAYS1 程式在 WM_CREATE 讯息处理期间处理此作业。它初始化 LOGPALETTE（「logical palette:逻辑调色盘」)结构的栏位，并将这个结构的指标传递给 CreatePalette 函式。CreatePalette 传回逻辑调色盘的代号，并将此代号储存在静态变数 hPalette 中。

LOGPALETTE 结构定义如下：

```
typedef struct
{
```

```
      WORD             palVersion ;
      WORD             palNumEntries ;
      PALETTEENTRY     palPalEntry[1] ;
}
LOGPALETTE, * PLOGPALETTE ;
```

第一个栏位通常设为 0x0300，表示相容 Windows 3.0。第二个栏位设定为调色盘表中的项目数。LOGPALETTE 结构中的第三个栏位是一个 PALETTEENTRY 结构的阵列，此结构也是一个调色盘项目。PALETTEENTRY 结构定义如下：

```
typedef struct
{
      BYTE peRed ;
      BYTE peGreen ;
      BYTE peBlue ;
      BYTE peFlags ;
}
PALETTEENTRY, * PPALETTEENTRY ;
```

每个 PALETTEENTRY 结构都定义了一个我们要在调色盘中使用的 RGB 颜色值。

注意，LOGPALETTE 中只能定义一个 PALETTEENTRY 结构的阵列。您需要为 LOGPALETTE 结构和附加的 PALETTEENTRY 结构配置足够大的记忆体空间。GRAYS2 需要 65 种灰阶，因此它为 LOGPALETTE 结构和 64 个附加的 PALETTEENTRY 结构配置了足够大的记忆体空间。GRAYS2 将 palNumEntries 栏位设定为 65，然後从 0 到 64 回圈，计算灰阶等级（一般是回圈索引的 4 倍，但不超过 255），将结构中的 peRed、peGreen 和 peBlue 栏位设定为此灰阶等级。peFlags 栏位设为 0。程式将指向这个记忆体块的指标传递给 CreatePalette，在一个静态变数中储存该调色盘代号，然後释放记忆体。

逻辑调色盘是 GDI 物件。程式应该删除它们建立的所有逻辑调色盘。WndProc 透过在 WM_DESTROY 讯息处理期间呼叫 DeleteObject，仔细地删除了逻辑调色盘。

注意逻辑调色盘是独立的装置内容。在真正使用之前，必须确保将其选进装置内容。在 WM_PAINT 讯息处理期间，SelectPalette 将逻辑调色盘选进装置内容。除了含有第三个参数以外，此函式与 SelectObject 函式相似。通常，第三个参数设为 FALSE。如果 SelectPalette 的第三个参数设为 TRUE，那么调色盘将始终是「背景调色盘」，这意味著当其他所有程式都显现了各自的调色盘之後，该调色盘才可以获得仍位於系统调色盘中的一个未使用项目。

在任何时候都只有一个逻辑调色盘能选进装置内容。函式将传回前一个选进装置内容的逻辑调色盘代号。如果您希望将此逻辑调色盘重新选进装置内容，则可以储存此代号。

通过将颜色映射到系统调色盘，RealizePalette 函式使 Windows 在装置内

容中「显现」逻辑调色盘，而系统调色盘是与显示卡实际的实际调色盘相对应。实际工作在此函式呼叫期间进行。Windows 必须决定呼叫函式的视窗是活动的还是非活动的，并尽可能将系统调色盘已改变通知给其他视窗（我们将简要说明一下通知的程序）。

回忆一下 GRAYS1，它用 RGB 巨集来指定纯色画刷的颜色。RGB 巨集建构一个 32 位元长整数（记作 COLORREF 值），其中高位元组是 0，3 个低位元组是红、绿和蓝的亮度。

使用 Windows 调色盘管理器的程式可以继续使用 RGB 颜色值来指定颜色。不过，这些 RGB 颜色值将不能存取逻辑调色盘中的附加颜色。它们的作用与没有使用调色盘管理器相同。要在逻辑调色盘中使用附加的颜色，就要用到 PALETTERGB 巨集。除了 COLORREF 值的高位元组设为 2 而不是 0 以外，「调色盘 RGB」颜色与 RGB 颜色很相似。

下面是重要的规则：

- 为了使用逻辑调色盘中的颜色，请用调色盘 RGB 值或调色盘索引来指定（我将简要讨论调色盘索引）。不要使用常规的 RGB 值。如果使用了常规的 RGB 值，您将得到一种标准颜色，而不是逻辑调色盘中的颜色。
- 没有将调色盘选进装置内容时，不要使用调色盘 RGB 值或调色盘索引。
- 尽管可以使用调色盘 RGB 值来指定逻辑调色盘中没有的颜色，但您还是要从逻辑调色盘获得颜色。

例如，在 GRAYS2 中处理 WM_PAINT 期间，当您选择并显现了逻辑调色盘之後，如果试图显示红色，则将显示灰阶。您必须用 RGB 颜色值来选择不在逻辑调色盘中的颜色。

注意，GRAYS2 从不检查视讯显示驱动程式是否支援调色盘管理程式。在不支援调色盘管理程式的显示模式（即所有非 256 种颜色的显示模式）下执行 GRAYS2 时，GRAYS2 的功能与 GRASY1 相同。

调色盘资讯

如果程式在逻辑调色盘中指定一种颜色，该颜色又是 20 种保留颜色之一，那么 Windows 将把逻辑调色盘项目映射给该颜色。另外，如果两个或多个应用程式都在它们的逻辑调色盘中指定了同一种颜色，那么这些应用程式将共用系统调色盘项目。程式可以通过将 PALETTEENTRY 结构的 peFlags 栏位指定为常数 PC_NOCOLLAPSE 来忽略该内定状态（其余两个可能的标记是 PC_EXPLICIT（用於显示系统调色盘）和 PC_RESERVED（用於调色盘动画），我将在本章的後面展示这两个标记）。

要帮助组织系统调色盘，Windows 调色盘管理器含有两个发送给主视窗的讯息。

第一个是 QM_QUERYNEWPALETTE。当主视窗活动时，该讯息发送给主视窗。如果程式在您的视窗上绘画时使用了调色盘管理器，则它必须处理该讯息。GRAYS2 展示具体的作法。程式获得装置内容代号，并选进调色盘，呼叫 RealizePalette，然後使视窗失效以产生 WM_PAINT 讯息。如果显现了逻辑调色盘，则视窗讯息处理程式从该讯息传回 TRUE，否则传回 FALSE。

当系统调色盘改成与 WM_QUERYNEWPALETTE 讯息的结果相同时，Windows 将 WM_PALETTECHANGED 讯息发送给由目前活动的视窗来启动并终止处理视窗链的所有主视窗。这允许前台视窗有优先权。传递给视窗讯息处理程式的 wParam 值是活动视窗的代号。只有当 wParam 不等於程式的视窗代号时，使用调色盘管理器的程式才会处理该讯息。

通常，在处理 WM_PALETTECHANGED 时，使用自订调色盘的任何程式都呼叫 SelectPalette 和 RealizePalette。後续的视窗在讯息处理期间呼叫 RealizePalette 时，Windows 首先检查逻辑调色盘中的 RGB 颜色是否与已载入到系统调色盘中的 RGB 颜色相匹配。如果两个程式需要相同的颜色，那么这两个程式就共同使用一个系统调色盘项目。接下来，Windows 检查未使用的系统调色盘项目。如果都已使用，则逻辑调色盘中的颜色从 20 种保留项目映射到最近的颜色。

如果不关心程式非活动时显示区域的外观，那么您不必处理 WM_PALETTECHANGED 讯息。否则，您有两个选择。GRAYS2 显示其中之一：在处理 WM_QUERYNEWPALETTE 讯息时，它获得装置内容，选进调色盘，然後呼叫 RealizePalette。这时就可以在处理 WM_QUERYNEWPALETTE 时呼叫 InvalidateRect 了。相反地，GRAYS2 呼叫 UpdateColors。这个函式通常比重新绘制视窗更有效，同时它改变视窗中图素的值来帮助保护以前的颜色。

使用调色盘管理器的许多程式都将让 WM_QUERYNEWPALETTE 和 WM_PALETTECHANGED 讯息用 GRAYS2 所显示的方法来处理。

调色盘索引方法

程式 16-3 所示的 GRAYS3 程式与 GRAYS2 非常相似，只是在处理 WM_PAINT 期间使用了呼叫 PALETTEINDEX 的巨集，而不是 PALETTERGB。

程式 16-3　GRAYS3

```
GRAYS3.C
/*------------------------------------------------------------------------
-
```

```
      GRAYS3.C --      Gray Shades Using Palette Manager
                                              (c) Charles Petzold, 1998
------------------------------------------------------------------------
-*/

#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                    PSTR    szCmdLine,    int
iCmdShow)
{
      static TCHAR szAppName[] = TEXT ("Grays3") ;
      HWND                                  hwnd ;
      MSG                           msg ;
      WNDCLASS                            wndclass ;
      wndclass.style                      = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc                = WndProc ;
      wndclass.cbClsExtra                 = 0 ;
      wndclass.cbWndExtra                 = 0 ;
      wndclass.hInstance                  = hInstance ;
      wndclass.hIcon                      = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                    = LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground      = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
      wndclass.lpszMenuName       = NULL ;
      wndclass.lpszClassName      = szAppName ;

      if (!RegisterClass (&wndclass))
     {
                        MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                              szAppName, MB_ICONERROR) ;
                  return 0 ;
      }

      hwnd = CreateWindow (  szAppName, TEXT ("Shades of Gray #3"),
                             WS_OVERLAPPEDWINDOW,
                             CW_USEDEFAULT, CW_USEDEFAULT,
                             CW_USEDEFAULT, CW_USEDEFAULT,
                             NULL, NULL, hInstance, NULL) ;

      ShowWindow (hwnd, iCmdShow) ;
      UpdateWindow (hwnd) ;

      while (GetMessage (&msg, NULL, 0, 0))
      {
                        TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
      }
```

```
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (   HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
     static HPALETTE        hPalette ;
     static int             cxClient, cyClient ;
     HBRUSH                 hBrush ;
     HDC                    hdc ;
     int                    i ;
     LOGPALETTE                   *     plp ;
     PAINTSTRUCT            ps ;
     RECT                   rect ;
     switch (message)
    {
     case  WM_CREATE:
                    //    Set up a LOGPALETTE structure and create a palette

               plp = malloc (sizeof (LOGPALETTE) + 64 * sizeof (PALETTEENTRY)) ;

                     plp->palVersion        = 0x0300 ;
                     plp->palNumEntries     = 65 ;

                     for (i = 0 ; i < 65 ; i++)
                     {
                     plp->palPalEntry[i].peRed    = (BYTE) min (255, 4 * i) ;
                     plp->palPalEntry[i].peGreen  = (BYTE) min (255, 4 * i) ;
                     plp->palPalEntry[i].peBlue   = (BYTE) min (255, 4 * i) ;
                     plp->palPalEntry[i].peFlags  = 0 ;
                     }
               hPalette = CreatePalette (plp) ;
                     free (plp) ;
                     return 0 ;

     case  WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

     case  WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;

                    // Select and realize the palette in the device context

                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
```

```
                              // Draw the fountain of grays

               for (i = 0 ; i < 65 ; i++)
               {
                         rect.left        = i * cxClient / 64 ;
                         rect.top    = 0 ;
                         rect.right       = (i + 1) * cxClient / 64 ;
                         rect.bottom      = cyClient ;

                         hBrush = CreateSolidBrush (PALETTEINDEX (i)) ;

                         FillRect (hdc, &rect, hBrush) ;
                         DeleteObject (hBrush) ;
                    }

                    EndPaint (hwnd, &ps) ;
                    return 0 ;

     case  WM_QUERYNEWPALETTE:
                    if (!hPalette)
                                   return FALSE ;

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    InvalidateRect (hwnd, NULL, FALSE) ;

                    ReleaseDC (hwnd, hdc) ;
                    return TRUE ;

     case  WM_PALETTECHANGED:
                    if (!hPalette || (HWND) wParam == hwnd)
                                   break ;

                    hdc   = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    UpdateColors (hdc) ;

                    ReleaseDC (hwnd, hdc) ;
                    break ;

     case  WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

「调色盘」索引的颜色不同於调色盘 RGB 颜色，其高位元组是 1，而低位元组的值是目前在装置内容中选择的、逻辑调色盘中的索引。在 GRAYS3 中，逻辑调色盘有 65 个项目，用於这些项目的索引从 0 到 64。值

PALETTEINDEX (0)

指黑色，

PALETTEINDEX (32)

指灰色，而

PALETTEINDEX (64)

指白色。

因为 Windows 不需要执行最近颜色的搜索，所以使用调色盘索引比使用 RGB 值更有效。

查询调色盘支援

您可以容易地验证：当 Windows 在 16 位元或 24 位元显示模式下执行时，GRAYS2 和 GRAYS3 程式执行良好。但是在某些情况下，要使用调色盘管理器的 Windows 应用程式可能要先确定装置驱动程式是否支援它。这时，您可以呼叫 GetDeviceCaps，并以视讯显示的装置内容代号和 PASTERCAPS 作为参数。函式将传回由一系列旗标组成的整数。通过在传回值和常数 RC_PALETTE 之间执行位元操作来检验支援的调色盘：

```
RC_PALETTE & GetDeviceCaps (hdc, RASTERCAPS)
```

如果此值非零，则视讯显示器装置驱动程式将支援调色盘操作。在这种情况之下，来自 GetDeviceCaps 的其他三个重要项目也是可用的。函式呼叫

```
GetDeviceCaps (hdc, SIZEPALETTE)
```

将传回在显示卡上调色盘表的总尺寸。这与同时显示的颜色总数相同。因为调色盘管理器只用於每图素 8 位元的视讯显示模式，所以此值将是 256。

函式呼叫

```
GetDeviceCaps (hdc, NUMRESERVED)
```

传回在调色盘表中的颜色数，该表是装置驱动程式为系统保留的，此值是 20。不呼叫调色盘管理器，这些只是 Windows 应用程式在 256 色显示模式下使用的纯色。要使用其余的 236 种颜色，程式必须使用调色盘管理器函式。

一个附加项目也可用：

```
GetDeviceCaps (hdc, COLORRES)
```

此值告诉您载入到硬体调色盘表的 RGB 颜色值解析度（以位元计）。这些是进入数位类比转换器的位元。某些视讯显示卡只使用 6 位元 ADC，所以该值是 18。其余使用 8 位元的 ADC，所以值是 24。

Windows 程式注意颜色解析度并因此采取一些动作是很有用的。例如，如果该颜色解析度是 18，那么程式将不可能要求到 128 种灰阶，因为只有 64 个离散的灰阶可用。要求到 128 种灰阶就不必用多余的项目来填充硬体调色盘表。

系统调色盘

我在前面提过，Windows 系统调色盘直接与显示卡上的硬体调色盘查询表相符（然而，硬体调色盘查询表可能比系统调色盘的颜色解析度低）。程式可以通过呼叫下面的函式来获得系统调色盘中的某些或全部的 RGB 项目：

```
GetSystemPaletteEntries (hdc, uStart, uNum, &pe) ;
```

只有显示卡模式支援调色盘操作时，该函式才能执行。第二个和第三个参数是无正负号整数，显示第一个调色盘项目的索引和调色盘项目数。最後一个参数是指向 PALETTEENTRY 型态的指标。

您可以在几种情况下使用该函式。程式可以定义 PALETTEENTRY 结构如下：

```
PALETTEENTRY pe ;
```

然後可按下面的方法多次呼叫 GetSystemPaletteEntries：

```
GetSystemPaletteEntries (hdc, i, 1, &pe) ;
```

其中的 i 从 0 到某个值，该值小於从 GetDeviceCaps（带有 SIZEPALETTE 索引 255）传回的值。或者，程式要获得所有的系统调色盘项目，可以通过定义指向 PALETTEENTRY 结构的指标，然後重新配置足够的记忆体块，以储存与调色盘大小指定同样多的 PALETTEENTRY 结构。

GetSystemPaletteEntries 函式确实允许您检验硬体调色盘表。系统调色盘中的项目按图素值增加的顺序排列，这些值用於表示视讯显示缓冲区中的颜色。我将简单地讨论一下具体作法。

其他调色盘函式

我们在前面看过，Windows 程式能够改变系统调色盘，但只是间接改变：第一步建立逻辑调色盘，它基本上是程式要使用的 RGB 颜色值阵列。CreatePalette 函式不会导致系统调色盘或者显示卡调色盘表的任何变化。逻辑调色盘必须在任何事情发生之前就选进装置内容并显现。

程式可以通过呼叫

```
GetPaletteEntries (hPalette, uStart, uNum, &pe) ;
```

来查询逻辑调色盘中的 RGB 颜色值。您可以按使用 GetSystemPaletteEntries 的方法来使用此函式。但是要注意，第一个参数是逻辑调色盘的代号，而不是装置内容的代号。

建立逻辑调色盘以後，让您改变其中的值的相应函式是：

```
SetPaletteEntries (hPalette, uStart, uNum, &pe) ;
```

另外，记住呼叫此函式不引起系统调色盘的任何变化——即使目前调色盘选进了装置内容。此函式也不改变逻辑调色盘的尺寸。要改变逻辑调色盘的尺寸，请使用 ResizePalette。

下面的函式接受 RGB 颜色引用值作为最後的参数，并将索引传回给逻辑调色盘，该逻辑调色盘与和它最接近的 RGB 颜色值相对应：

```
iIndex = GetNearestPaletteIndex (hPalette, cr) ;
```

第二个参数是 COLORREF 值。如果希望的话，呼叫 GetPaletteEntries 就可以获得逻辑调色盘中实际的 RGB 颜色值。

如果程式在 8 位元显示模式下需要多於 236 种自订颜色，则可以呼叫 GetSystemPaletteUse。这允许程式设定 254 种自订颜色；系统仅保留黑色和白色。不过，程式仅在最大化充满全萤幕时才允许这样，而且它还将一些系统颜色设为黑色和白色，以便标题列和功能表等仍然可见。

位元映射操作问题

从第五章可以了解到，GDI 允许使用不同的「绘画模式」或「位元映射操作」来画线并填充区域。用 SetROP2 设定绘画模式，其中的「2」表示两个物件之间的二元（binary）位元映射操作。三元位元映射操作用於处理 BitBlt 和类似功能。这些位元映射操作决定了正在画的物件图素与表面图素的结合方式。例如，您可以画一条直线，以便线上的图素与显示的图素按位元异或的方式相结合。

位元映射操作就是在图素位元上照著各个位元的顺序进行操作。改变调色盘会影响到这些位元映射操作。位元映射操作的操作物件是图素位元，而这些图素位元可能与实际颜色没有关联。

透过执行 GRAYS2 或 GRAYS3 程式，您自己就可以得出这个结论。调整尺寸时，拖动顶部或底部的边界穿过视窗，Windows 利用反转背景图素位元的位元映射操作来显示拖动尺寸的边界，其目的是使拖动尺寸边界总是可见的。但在 GRAYS2 和 GRAYS3 程式中，您将看到各种随机变换的颜色，这些颜色恰好与对应於调色盘表中未使用的项目，那是反转显示图素位元的结果。可视颜色没有反转——只有图素位元反转了。

正如您在表 16-1 中所看到的一样，20 种标准保留颜色位於系统调色盘的顶部和底部，以便位元映射操作的结果仍然正常。然而，一旦您开始修改调色盘——尤其是替换了保留颜色——那么颜色物件的位元映射操作就变得没有意义了。

唯一保证的是位元映射操作将用黑色和白色运作。黑色是系统调色盘中的第一个项目（所有的图素位元都设为 0），而白色是最後的项目（所有的图素位

元都设为 1）。这两个项目不能改变。如果需要预知在颜色物件上进行位元映射操作的结果，则可以先获得系统调色盘表，然後查看不同图素位元值的 RGB 颜色值。

查看系统调色盘

在 Windows 下执行的程式将处理逻辑调色盘，为使逻辑调色盘更好地服务於所有使用逻辑调色盘的程式，Windows 将在系统调色盘中设定颜色。该系统调色盘复制了显示卡的硬体对照表内容。这样，查看系统调色盘有助於调适调色盘应用程式。

因为对於这个问题有三种截然不同的处理方式，所以我将向您展示三个程式，以显示系统调色盘的内容。

SYSPAL1 程式，如程式 16-4 所示，使用了前面所讲的 GetSystemPaletteEntries 函式。

程式 16-4 SYSPAL1

```
SYSPAL1.C
/*------------------------------------------------------------------------
-
   SYSPAL1.C --   Displays system palette
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------------
-*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("SysPal1") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                    PSTR    szCmdLine,    int
iCmdShow)
{
      HWND                  hwnd ;
      MSG                   msg ;
      WNDCLASS              wndclass ;
      wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc              = WndProc ;
      wndclass.cbClsExtra               = 0 ;
      wndclass.cbWndExtra               = 0 ;
      wndclass.hInstance                = hInstance ;
      wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
```

```
     wndclass.hbrBackground          = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName             = NULL ;
     wndclass.lpszClassName            = szAppName ;

     if (!RegisterClass (&wndclass))
     {
               MessageBox (     NULL, TEXT ("This program requires Windows
NT!"),
                                        szAppName, MB_ICONERROR) ;
                    return 0 ;
     }

     hwnd = CreateWindow (  szAppName, TEXT ("System Palette #1"),
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

     if (!hwnd)
                    return 0 ;
     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
     HDC   hdc ;
     int   iPalSize ;

     hdc = GetDC (hwnd) ;
     iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
     ReleaseDC (hwnd, hdc) ;

     if (iPalSize != 256)
     {
          MessageBox (hwnd,TEXT ("This program requires that the video ")
                    TEXT ("display mode have a 256-color palette."),
                              szAppName, MB_ICONERROR) ;
               return FALSE ;
     }
     return TRUE ;
```

```
}

LRESULT CALLBACK WndProc (   HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
      static int             cxClient, cyClient ;
      static SIZE            sizeChar ;
      HDC              hdc ;
      HPALETTE               hPalette ;
      int              i, x, y ;
      PAINTSTRUCT            ps ;
      PALETTEENTRY           pe [256] ;
      TCHAR                                  szBuffer [16] ;

      switch (message)
      {
      case WM_CREATE:
                        if (!CheckDisplay (hwnd))
                                        return -1 ;
                        hdc = GetDC (hwnd) ;
                        SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;
                        GetTextExtentPoint32  (hdc,  TEXT  ("FF-FF-FF"),  10,
&sizeChar) ;
                        ReleaseDC (hwnd, hdc) ;
                        return 0 ;

      case  WM_DISPLAYCHANGE:
                        if (!CheckDisplay (hwnd))
                                        DestroyWindow (hwnd) ;

                        return 0 ;

      case  WM_SIZE:
                        cxClient = LOWORD (lParam) ;
                        cyClient = HIWORD (lParam) ;
                        return 0 ;

      case  WM_PAINT:
                        hdc = BeginPaint (hwnd, &ps) ;
                        SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;

                        GetSystemPaletteEntries (hdc, 0, 256, pe) ;

                        for (i = 0, x = 0, y = 0 ; i < 256 ; i++)
                        {
                        wsprintf ( szBuffer, TEXT ("%02X-%02X-%02X"),
                pe[i].peRed, pe[i].peGreen, pe[i].peBlue) ;
```

```
                    TextOut (hdc, x, y, szBuffer, lstrlen (szBuffer)) ;

                    if (( x += sizeChar.cx) + sizeChar.cx > cxClient)
                                        {
                                        x = 0 ;

                                        if (( y += sizeChar.cy) > cyClient)
                                                    break ;
                                        }
                    }
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

     case  WM_PALETTECHANGED:
                    InvalidateRect (hwnd, NULL, FALSE) ;
                    return 0 ;

     case  WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

与 SYSPAL 系列中的其他程式一样，除非带有 SIZEPALETTE 参数的 GetDeviceCaps 传回值为 256，否则 SYSPAL1 不会执行。

注意无论 SYSPAL1 的显示区域什么时候收到 WM_PALETTECHANGED 讯息，它都是无效的。在合并 WM_PAINT 讯息处理期间，SYSPAL1 呼叫 GetSystemPaletteEntries，并用一个含 256 个 PALETTEENTRY 结构的阵列作为参数。RGB 值作为文字字串显示在显示区域。程式执行时，注意 20 种保留颜色是 RGB 值列表中的前 10 个和後 10 个，这与表 16-1 所示相同。

当 SYSPAL1 显示有用的资讯时，它与实际看到的 256 种颜色不同。那就是 SYSPAL2 的作业，如程式 16-5 所示。

程式 16-5　SYSPAL2

```
SYSPAL2.C
/*------------------------------------------------------------------------
-
   SYSPAL2.C --   Displays system palette
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------
*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
```

```
TCHAR szAppName [] = TEXT ("SysPal2") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND                        hwnd ;
     MSG                         msg ;
     WNDCLASS               wndclass ;

     wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                         =        LoadIcon        (NULL,
IDI_APPLICATION) ;
     wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground       = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName        = NULL ;
     wndclass.lpszClassName       = szAppName ;

     if (!RegisterClass (&wndclass))
     {
                    MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                                          szAppName,
MB_ICONERROR) ;
                    return 0 ;
    }

     hwnd = CreateWindow (  szAppName, TEXT ("System Palette #2"),
                            WS_OVERLAPPEDWINDOW,
                            CW_USEDEFAULT, CW_USEDEFAULT,
                            CW_USEDEFAULT, CW_USEDEFAULT,
                            NULL, NULL, hInstance, NULL) ;
     if (!hwnd)
                    return 0 ;
     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}
```

```
BOOL CheckDisplay (HWND hwnd)
{
      HDC hdc ;
      int iPalSize ;

      hdc = GetDC (hwnd) ;
      iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
      ReleaseDC (hwnd, hdc) ;

      if (iPalSize != 256)
      {
      MessageBox (hwnd, TEXT ("This program requires that the video ")
                  TEXT ("display mode have a 256-color palette."),
                              szAppName, MB_ICONERROR) ;
                  return FALSE ;
      }
      return TRUE ;
}

LRESULT CALLBACK WndProc (   HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
      static HPALETTE        hPalette ;
      static int             cxClient, cyClient ;
      HBRUSH                 hBrush ;
      HDC                    hdc ;
      int                    i, x, y ;
      LOGPALETTE                   *     plp ;
      PAINTSTRUCT            ps ;
      RECT                   rect ;

      switch (message)
     {
      case  WM_CREATE:
                        if (!CheckDisplay (hwnd))
                                          return -1 ;

                  plp = malloc (sizeof (LOGPALETTE) + 255 * sizeof (PALETTEENTRY)) ;

                  plp->palVersion              = 0x0300 ;
                  plp->palNumEntries           = 256 ;

                  for (i = 0 ; i < 256 ; i++)
                  {
                              plp->palPalEntry[i].peRed         = i ;
                              plp->palPalEntry[i].peGreen       = 0 ;
                                          plp->palPalEntry[i].peBlue   = 0 ;
```

```
                                        plp->palPalEntry[i].peFlags  =
PC_EXPLICIT ;
                        }

                    hPalette = CreatePalette (plp) ;
                    free (plp) ;
                    return 0 ;

     case  WM_DISPLAYCHANGE:
                        if (!CheckDisplay (hwnd))
                                        DestroyWindow (hwnd) ;

                        return 0 ;

     case  WM_SIZE:
                        cxClient = LOWORD (lParam) ;
                        cyClient = HIWORD (lParam) ;
                        return 0 ;

     case  WM_PAINT:
                        hdc = BeginPaint (hwnd, &ps) ;

                        SelectPalette (hdc, hPalette, FALSE) ;
                        RealizePalette (hdc) ;

                        for (y = 0 ; y < 16 ; y++)
                        for (x = 0 ; x < 16 ; x++)
                        {
                                    hBrush = CreateSolidBrush (PALETTEINDEX (16
* y + x)) ;
                                    SetRect (&rect, x      * cxClient /16, y*
cyClient / 16,
                         (x + 1)`   *     cxClient    /     16,(y+1)    *
cyClient / 16);
                                    FillRect (hdc, &rect, hBrush) ;
                                    DeleteObject (hBrush) ;
                        }
                        EndPaint (hwnd, &ps) ;
                        return 0 ;

     case  WM_PALETTECHANGED:
                        if ((HWND) wParam != hwnd)
                                        InvalidateRect (hwnd, NULL, FALSE) ;

                        return 0 ;

     case  WM_DESTROY:
                        DeleteObject (hPalette) ;
```

```
                    PostQuitMessage (0) ;
                    return 0 ;
    }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SYSPAL2 在 WM_CREATE 讯息处理期间建立了逻辑调色盘。但是请注意：逻辑调色盘中所有的 256 个值都是从 0 到 255 的调色盘索引，并且 peFlags 栏位是 PC_EXPLICIT。该旗标是这样定义的：「逻辑调色盘项目的较低字组指定了一个硬体调色盘索引。此旗标允许应用程式显示硬体调色盘的内容。」该旗标就是专为我们要做的这件事情而设计的。

在 WM_PAINT 讯息处理期间，SYSPAL2 将该调色盘选进装置内容并显现它。这不会引起系统调色盘的任何重组，而是允许程式使用 PALETTEINDEX 巨集来指定系统调色盘中的颜色。按此方法，SYSPAL2 显示了 256 个矩形。另外，当您执行该程式时，注意顶行和底行的前 10 种和後 10 种颜色是 20 种保留颜色，如表 16-1 所示。当您执行使用自己逻辑调色盘的程式时，显示就改变了。

如果您既喜欢看 SYSPAL2 中的颜色，又喜欢 RGB 的值，那么请与第八章的 WHATCLR 程式同时执行。

SYSPAL 系列中的第三版使用的技术对我来说是最近才出现的——从我开始研究 Windows 调色盘管理器七年多後，才出现了那些技术。

事实上，所有的 GDI 函式都直接或间接地指定颜色作为 RGB 值。在 GDI 内部，这将转换成与那个颜色相关的图素位元。在某些显示模式中（例如，16 位元或 24 位元颜色模式），这些转换是相当直接的。在其他显示模式中（4 位元或 8 位元颜色），这可能涉及最接近颜色的搜索。

然而，有两个 GDI 函式让您直接指定图素位元中的颜色。当然在这种方式中使用的这两个函式都与设备高度相关。它们太依赖设备了，以至於它们可以直接显示视讯显示卡上实际的调色盘对照表。这两个函式是 BitBlt 和 StretchBlt。

程式 16-6 所示的 SYSPAL3 程式显示了使用 StretchBlt 显示系统调色盘中颜色的方法。

程式 16-6　SYSPAL3

```
SYSPAL3.C
/*------------------------------------------------------------------------
-
      SYSPAL3.C --      Displays system palette
                                             (c) Charles Petzold, 1998
------------------------------------------------------------------------
*/
```

```
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName [] = TEXT ("SysPal3") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                    PSTR    szCmdLine,    int
iCmdShow)
{
      HWND                  hwnd ;
      MSG                   msg ;
      WNDCLASS         wndclass ;

      wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc              = WndProc ;
      wndclass.cbClsExtra               = 0 ;
      wndclass.cbWndExtra               = 0 ;
      wndclass.hInstance                = hInstance ;
      wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground      = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
      wndclass.lpszMenuName       = NULL ;
      wndclass.lpszClassName      = szAppName ;
      if (!RegisterClass (&wndclass))
      {
                  MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                                          szAppName,
MB_ICONERROR) ;
                return 0 ;
      }

      hwnd = CreateWindow (  szAppName, TEXT ("System Palette #3"),
                  WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;

      if (!hwnd)
                  return 0 ;
      ShowWindow (hwnd, iCmdShow) ;
      UpdateWindow (hwnd) ;

      while (GetMessage (&msg, NULL, 0, 0))
      {
                  TranslateMessage (&msg) ;
                  DispatchMessage (&msg) ;
      }
      return msg.wParam ;
}
```

```
BOOL CheckDisplay (HWND hwnd)
{
      HDC hdc ;
      int iPalSize ;

      hdc = GetDC (hwnd) ;
      iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
      ReleaseDC (hwnd, hdc) ;

      if (iPalSize != 256)
      {
      MessageBox (hwnd,TEXT("This program requires that the video ")
                  TEXT("display mode have a 256-color palette."),
                           szAppName, MB_ICONERROR) ;
                     return FALSE ;
      }
      return TRUE ;
}

LRESULT CALLBACK WndProc (   HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
      static HBITMAP hBitmap ;
      static int                  cxClient, cyClient ;
      BYTE                        bits [256] ;
      HDC                         hdc, hdcMem ;
      int                         i ;
      PAINTSTRUCT                 ps ;

      switch (message)
      {
      case  WM_CREATE:
                        if (! CheckDisplay (hwnd))
                                                return -1 ;

                        for ( i = 0 ; i < 256 ; i++)
                                                bits [i] = i ;

                        hBitmap = CreateBitmap (16, 16, 1, 8, &bits) ;
                        return 0 ;

      case  WM_DISPLAYCHANGE:
                        if (!CheckDisplay (hwnd))
                                        DestroyWindow (hwnd) ;

                        return 0 ;
```

```
     case  WM_SIZE:
                       cxClient = LOWORD (lParam) ;
                       cyClient = HIWORD (lParam) ;
                       return 0 ;

     case  WM_PAINT:
                       hdc = BeginPaint (hwnd, &ps) ;

                       hdcMem = CreateCompatibleDC (hdc) ;
                       SelectObject (hdcMem, hBitmap) ;

                       StretchBlt (hdc,    0, 0, cxClient, cyClient,
                                     hdcMem, 0, 0, 16, 16, SRCCOPY) ;

                       DeleteDC (hdcMem) ;
                       EndPaint (hwnd, &ps) ;
                       return 0 ;
     case  WM_DESTROY:
                       DeleteObject (hBitmap) ;
                       PostQuitMessage (0) ;
                       return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

在 WM_CREATE 讯息处理期间，SYSPAL3 使用 CreateBitmap 来建立 16 16 的每图素 8 位元的点阵图。该函式的最後一个参数是包括数值 0 到 255 的 256 位元组阵列。这些是 256 种可能的图素位元值。在处理 WM_PAINT 讯息的程序中，程式将这个点阵图选进记忆体装置内容，用 StretchBlt 来显示并填充该显示区域。Windows 仅将点阵图中的图素位元传输到视讯显示器硬体，从而允许这些图素位元存取调色盘对照表中的 256 个项目。程式的显示区域甚至不必使接收 WM_PALETTECHANGED 讯息无效——对於对照表的任何修改都会立即影响到 SYSPAL3 的显示。

调色盘动画

在本节的标题中看到「动画」一词，并开始考虑萤幕周围执行的「电脑宠物」时，您的眼前可能会为之一亮。是的，您可以使用 Windows 调色盘管理器作一些动画，而且是有一定专业水平的动画。

通常，Windows 下的动画就是快速连续地显示一系列点阵图。调色盘动画与这种方法有很大的区别。您透过在萤幕上绘制您所需要的每件东西开始，然後您处理调色盘来改变这些物件的颜色，可能是画一些相对於萤幕背景来说是不可见的图像。您用这种方法就可以获得动画效果，而不必重画任何东西。调色

盘动画的速度是相当快的。

对於调色盘动画，最初的建立工作与我们前面看见的有些不同：对於动画期间要修改的每种 RGB 颜色值，PALETTEENTRY 结构的 peFlags 栏位必须设定为 PC_RESERVED。

通常，就像我们所看到的一样，在建立逻辑调色盘时，您将 peFlags 标记设为 0。这允许 GDI 将多个逻辑调色盘中同样的颜色映射到相同的系统调色盘项目。例如，假设两个 Windows 程式都建立了包含 RGB 项目 10-10-10 的逻辑调色盘，那么在系统调色盘表中，Windows 只需要一个 10-10-10 项目。但如果这两个程式中的一个使用调色盘动画，那您就不要再让 GDI 使用调色盘了。调色盘动画意味著速度非常快——而且如果不重画，它也只可能提高速度。当使用调色盘动画的程式修改调色盘时，它不会影响其他程式，或者迫使 GDI 重组系统调色盘表。PC_RESERVED 的 peFlags 值为单个逻辑调色盘储存系统调色盘项目。

使用调色盘动画时，通常您可以在 WM_PAINT 讯息处理期间呼叫 SelectPalette 和 RealizePalette，使用 PALETTEINDEX 巨集来指定颜色。该巨集将一个索引带进逻辑调色盘表。

对於动画，您可能要通过改变调色盘来回应 WM_TIMER 讯息。要改变逻辑调色盘中的 RGB 颜色值，请使用一个 PALETTEENTRY 结构的阵列来呼叫函式 AnimatePalette。此函式速度很快，因为它只需要改变系统调色盘以及显示卡硬体调色盘表中的项目。

跳动的球

程式 16-7 显示了 BOUNCE 程式的元件，但还有一个程式可显示跳动的球。为了简单起见，根据显示区域的大小将球画成了椭圆形。因为本章有几个调色盘动画程式，所以 PALANIM.C（「调色盘动画」）档案包含一些通用内容。

程式 16-7　BOUNCE

```
PALANIM.C
/*-----------------------------------------------------------------------------
      PALANIM.C --      Palette Animation Shell Program
                                              s(c) Charles Petzold, 1998
-----------------------------------------------------------------------------*
/

#include <windows.h>
extern HPALETTE  CreateRoutine          (HWND) ;
extern void            PaintRoutine     (HDC, int, int) ;
extern void            TimerRoutine     (HDC, HPALETTE) ;
extern void            DestroyRoutine   (HWND, HPALETTE) ;
```

```
LRESULT    CALLBA CK   WndProc  (HWND, UINT, WPARAM, LPARAM) ;

extern TCHAR szAppName [] ;
extern TCHAR szTitle [] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                    PSTR    szCmdLine,    int
iCmdShow)
{
      HWND                         hwnd ;
      MSG                          msg ;
      WNDCLASS                wndclass ;
      wndclass.style                   = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc               = WndProc ;
      wndclass.cbClsExtra                = 0 ;
      wndclass.cbWndExtra                = 0 ;
      wndclass.hInstance                 = hInstance ;
      wndclass.hIcon                     = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                   = LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground       = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
      wndclass.lpszMenuName        = NULL ;
      wndclass.lpszClassName       = szAppName ;

      if (!RegisterClass (&wndclass))
      {
                        MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                                                  szAppName,
MB_ICONERROR) ;
                        return 0 ;
    }

      hwnd = CreateWindow (  szAppName, szTitle,
                      WS_OVERLAPPEDWINDOW,
                      CW_USEDEFAULT, CW_USEDEFAULT,
                      CW_USEDEFAULT, CW_USEDEFAULT,
                      NULL, NULL, hInstance, NULL) ;

      if (!hwnd)
                        return 0 ;
      ShowWindow (hwnd, iCmdShow) ;
      UpdateWindow (hwnd) ;

      while (GetMessage (&msg, NULL, 0, 0))
    {
                        TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
      }
```

```
     return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
     HDC hdc ;
     int iPalSize ;

     hdc = GetDC (hwnd) ;
     iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
     ReleaseDC (hwnd, hdc) ;

     if (iPalSize != 256)
     {
    MessageBox (hwnd, TEXT ("This program requires that the video ")
                TEXT ("display mode have a 256-color palette."),
                         szAppName, MB_ICONERROR) ;
               return FALSE ;
     }
     return TRUE ;
}

LRESULT CALLBACK WndProc (  HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
     static HPALETTE  hPalette ;
     static int                  cxClient, cyClient ;
     HDC                                   hdc ;
     PAINTSTRUCT                 ps ;

     switch (message)
     {
     case  WM_CREATE:
               if (!CheckDisplay (hwnd))
                                   return -1 ;

               hPalette = CreateRoutine (hwnd) ;
               return 0 ;

     case  WM_DISPLAYCHANGE:
               if (!CheckDisplay (hwnd))
                                   DestroyWindow (hwnd) ;

               return 0 ;

     case  WM_SIZE:
               cxClient = LOWORD (lParam) ;
               cyClient = HIWORD (lParam) ;
```

```
            return 0 ;

     case  WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;

            PaintRoutine (hdc, cxClient, cyClient) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;
     case  WM_TIMER:
            hdc = GetDC (hwnd) ;

            SelectPalette (hdc, hPalette, FALSE) ;

            TimerRoutine (hdc, hPalette) ;

                  ReleaseDC (hwnd, hdc) ;
                  return 0 ;

     case  WM_QUERYNEWPALETTE:
            if (!hPalette)
                        return FALSE ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            InvalidateRect (hwnd, NULL, TRUE) ;

            ReleaseDC (hwnd, hdc) ;
            return TRUE ;

     case  WM_PALETTECHANGED:
                  if (!hPalette || (HWND) wParam == hwnd)
                                    break ;

                  hdc = GetDC (hwnd) ;
                  SelectPalette (hdc, hPalette, FALSE) ;
                  RealizePalette (hdc) ;
                  UpdateColors (hdc) ;

                  ReleaseDC (hwnd, hdc) ;
                  break ;

     case  WM_DESTROY:
                  DestroyRoutine (hwnd, hPalette) ;
```

```
                        PostQuitMessage (0) ;
                        return 0 ;
      }
      return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

BOUNCE.C

```
/*------------------------------------------------------------------------
      BOUNCE.C --       Palette Animation Demo
                                              (c) Charles Petzold, 1998
------------------------------------------------------------------------
*/

#include <windows.h>
#define ID_TIMER 1
TCHAR szAppName  [] = TEXT ("Bounce") ;
TCHAR szTitle    [] = TEXT ("Bounce: Palette Animation Demo") ;

static LOGPALETTE * plp ;
HPALETTE CreateRoutine (HWND hwnd)
{
      HPALETTE          hPalette ;
      int               i ;

      plp = malloc (sizeof (LOGPALETTE) + 33 * sizeof (PALETTEENTRY)) ;
      plp->palVersion        = 0x0300 ;
      plp->palNumEntries     = 34 ;

      for (i = 0 ; i < 34 ; i++)
      {
                        plp->palPalEntry[i].peRed    = 255 ;
                        plp->palPalEntry[i].peGreen  = (i == 0 ? 0 : 255) ;
                        plp->palPalEntry[i].peBlue   = (i == 0 ? 0 : 255) ;
                        plp->palPalEntry[i].peFlags  = (i == 33 ? 0 : PC_RESERVED) ;
    }
      hPalette = CreatePalette (plp) ;
      SetTimer (hwnd, ID_TIMER, 50, NULL) ;
      return hPalette ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
      HBRUSH      hBrush ;
      int         i, x1, x2, y1, y2 ;
      RECT        rect ;
                              // Draw window background using palette index 33

      SetRect (&rect, 0, 0, cxClient, cyClient) ;
      hBrush = CreateSolidBrush (PALETTEINDEX (33)) ;
```

```
     FillRect (hdc, &rect, hBrush) ;
     DeleteObject (hBrush) ;

                    // Draw the 33 balls
     SelectObject (hdc, GetStockObject (NULL_PEN)) ;
     for (i = 0 ; i < 33 ; i++)
     {
                    x1 =  i * cxClient / 33 ;
                    x2 = (i + 1)* cxClient / 33 ;

                    if (i < 9)
                    {
                              y1  = i * cyClient / 9 ;
                              y2 = (i + 1) * cyClient / 9 ;
                    }
               else if (i < 17)
                         {
                              y1 = (16 - i) * cyClient / 9 ;
                              y2 = (17 - i) * cyClient / 9 ;
                         }
               else if (i < 25)
               {
                              y1 = (i - 16) * cyClient / 9 ;
                              y2 = (i - 15) * cyClient / 9 ;
               }
               else
               {
                              y1 = (32 - i) * cyClient / 9 ;
                              y2 = (33 - i) * cyClient / 9 ;
               }

               hBrush = CreateSolidBrush (PALETTEINDEX (i)) ;
               SelectObject (hdc, hBrush) ;
               Ellipse (hdc, x1, y1, x2, y2) ;
               DeleteObject     (SelectObject     (hdc,     GetStockObject
(WHITE_BRUSH))) ;
     }
     return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
     static BOOL bLeftToRight = TRUE ;
     static int  iBall ;

               // Set old ball to white
     plp->palPalEntry[iBall].peGreen = 255 ;
     plp->palPalEntry[iBall].peBlue  = 255 ;
```

```
      iBall += (bLeftToRight ? 1 : -1) ;
      if (  iBall == (bLeftToRight ? 33 : -1))
      {
                        iBall = (bLeftToRight ? 31 : 1) ;
                        bLeftToRight ^= TRUE ;
      }

                  // Set new ball to red

      plp->palPalEntry[iBall].peGreen = 0 ;
      plp->palPalEntry[iBall].peBlue  = 0 ;

                  // Animate the palette

      AnimatePalette (hPalette, 0, 33, plp->palPalEntry) ;
      return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
      KillTimer (hwnd, ID_TIMER) ;
      DeleteObject (hPalette) ;
      free (plp) ;
      return ;
}
```

除非 Windows 处於支援调色盘的显示模式下，否则调色盘动画将不能工作。因此，PALANIM.C 通过呼叫 CheckDisplay 函式（与 SYSPAL 程式中的函式相同）来开始处理 WM_CREATE。

PALANIM.C 呼叫 BOUNCE.C 中的四个函式：在 WM_CREATE 讯息处理期间呼叫 CreateRoutine（在 BOUNCE 中用於建立逻辑调色盘）；在 WM_PAINT 讯息处理期间呼叫 PaintRoutine；在 WM_TIMER 讯息处理期间呼叫 TimerRoutine；在 WM_DESTROY 讯息处理期间呼叫 DestroyRoutine（在 BOUNCE 中用於清除）。在呼叫 PaintRoutine 和 TimerRoutine 之前，PALANIM.C 获得装置内容，并将其选进逻辑调色盘。在呼叫 PaintRoutine 之前，它也显现调色盘。PALANIM.C 期望 TimerRoutine 呼叫 AnimatePalette。尽管 AnimatePalette 需要从装置内容中选择调色盘，但它不需要呼叫 RealizePalette。

BOUNCE 中的球按「W」路线在显示区域中来回跳动。显示区域背景是白色，球是红色。任何时候，都可以在 33 个不重叠的位置之一看见球。这需要 34 个调色盘项目：一个用於背景，其他 33 个用於不同位置的球。在 CreateRoutine 中，BOUNCE 初始化 PALETTEENTRY 结构的一个阵列，将第一个调色盘项目（与球在左上角的位置对应）设定为红色，其他的设定为白色。注意，对於除背景以

外的所有项目，peFlags 栏位都设定为 PC_RESERVED（背景是最後的一个调色盘项目）。BOUNCE 通过将 Windows 计时器的间隔设定为 50 毫秒来终止 CreateRoutine。

BOUNCE 在 PaintRoutine 完成所有的绘画工作。视窗背景用一个实心画刷和调色盘索引 33 所指定的颜色来绘制。33 个球的颜色是依据从 0 到 32 的调色盘索引的颜色。当 BOUNCE 第一次在显示区域内绘画时，0 的调色盘索引映射成红色，其他调色盘索引映射到白色。这导致球出现在左上角。

当 WndProc 处理 WM_TIMER 讯息并呼叫 TimerRoutine 时，动画就发生了。TimerRoutine 通过呼叫 AnimatePalette 来结束，语法如下：

```
AnimatePalette (hPalette, uStart, uNum, &pe) ;
```

其中，第一个参数是调色盘代号，最後一个参数是指向阵列的指标，该阵列由一个或多个 PALETTEENTRY 结构组成。该函式改变逻辑调色盘中从 uStart 项目到 uNum 项目之间的若干项目。逻辑调色盘中新的 uStart 项目是 PALETTEENTRY 结构中的第一个成员。当心！uStart 参数是进入原始逻辑调色盘表的索引，而不是进入 PALETTEENTRY 阵列的索引。

为了方便起见，BOUNCE 使用 PALETTEENTRY 结构的阵列，该结构是建立逻辑调色盘时使用的 LOGPALETTE 结构的一部分。球的目前位置（从 0 到 32）储存在静态变数 iBall 中。在 TimerRoutine 期间，BOUNCE 将 PALETTEENTRY 成员设为白色。然後计算球的下一个位置，并将该元素设为红色。用下面的呼叫来改变调色盘：

```
AnimatePalette (hPalette, 0, 33, plp->palPalEntry) ;
```

GDI 改变 33 逻辑调色盘项目中的第一个（尽管实际上只改变了两个），使它与系统调色盘表中的变化相对应，然後修改显示卡上的硬体调色盘表。这样，不用重画球就开始移动了。

BOUNCE 执行时，您会发现同时执行 SYSPAL2 或 SYSPAL3 效果会更好。

尽管 AnimatePalette 执行得非常快，但是当只有一两个项目改变时，您还应该尽量避免改变所有的逻辑调色盘项目。这在 BOUNCE 中有点复杂，因为球要来回地跳——iBall 要先增加，然後再减少。一种方法是使用两个变数：分别称为 iBallOld（设定球的目前位置）和 iBallMin（iBall 和 iBallOld 中较小的）。然後您就可以像下面这样呼叫 AnimatePalette 来改变两个项目了：

```
iBallMin = min (iBall, iBallOld) ;
AnimatePalette (hPal, iBallMin, 2, plp->palPalEntry + iBallMin) ;
```

还有另一种方法：我们先假定您定义了一个 PALETTEENTRY 结构：

```
PALETTEENTRY pe ;
```

在 TimerRoutine 期间，您将 PALETTEENTRY 栏位设为白色，并呼叫 AnimatePalette 来改变逻辑调色盘中 iBall 位置的一个项目：

```
pe.peRed         = 255 ;
      pe.peGreen       = 255 ;
      pe.peBlue        = 255 ;
      pe.peFlags       = PC_RESERVED ;
      AnimatePalette (hPalette, iBall, 1, &pe) ;
```

然後计算显示在 BOUNCE 中的 iBall 的新值，将 PALETTEENTRY 结构的栏位定义为红色，然後再次呼叫 AnimatePalette：

```
pe.peRed         = 255 ;
      pe.peGreen       = 0 ;
      pe.peBlue        = 0 ;
      pe.peFlags       = PC_RESERVED ;
      AnimatePalette (hPalette, iBall, 1, &pe) ;
```

尽管跳动的球是对动画的一个传统的简单说明，但它实际上并不适合调色盘动画，因为必须先画出球的所有可能位置。调色盘动画更适合於显示运动的重复图案。

一个项目的调色盘动画

调色盘动画中一个更有趣的方面就是，可以只使用一个调色盘项目来完成一些有趣的技术。例如程式 16-8 所示的 FADER 程式。这个程式也需要前面的 PALANIM.C 档案。

程式 16-8 FADER

```
FADER.C
/*------------------------------------------------------------------------
      FADER.C --  Palette Animation Demo
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
#define ID_TIMER 1
TCHAR szAppName  []    = TEXT ("Fader") ;
TCHAR szTitle    []    = TEXT ("Fader: Palette Animation Demo") ;

static LOGPALETTE lp ;
HPALETTE CreateRoutine (HWND hwnd)
{
      HPALETTE hPalette ;
      lp.palVersion                    = 0x0300 ;
      lp.palNumEntries                 = 1 ;
      lp.palPalEntry[0].peRed          = 255 ;
      lp.palPalEntry[0].peGreen        = 255 ;
      lp.palPalEntry[0].peBlue         = 255 ;
      lp.palPalEntry[0].peFlags        = PC_RESERVED ;
```

```
      hPalette = CreatePalette (&lp) ;
      SetTimer (hwnd, ID_TIMER, 50, NULL) ;
      return hPalette ;
}

void  PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
      static TCHAR           szText [] = TEXT (" Fade In and Out ") ;
      int                              x, y ;
      SIZE                                   sizeText ;

      SetTextColor (hdc, PALETTEINDEX (0)) ;
      GetTextExtentPoint32 (hdc, szText, lstrlen (szText), &sizeText) ;

      for (x = 0 ; x < cxClient ; x += sizeText.cx)
      for (y = 0 ; y < cyClient ; y += sizeText.cy)
      {
                  TextOut (hdc, x, y, szText, lstrlen (szText)) ;
      }

      return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
      static BOOL bFadeIn = TRUE ;
      if (bFadeIn)
      {
                        lp.palPalEntry[0].peRed   -= 4 ;
                        lp.palPalEntry[0].peGreen -= 4 ;

                        if (  lp.palPalEntry[0].peRed == 3)
                                          bFadeIn = FALSE ;
      }
      else
      {
                        lp.palPalEntry[0].peRed   += 4 ;
                        lp.palPalEntry[0].peGreen += 4 ;

                        if     (lp.palPalEntry[0].peRed == 255)
                                          bFadeIn = TRUE ;
      }

      AnimatePalette (hPalette, 0, 1, lp.palPalEntry) ;
      return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
```

```
{
      KillTimer (hwnd, ID_TIMER) ;
      DeleteObject (hPalette) ;
      return ;
}
```

FADER 在显示区域上显示满了文字字串「Fade In And Out」。文字首先显示为白色，这对於白色背景的视窗来说是看不出来的。通过使用调色盘动画，FADER 慢慢地将文字的颜色改为蓝色，然後再改回白色，这样一遍一遍地重复。文字就有渐现渐隐的显示效果了。

FADER用CreateRoutine函式建立了逻辑调色盘,它只需要一个调色盘项目，并将颜色初始化为白色——红色、绿色和蓝色值都设为 255。在 PaintRoutine 中（您可能想起，当逻辑调色盘选进装置内容并显现以後，PALANIM 呼叫过此函式），FADER 呼叫 SetTextColor 将文字颜色设定为 PALETTEINDEX(0)。这意味著文字颜色设定为调色盘表格中的第一个项目，此项目初始为白色。然後 FADER 用「Fade In And Out」文字字串填充显示区域。这时，视窗背景是白色，文字也是白色，所以文字不可见。

在 TimerRoutine 函式中，FADER 通过改变 PALETTEENTRY 结构并将其传递给 AnimatePalette 来完成调色盘动画。最初，对每一个 WM_TIMER 讯息，程式都将红色和绿色值减 4，直到等於 3；然後将这些值加 4，直到等於 255。这将使文字颜色逐渐从白色变到蓝色，然後又回到白色。

程式 16-9 所示的 ALLCOLOR 程式只用了逻辑调色盘的一个项目来显示显示卡可以著色的所有颜色。当然，程式不是同时显示这些颜色，而是连续显示。如果显示卡有 18 位元的解析度（这时能有 262144 种不同的颜色），那么在两种颜色间隔 55 毫秒的速度下，只需要 4 小时就可以在萤幕上看到所有的颜色。

程式 16-9　ALLCOLOR

```
ALLCOLOR.C
/*------------------------------------------------------------------------
-
      ALLCOLOR.C --          Palette Animation Demo
                                                  (c) Charles Petzold, 1998
------------------------------------------------------------------------
-*/

#include <windows.h>
#define ID_TIMER    1

TCHAR szAppName  [] = TEXT ("AllColor") ;
TCHAR szTitle    [] = TEXT ("AllColor: Palette Animation Demo") ;

static     int                          iIncr ;
```

```
static     PALETTEENTRY     pe ;

HPALETTE CreateRoutine (HWND hwnd)
{
     HDC              hdc ;
     HPALETTE              hPalette ;
     LOGPALETTE       lp ;

                      // Determine the color resolution and set iIncr
     hdc = GetDC (hwnd) ;
     iIncr = 1 << (8 - GetDeviceCaps (hdc, COLORRES) / 3) ;
     ReleaseDC (hwnd, hdc) ;

                      // Create the logical palette
     lp.palVersion                    = 0x0300 ;
     lp.palNumEntries                   = 1 ;
     lp.palPalEntry[0].peRed            = 0 ;
     lp.palPalEntry[0].peGreen          = 0 ;
     lp.palPalEntry[0].peBlue           = 0 ;
     lp.palPalEntry[0].peFlags          = PC_RESERVED ;

     hPalette = CreatePalette (&lp) ;
                      // Save global for less typing
     pe = lp.palPalEntry[0] ;
     SetTimer (hwnd, ID_TIMER, 10, NULL) ;
     return hPalette ;
}

void DisplayRGB (HDC hdc, PALETTEENTRY * ppe)
{
     TCHAR szBuffer [16] ;
     wsprintf (szBuffer, TEXT (" %02X-%02X-%02X "),
                                        ppe->peRed,             ppe->peGreen,
ppe->peBlue) ;
     TextOut (hdc, 0, 0, szBuffer, lstrlen (szBuffer)) ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
     HBRUSH           hBrush ;
     RECT                  rect ;

                      // Draw Palette Index 0 on entire window

     hBrush = CreateSolidBrush (PALETTEINDEX (0)) ;
     SetRect (&rect, 0, 0, cxClient, cyClient) ;
     FillRect (hdc, &rect, hBrush) ;
     DeleteObject (SelectObject (hdc, GetStockObject (WHITE_BRUSH))) ;
```

```
                        // Display the RGB value
      DisplayRGB (hdc, &pe) ;
      return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
      static BOOL  bRedUp = TRUE, bGreenUp = TRUE, bBlueUp = TRUE ;
                                    // Define new color value
      pe.peBlue += (bBlueUp ? iIncr : -iIncr) ;
      if (  pe.peBlue == (BYTE) (bBlueUp ? 0 : 256 - iIncr))
      {
                  pe.peBlue = (bBlueUp ? 256 - iIncr : 0) ;
                        bBlueUp ^= TRUE ;
                        pe.peGreen += (bGreenUp ? iIncr : -iIncr) ;

                  if (  pe.peGreen == (BYTE) (bGreenUp ? 0 : 256 - iIncr))
            {
                              pe.peGreen = (bGreenUp ? 256 - iIncr : 0) ;
                                          bGreenUp ^= TRUE ;
                              pe.peRed += (bRedUp ? iIncr : -iIncr) ;

                                          if (  pe.peRed == (BYTE) (bRedUp ? 0 :
256 - iIncr))
                                          {
                                          pe.peRed = (bRedUp ? 256 - iIncr : 0) ;
                                          bRedUp ^= TRUE ;
                                          }
                        }
       }

                                    // Animate the palette
      AnimatePalette (hPalette, 0, 1, &pe) ;
      DisplayRGB (hdc, &pe) ;
      return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
      KillTimer (hwnd, ID_TIMER) ;
      DeleteObject (hPalette) ;
      return ;
}
```

在结构上，ALLCOLOR 与 FADER 非常相似。在 CreateRoutine 中，ALLCOLOR 只用一个设为黑色的调色盘项目（PALETTEENTRY 结构的 red、green 和 blue 栏位设为 0）来建立调色盘。在 PaintRoutine 中，ALLCOLOR 用 PALETTEINDEX(0)

建立实心画刷，并呼叫 FillRect 来用此画刷为整个显示区域著色。

在 TimerRoutine 中，ALLCOLOR 通过改变 PALETTEENTRY 颜色并呼叫 AnimatePalette 来启动调色盘。我编写 ALLCOLOR 程式，以便颜色变化顺畅。首先，蓝色值渐渐增加。达到最大时，绿色值增加，而蓝色值渐渐减少。红色、绿色和蓝色值的增加和减少取决於 iIncr 变数。在 CreateRoutine 期间，这将根据用 COLORRES 参数从 GetDeviceCaps 传回的值来计算。例如，如果 GetDeviceCaps 传回 18，那么 iIncr 设为 4——获得所有颜色所需要的最小值。

ALLCOLOR 还在显示区域的左上角显示目前的 RGB 颜色值。我最初添加这个程式码是出於测试目的，但是现在证明它是有用的，所以我保留了它。

工程应用程式

在工程应用程式中，动画对於显示机械或电的作用过程很有用。在电脑萤幕上显示内燃引擎虽然简单，但是动画可以使它变得更加生动，且更清楚地显示其工作程序。

使用调色盘动画的一个好范例就是显示流体通过管子的过程。这是一个例子，图像不必十分精确——实际上，如果图像很精确（就像看透明的管子），则很难说明管子里的流体是如何运动的。这时用符号会更好一些。程式 16-10 所示的 PIPES 程式是此技术的简单示范：在显示区域有两个水平的管子，流体在上面的管子里从左向右流动，而在下面的管子里从右向左移动。

程式 16-10 PIPES

PIPES.C

```
/*------------------------------------------------------------------------
      PIPES.C --       Palette Animation Demo
                                              (c) Charles Petzold, 1998
------------------------------------------------------------------------*
/

#include <windows.h>
#define ID_TIMER 1
TCHAR szAppName [] = TEXT ("Pipes") ;
TCHAR szTitle   [] = TEXT ("Pipes: Palette Animation Demo") ;

static LOGPALETTE * plp ;

HPALETTE CreateRoutine (HWND hwnd)
{
      HPALETTE         hPalette ;
      int              i ;
```

```
     plp = malloc (sizeof (LOGPALETTE) + 32 * sizeof (PALETTEENTRY)) ;
                          // Initialize the fields of the LOGPALETTE structure
     plp->palVersion        = 0x300 ;
     plp->palNumEntries     = 16 ;

     for (i = 0 ; i <= 8 ; i++)
     {
                    plp->palPalEntry[i].peRed    = (BYTE) min (255, 0x20 * i) ;
                    plp->palPalEntry[i].peGreen  = 0 ;
                    plp->palPalEntry[i].peBlue   = (BYTE) min (255, 0x20 * i) ;
                    plp->palPalEntry[i].peFlags  = PC_RESERVED ;

                    plp->palPalEntry[16 - i]     = plp->palPalEntry[i] ;
                    plp->palPalEntry[16 + i]     = plp->palPalEntry[i] ;
                    plp->palPalEntry[32 - i]     = plp->palPalEntry[i] ;
     }

     hPalette = CreatePalette (plp) ;
     SetTimer (hwnd, ID_TIMER, 100, NULL) ;
     return hPalette ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
     HBRUSH     hBrush ;
     int              i ;
     RECT             rect ;

                              // Draw window background

     SetRect (&rect, 0, 0, cxClient, cyClient) ;
     hBrush = SelectObject (hdc, GetStockObject (WHITE_BRUSH)) ;
     FillRect (hdc, &rect, hBrush) ;

                              // Draw the interiors of the pipes
     for (i = 0 ; i < 128 ; i++)
     {
                    hBrush = CreateSolidBrush (PALETTEINDEX (i % 16)) ;
                    SelectObject (hdc, hBrush) ;

                    rect.left              = (127 - i) * cxClient / 128 ;
                    zrect.right            = (128 - i) * cxClient / 128 ;
                    rect.top         = 4 * cyClient / 14 ;
                    rect.bottom            = 5 * cyClient / 14 ;

                    FillRect (hdc, &rect, hBrush) ;

                    rect.left              =     i * cxClient / 128 ;
```

```
                        rect.right              = (   i + 1)      *  cxClient  /
128 ;
                        rect.top          =     9     * cyClient / 14 ;
                        rect.bottom             =     10    * cyClient / 14 ;

                        FillRect (hdc, &rect, hBrush) ;

                  DeleteObject     (SelectObject     (hdc,     GetStockObject
(WHITE_BRUSH))) ;
     }

                                    // Draw the edges of the pipes
      MoveToEx          (hdc, 0,  4 * cyClient / 14, NULL) ;
      LineTo            (hdc, cxClient,  4      * cyClient / 14) ;

      MoveToEx          (hdc, 0,  5 * cyClient / 14, NULL) ;
      LineTo            (hdc, cxClient,  5      * cyClient / 14) ;

      MoveToEx          (hdc, 0,  9 * cyClient / 14, NULL) ;
      LineTo            (hdc, cxClient, 9 * cyClient / 14) ;

      MoveToEx          (hdc, 0,  10 * cyClient / 14, NULL) ;
      LineTo   (hdc, cxClient,            10 * cyClient / 14) ;
      return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
      static int iIndex ;
      AnimatePalette (hPalette, 0, 16, plp->palPalEntry + iIndex) ;
      iIndex = (iIndex + 1) % 16 ;

      return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
      KillTimer (hwnd, ID_TIMER) ;
      DeleteObject (hPalette) ;
      free (plp) ;
      return ;
}
```

PIPES 为动画使用了 16 个调色盘项目，而您可能会使用更少的项目。最小化时，真正需要的是有足够的项目来显示流动的方向。用三个调色盘项目要比用一个静态箭头好。

程式 16-11 所示的 TUNNEL 程式是这组程式中最贪心的程式，它为动画使用

了 128 个调色盘项目，但是从效果来看，值得这样做。

程式 16-11 TUNNEL

```
TUNNEL.C
/*------------------------------------------------------------------------
-
      TUNNEL.C --      Palette Animation Demo
                                              (c) Charles Petzold, 1998
------------------------------------------------------------------------
*/

#include <windows.h>
#define ID_TIMER 1
TCHAR szAppName  [] = TEXT ("Tunnel") ;
TCHAR szTitle    [] = TEXT ("Tunnel: Palette Animation Demo") ;

static LOGPALETTE * plp ;
HPALETTE CreateRoutine (HWND hwnd)
{
      BYTE             byGrayLevel ;
      HPALETTE         hPalette ;
      int              i ;

      plp = malloc (sizeof (LOGPALETTE) + 255 * sizeof (PALETTEENTRY)) ;
                       // Initialize the fields of the LOGPALETTE structure
      plp->palVersion        = 0x0300 ;
      plp->palNumEntries     = 128 ;

      for (i = 0 ; i < 128 ; i++)
      {
                       if (i < 64)
                                       byGrayLevel = (BYTE) (4 * i) ;
                       else
                                       byGrayLevel = (BYTE) min (255, 4 * (128
- i)) ;
                       plp->palPalEntry[i].peRed    = byGrayLevel ;
                       plp->palPalEntry[i].peGreen  = byGrayLevel ;
                       plp->palPalEntry[i].peBlue   = byGrayLevel ;
                       plp->palPalEntry[i].peFlags  = PC_RESERVED ;

                       plp->palPalEntry[i + 128].peRed    = byGrayLevel ;
                       plp->palPalEntry[i + 128].peGreen = byGrayLevel ;
                       plp->palPalEntry[i + 128].peBlue  = byGrayLevel ;
                       plp->palPalEntry[i + 128].peFlags = PC_RESERVED ;
      }

      hPalette = CreatePalette (plp) ;
      SetTimer (hwnd, ID_TIMER, 50, NULL) ;
```

```
     return hPalette ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
     HBRUSH     hBrush ;
     int              i ;
     RECT             rect ;

     for (i = 0 ; i < 127 ; i++)
    {
                                   // Use a RECT structure for each of 128
rectangles
                rect.left   =   i * cxClient / 255 ;
                     rect.top    =   i * cyClient / 255 ;
                     rect.right  = cxClient - i * cxClient / 255 ;
                     rect.bottom = cyClient - i * cyClient / 255 ;

                     hBrush = CreateSolidBrush (PALETTEINDEX (i)) ;
                                   // Fill the rectangle and delete the brush

                     FillRect (hdc, &rect, hBrush) ;
                     DeleteObject (hBrush) ;
     }
     return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
     static int iLevel ;
     iLevel = (iLevel + 1) % 128 ;
     AnimatePalette (hPalette, 0, 128, plp->palPalEntry + iLevel) ;
     return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
     KillTimer (hwnd, ID_TIMER) ;
     DeleteObject (hPalette) ;
     free (plp) ;
     return ;
}
```

TUNNEL 在 128 个调色盘项目中使用 64 种移动的灰阶——从黑到白，再从白到黑——表现在隧道旅行的效果。

调色盘和真实世界图像

当然，尽管我们已经完成了许多有趣的事：连续显示色彩的网底、做了调色盘动画，但调色盘管理器的真正目的是允许在 8 位元显示模式下显示真实世界中的图像。对於本章的其余部分，我们正好研究一下。正如您所期望的，在使用 packed DIB、GDI 点阵图物件和 DIB 区块时，必须按照不同的方法来使用调色盘。下面的六个程式阐明了用调色盘来处理点阵图的各种技术。

调色盘和 packed DIB

下面三个程式，有助於我们建立处理 packed DIB 记忆体块的一系列函式。这些函式都在程式 16-12 所示的 PACKEDIB 档案中。

程式 16-12　PACKEDIB 档案

```
PACKEDIB.H
/*------------------------------------------------------------------------
   PACKEDIB.H --  Header file for PACKEDIB.C
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

BITMAPINFO * PackedDibLoad (PTSTR szFileName) ;
int PackedDibGetWidth (BITMAPINFO * pPackedDib) ;
int PackedDibGetHeight (BITMAPINFO * pPackedDib) ;
int PackedDibGetBitCount (BITMAPINFO * pPackedDib) ;
int PackedDibGetRowLength (BITMAPINFO * pPackedDib) ;
int PackedDibGetInfoHeaderSize (BITMAPINFO * pPackedDib) ;
int PackedDibGetColorsUsed (BITMAPINFO * pPackedDib) ;
int PackedDibGetNumColors (BITMAPINFO * pPackedDib) ;
int PackedDibGetColorTableSize (BITMAPINFO * pPackedDib) ;
RGBQUAD * PackedDibGetColorTablePtr (BITMAPINFO * pPackedDib) ;
RGBQUAD * PackedDibGetColorTableEntry (BITMAPINFO * pPackedDib, int i) ;
BYTE * PackedDibGetBitsPtr (BITMAPINFO * pPackedDib) ;
int PackedDibGetBitsSize (BITMAPINFO * pPackedDib) ;
HPALETTE PackedDibCreatePalette (BITMAPINFO * pPackedDib) ;
PACKEDIB.C
/*------------------------------------------------------------------------
      PACKEDIB.C --          Routines for using packed DIBs
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*
/

#include <windows.h>

/*------------------------------------------------------------------------
```

```
-
      PackedDibLoad: Load DIB File as Packed-Dib Memory Block
------------------------------------------------------------------------
*/

BITMAPINFO * PackedDibLoad (PTSTR szFileName)
{
      BITMAPFILEHEADER       bmfh ;
      BITMAPINFO                   *     pbmi ;
      BOOL                   bSuccess ;
      DWORD                  dwPackedDibSize, dwBytesRead ;
      HANDLE                 hFile ;
                  // Open the file: read access, prohibit write access

      hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHARE_READ, NULL,
       OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
      if (hFile == INVALID_HANDLE_VALUE)
                        return NULL ;
                  // Read in the BITMAPFILEHEADER
      bSuccess = ReadFile (  hFile, &bmfh, sizeof (BITMAPFILEHEADER),
                        &dwBytesRead, NULL) ;

      if (!bSuccess     || (dwBytesRead != sizeof (BITMAPFILEHEADER))
                        || (bmfh.bfType != * (WORD *) "BM"))
      {
                        CloseHandle (hFile) ;
                        return NULL ;
      }

                        // Allocate memory for the packed DIB & read it in
      dwPackedDibSize = bmfh.bfSize - sizeof (BITMAPFILEHEADER) ;
      pbmi = malloc (dwPackedDibSize) ;
      bSuccess = ReadFile (hFile, pbmi, dwPackedDibSize, &dwBytesRead, NULL) ;
      CloseHandle (hFile) ;

      if (!bSuccess || (dwBytesRead != dwPackedDibSize))
      {
                        free (pbmi) ;
                        return NULL ;
      }

      return pbmi ;
}

/*------------------------------------------------------------------------
   Functions to get information from packed DIB
------------------------------------------------------------------------
*/
```

```
int PackedDibGetWidth (BITMAPINFO * pPackedDib)
{
     if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
                         return
((PBITMAPCOREINFO)pPackedDib)->bmciHeader.bcWidth ;
     else
                         return pPackedDib->bmiHeader.biWidth ;
}

int PackedDibGetHeight (BITMAPINFO * pPackedDib)
{
     if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
                         return
((PBITMAPCOREINFO)pPackedDib)->bmciHeader.bcHeight ;
     else
                         return abs (pPackedDib->bmiHeader.biHeight) ;
}

int PackedDibGetBitCount (BITMAPINFO * pPackedDib)
{
     if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
                         return
((PBITMAPCOREINFO)pPackedDib)->bmciHeader.bcBitCount ;
     else
                         return pPackedDib->bmiHeader.biBitCount ;
}

int PackedDibGetRowLength (BITMAPINFO * pPackedDib)
{
     return ((  PackedDibGetWidth (pPackedDib) *
                                        PackedDibGetBitCount (pPackedDib) + 31)
& ~31) >> 3 ;
}

/*-------------------------------------------------------------------------
-
     PackedDibGetInfoHeaderSize includes possible color masks!
-------------------------------------------------------------------------
*/

int PackedDibGetInfoHeaderSize (BITMAPINFO * pPackedDib)
{
     if (  pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
                         return
((PBITMAPCOREINFO)pPackedDib)->bmciHeader.bcSize ;

     else  if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPINFOHEADER))
```

```
                        return pPackedDib->bmiHeader.biSize +
                  (pPackedDib->bmiHeader.biCompression ==
                   BI_BITFIELDS ? 12 : 0) ;
      else return pPackedDib->bmiHeader.biSize ;
}

/*-------------------------------------------------------------------------
      PackedDibGetColorsUsed returns value in information header;
                        could be 0 to indicate non-truncated color table!
-------------------------------------------------------------------------
*/

int PackedDibGetColorsUsed (BITMAPINFO * pPackedDib)
{
      if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
                        return 0 ;
      else
                        return pPackedDib->bmiHeader.biClrUsed ;
}

/*-------------------------------------------------------------------------
--
      PackedDibGetNumColors is actual number of entries in color table
-------------------------------------------------------------------------
-*/

int PackedDibGetNumColors (BITMAPINFO * pPackedDib)
{
      int iNumColors ;
      iNumColors = PackedDibGetColorsUsed (pPackedDib) ;
      if (  iNumColors ==    0 && PackedDibGetBitCount (pPackedDib) < 16)
                         iNumColors =1 << PackedDibGetBitCount (pPackedDib) ;

      return iNumColors ;
}

int PackedDibGetColorTableSize (BITMAPINFO * pPackedDib)
{
      if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
                        return  PackedDibGetNumColors  (pPackedDib)  *  sizeof
(RGBTRIPLE) ;
      else
                        return  PackedDibGetNumColors  (pPackedDib)  *  sizeof
(RGBQUAD) ;
}

RGBQUAD * PackedDibGetColorTablePtr (BITMAPINFO * pPackedDib)
{
```

```
     if (PackedDibGetNumColors (pPackedDib) == 0)
                    return 0 ;
     return (RGBQUAD *) (((BYTE *)     pPackedDib) +
                     PackedDibGetInfoHeaderSize (pPackedDib)) ;
}

RGBQUAD * PackedDibGetColorTableEntry (BITMAPINFO * pPackedDib, int i)
{
     if    (     PackedDibGetNumColors (pPackedDib) == 0)
                    return 0 ;

     if    (     pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
                    return (RGBQUAD *)
                                 (((RGBTRIPLE  *)  PackedDibGetColorTablePtr
(pPackedDib)) + i) ;
     else
                return PackedDibGetColorTablePtr (pPackedDib) + i ;
}

/*-------------------------------------------------------------------------
   PackedDibGetBitsPtr finally!
-------------------------------------------------------------------------
*/

BYTE * PackedDibGetBitsPtr (BITMAPINFO * pPackedDib)
{
     return ((BYTE *) pPackedDib)+     PackedDibGetInfoHeaderSize
(pPackedDib) +
                                PackedDibGetColorTableSize (pPackedDib) ;
}

/*-------------------------------------------------------------------------
---
     PackedDibGetBitsSize can be calculated from the height and row length
                if it's not explicitly in the biSizeImage field
-------------------------------------------------------------------------
-*/

int PackedDibGetBitsSize (BITMAPINFO * pPackedDib)
{
          if ((pPackedDib->bmiHeader.biSize != sizeof (BITMAPCOREHEADER)) &&
                           (pPackedDib->bmiHeader.biSizeImage != 0))
                           return pPackedDib->bmiHeader.biSizeImage ;

          return     PackedDibGetHeight (pPackedDib) *
                                PackedDibGetRowLength (pPackedDib) ;
}
```

```
/*-------------------------------------------------------------------------
-
   PackedDibCreatePalette creates logical palette from PackedDib
-------------------------------------------------------------------------
-*/
HPALETTE PackedDibCreatePalette (BITMAPINFO * pPackedDib)
{
      HPALETTE   hPalette ;
      int       i, iNumColors ;
      LOGPALETTE  *     plp ;
      RGBQUAD    *      prgb ;

      if (0 == ( iNumColors = PackedDibGetNumColors (pPackedDib)))
                  return NULL ;
      plp = malloc (sizeof   (LOGPALETTE) *
       (iNumColors - 1) * sizeof (PALETTEENTRY)) ;

      plp->palVersion    = 0x0300 ;
      plp->palNumEntries = iNumColors ;
      for (i = 0 ; i < iNumColors ; i++)
      {
                        prgb = PackedDibGetColorTableEntry (pPackedDib, i) ;
                        plp->palPalEntry[i].peRed    = prgb->rgbRed ;
                        plp->palPalEntry[i].peGreen  = prgb->rgbGreen ;
                        plp->palPalEntry[i].peBlue   = prgb->rgbBlue ;
                        plp->palPalEntry[i].peFlags  = 0 ;
      }

      hPalette = CreatePalette (plp) ;
      free (plp) ;

      return hPalette ;
}
```

第一个函式是 PackedDibLoad，它将唯一的参数作为档案名，并传回指向记忆体中 packed DIB 的指标。其他所有函式都将这个 packed DIB 指标作为它们的第一个参数并传回有关 DIB 的资讯。这些函式按「由下而上」顺序排列到档案中。每个函式都使用从前面函式获得的资讯。

我不倾向於说这是在处理 packed DIB 时有用的「完整」函式集。而且，我也不想汇编一个真正的扩展集，因为我不认为这是处理 packed DIB 的一个好方法。在写类似下面的函式时，您会很明显地发现这一点：

```
dwPixel = PackedDibGetPixel (pPackedDib, x, y) ;
```

这种函式包括太多的巢状函式呼叫，以致於效率非常低而且很慢。本章的後面将讨论一种我认为更好的方法。

另外，您将注意到，其中许多函式都需要对 OS/2 相容的 DIB 采取不同的处

理程序；这样，函式将频繁地检查 BITMAPINFO 结构的第一个栏位是否与 BITMAPCOREHEADER 结构的大小相同。

特别注意最後一个函式 PackedDibCreatePalette。这个函式用 DIB 中的颜色表来建立调色盘。如果 DIB 中没有颜色表（这意味著 DIB 的每图素有 16、24 或 32 位元），那么就不建立调色盘。我们有时会将从 DIB 颜色表建立的调色盘称为 DIB 自己的 调色盘。

PACKEDIB 档案都放在 SHOWDIB3，如程式 16-13 所示。

程式 16-13 SHOWDIB3

```
SHOWDIB3.C
/*------------------------------------------------------------------------
      SHOWDIB3.C --          Displays DIB with native palette
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------
*/

#include <windows.h>
#include "PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib3") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
      HWND        hwnd ;
      MSG         msg ;
      WNDCLASS    wndclass ;

      wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc              = WndProc ;
      wndclass.cbClsExtra               = 0 ;
      wndclass.cbWndExtra               = 0 ;
      wndclass.hInstance                = hInstance ;
      wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground      = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
      wndclass.lpszMenuName       = szAppName ;
      wndclass.lpszClassName      = szAppName ;
      if (!RegisterClass (&wndclass))
      {
                        MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                         szAppName, MB_ICONERROR) ;
                        return 0 ;
    }
```

```
     hwnd = CreateWindow (  szAppName, TEXT ("Show DIB #3: Native Palette"),
                        WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
                TranslateMessage (&msg) ;
                     DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (   HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
     static BITMAPINFO *    pPackedDib ;
     static HPALETTE                    hPalette ;
     static int                         cxClient, cyClient ;
     static OPENFILENAME    ofn ;
     static TCHAR           szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
     static TCHAR           szFilter[] =     TEXT       ("Bitmap       Files
(*.BMP)\0*.bmp\0")
      TEXT ("All Files (*.*)\0*.*\0\0") ;
     HDC                                        hdc ;
     PAINTSTRUCT                                ps ;

     switch (message)
     {
     case  WM_CREATE:
                        ofn.lStructSize             = sizeof (OPENFILENAME) ;
                        ofn.hwndOwner               = hwnd ;
                        ofn.hInstance               = NULL ;
                        ofn.lpstrFilter             = szFilter ;
                        ofn.lpstrCustomFilter       = NULL ;
                        ofn.nMaxCustFilter          = 0 ;
                        ofn.nFilterIndex            = 0 ;
                        ofn.lpstrFile               = szFileName ;
                        ofn.nMaxFile                = MAX_PATH ;
                        ofn.lpstrFileTitle          = szTitleName ;
                        ofn.nMaxFileTitle           = MAX_PATH ;
                        ofn.lpstrInitialDir         = NULL ;
                        ofn.lpstrTitle              = NULL ;
```

```
                    ofn.Flags                     = 0 ;
                    ofn.nFileOffset               = 0 ;
                    ofn.nFileExtension            = 0 ;
                    ofn.lpstrDefExt               = TEXT ("bmp") ;
                    ofn.lCustData                 = 0 ;
                    ofn.lpfnHook                  = NULL ;
                    ofn.lpTemplateName            = NULL ;

                    return 0 ;

case  WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

case  WM_COMMAND:
                    switch (LOWORD (wParam))
                    {
                    case IDM_FILE_OPEN:

                    // Show the File Open dialog box

                                        if (!GetOpenFileName (&ofn))
                                                  return 0 ;

                    // If there's an existing packed DIB, free the memory

                                        if (pPackedDib)
                                        {
                                              free (pPackedDib) ;
                                              pPackedDib = NULL ;
                                        }

                    // If there's an existing logical palette, delete it

                                        if (hPalette)
                                        {
                                                  DeleteObject (hPalette) ;
                                                  hPalette = NULL ;
                                        }
                                              // Load the packed DIB into memory

                                        SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                                        ShowCursor (TRUE) ;

                                        pPackedDib = PackedDibLoad (szFileName) ;

                                        ShowCursor (FALSE) ;
```

```
                                    SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                                                  if (pPackedDib)
                                                  {
                                    // Create the palette from the DIB color table

                                    hPalette = PackedDibCreatePalette (pPackedDib) ;
                                                  }
                                                  else
                                                  {
                         MessageBox (      hwnd, TEXT ("Cannot load DIB file"),
                                       szAppName, 0) ;
                                                  }
                                               InvalidateRect (hwnd, NULL, TRUE) ;
                                               return 0 ;
                              }
                              break ;

     case  WM_PAINT:
                              hdc = BeginPaint (hwnd, &ps) ;

                              if (hPalette)
                              {
                                               SelectPalette (hdc, hPalette, FALSE) ;
                                               RealizePalette (hdc) ;
                              }

                              if (pPackedDib)
                              SetDIBitsToDevice      (hdc,      0,0,PackedDibGetWidth
(pPackedDib),
                                                       PackedDibGetHeight
(pPackedDib),
                                               0,0,0,PackedDibGetHeight (pPackedDib),
                                                    PackedDibGetBitsPtr (pPackedDib),
                                                      pPackedDib,
                         DIB_RGB_COLORS) ;
                              EndPaint (hwnd, &ps) ;
                              return 0 ;

     case  WM_QUERYNEWPALETTE:
                              if (!hPalette)
                                               return FALSE ;

                              hdc = GetDC (hwnd) ;
                              SelectPalette (hdc, hPalette, FALSE) ;
                              RealizePalette (hdc) ;
                              InvalidateRect (hwnd, NULL, TRUE) ;
```

```
                    ReleaseDC (hwnd, hdc) ;
                    return TRUE ;

     case  WM_PALETTECHANGED:
                    if (!hPalette || (HWND) wParam == hwnd)
                                   break ;

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    UpdateColors (hdc) ;

                    ReleaseDC (hwnd, hdc) ;
                    break ;


     case  WM_DESTROY:
                    if (pPackedDib)
                                   free (pPackedDib) ;

                    if (hPalette)
                                   DeleteObject (hPalette) ;

                    PostQuitMessage (0) ;
                    return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB3.RC （摘录）

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
/
// Menu
SHOWDIB3 MENU DISCARDABLE
BEGIN
     POPUP "&File"
     BEGIN
      MENUITEM "&Open",                IDM_FILE_OPEN
     END
END
```

RESOURCE.H （摘录）

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib3.rc

#define IDM_FILE_OPEN       40001
```

SHOWDIB3 中的视窗讯息处理程式将 packed DIB 指标作为静态变数来维护，视窗讯息处理程式在「File Open」命令期间呼叫 PACKEDIB.C 中的 PackedDibLoad 函式时获得了此指标。在处理此命令的过程中，SHOWDIB3 也呼叫 PackedDibCreatePalette 来获得可能用於 DIB 的调色盘。注意，无论 SHOWDIB3 什么时候准备载入新的 DIB，都应先释放前一个 DIB 的记忆体，并删除前一个 DIB 的调色盘。在处理 WM_DESTROY 讯息的程序中，最後的 DIB 最後释放，最後的调色盘最後删除。

处理 WM_PAINT 讯息很简单：如果存在调色盘，则 SHOWDIB3 将它选进装置内容并显现它。然後它呼叫 SetDIBitsToDevice，并传递有关 DIB 的函式资讯（例如宽、高和指向 DIB 图素位元的指标 ），这些资讯从 PACKEDIB 中的函式获得。

另外，请记住 SHOWDIB3 依据 DIB 中的颜色表建立了调色盘。如果在 DIB 中没有颜色表——通常是 16 位元、24 位元和 32 位元 DIB 的情况——就不建立调色盘。在 8 位元显示模式下显示 DIB 时，它只能用标准保留的 20 种颜色显示。

对这个问题有两种解决方法：第一种是简单地使用「通用」调色盘，这种调色盘适用於许多图形。您也可以自己建立调色盘。第二种解决方法是分析 DIB 的图素位元，并决定要显示图像的最佳颜色。很明显，第二种方法将涉及更多的工作（对於程式写作者和处理器都是如此），但是我将在本章结束之前告诉您如何使用第二种方法。

「通用」调色盘

程式 16-14 所示的 SHOWDIB4 程式建立了一个通用的调色盘，它用於显示载入到程式中的所有 DIB。另外，SHOWDIB4 与 SHOWDIB3 非常相似。

程式 16-14　SHOWDIB4

```
SHOWDIB4.C
/*------------------------------------------------------------------------
-
      SHOWDIB4.C --    Displays DIB with "all-purpose" palette
                                              (c) Charles Petzold, 1998
------------------------------------------------------------------------
-*/

#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib4") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                            PSTR szCmdLine, int iCmdShow)
```

```
{
     HWND                  hwnd ;
     MSG                   msg ;
     WNDCLASS         wndclass ;

     wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground      = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName       = szAppName ;
     wndclass.lpszClassName      = szAppName ;
     if (!RegisterClass (&wndclass))
     {
                    MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                                              szAppName,
MB_ICONERROR) ;
                    return 0 ;
     }

     hwnd = CreateWindow (  szAppName,  TEXT  ("Show  DIB  #4:  All-Purpose
Palette"),
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

/*------------------------------------------------------------------------
---
     CreateAllPurposePalette: Creates a palette suitable for a wide variety
                of images; the palette has 247 entries, but 15 of them are
           duplicates or match the standard 20 colors.
------------------------------------------------------------------------
```

```
-*/

HPALETTE CreateAllPurposePalette (void)
{
      HPALETTE hPalette ;
      int                         i, incr, R, G, B ;
      LOGPALETTE *      plp ;

      plp = malloc (sizeof (LOGPALETTE) + 246 * sizeof (PALETTEENTRY)) ;
      plp->palVersion  = 0x0300 ;
      plp->palNumEntries     = 247 ;

                  // The following loop calculates 31 gray shades, but 3 of them
                  //    will match the standard 20 colors

      for (i = 0, G = 0, incr = 8 ; G <= 0xFF ; i++, G += incr)
      {
                        plp->palPalEntry[i].peRed    = (BYTE) G ;
                        plp->palPalEntry[i].peGreen  = (BYTE) G ;
                        plp->palPalEntry[i].peBlue   = (BYTE) G ;
                        plp->palPalEntry[i].peFlags  = 0 ;

                        incr = (incr == 9 ? 8 : 9) ;
      }

                  // The following loop is responsible for 216 entries, but 8 of
                  // them will match the standard 20 colors, and another
                  // 4 of them will match the gray shades above.

      for (R = 0 ; R <= 0xFF ; R += 0x33)
      for (G = 0 ; G <= 0xFF ; G += 0x33)
      for (B = 0 ; B <= 0xFF ; B += 0x33)
     {
                        plp->palPalEntry [i].peRed        = (BYTE) R ;
                        plp->palPalEntry [i].peGreen      = (BYTE) G ;
                        plp->palPalEntry [i].peBlue       = (BYTE) B ;
                        plp->palPalEntry [i].peFlags      = 0 ;

                        i++ ;
      }
      hPalette = CreatePalette (plp) ;
      free (plp) ;
      return hPalette ;
}

LRESULT CALLBACK WndProc (   HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
```

```
     static     BITMAPINFO *     pPackedDib ;
     static     HPALETTE                    hPalette ;
     static     int                         cxClient, cyClient ;
     static     OPENFILENAME     ofn ;
     static     TCHAR       szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
     static     TCHAR       szFilter[] =     TEXT       ("Bitmap       Files
(*.BMP)\0*.bmp\0")
      TEXT ("All Files (*.*)\0*.*\0\0") ;
     HDC                                                    hdc ;
     PAINTSTRUCT                                 ps ;
     switch (message)
    {
     case  WM_CREATE:
                    ofn.lStructSize             = sizeof (OPENFILENAME) ;
                    ofn.hwndOwner               = hwnd ;
                    ofn.hInstance               = NULL ;
                    ofn.lpstrFilter             = szFilter ;
                    ofn.lpstrCustomFilter       = NULL ;
                    ofn.nMaxCustFilter          = 0 ;
                    ofn.nFilterIndex            = 0 ;
                    ofn.lpstrFile               = szFileName ;
                    ofn.nMaxFile                = MAX_PATH ;
                    ofn.lpstrFileTitle          = szTitleName ;
                    ofn.nMaxFileTitle           = MAX_PATH ;
                    ofn.lpstrInitialDir         = NULL ;
                    ofn.lpstrTitle              = NULL ;
                    ofn.Flags                   = 0 ;
                    ofn.nFileOffset             = 0 ;
                    ofn.nFileExtension          = 0 ;
                    ofn.lpstrDefExt             = TEXT ("bmp") ;
                    ofn.lCustData               = 0 ;
                    ofn.lpfnHook                = NULL ;
                    ofn.lpTemplateName          = NULL ;

                                   // Create the All-Purpose Palette

                    hPalette = CreateAllPurposePalette () ;
                    return 0 ;

     case  WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
               return 0 ;

     case  WM_COMMAND:
                    switch (LOWORD (wParam))
                    {
                    case  IDM_FILE_OPEN:
```

```
                                        // Show the File Open dialog box

                                    if (!GetOpenFileName (&ofn))
                                                return 0 ;

                // If there's an existing packed DIB, free the memory
                                    if (pPackedDib)
                                        {
                                            free (pPackedDib) ;
                                            pPackedDib = NULL ;
                                        }

                                            // Load the packed DIB into memory

                                    SetCursor     (LoadCursor     (NULL,
IDC_WAIT)) ;
                                    ShowCursor (TRUE) ;

                                    pPackedDib      =      PackedDibLoad
(szFileName) ;

                                    ShowCursor (FALSE) ;
                                    SetCursor     (LoadCursor     (NULL,
IDC_ARROW)) ;

                                    if (!pPackedDib)
                                    {
                MessageBox (      hwnd, TEXT ("Cannot load DIB file"),
                                    szAppName, 0) ;
                                    }
                                    InvalidateRect (hwnd, NULL, TRUE) ;
                                    return 0 ;
                        }
                        break ;

      case  WM_PAINT:
                        hdc = BeginPaint (hwnd, &ps) ;

                        if (pPackedDib)
                        {
                                    SelectPalette (hdc, hPalette, FALSE) ;
                                    RealizePalette (hdc) ;

                             SetDIBitsToDevice     (hdc,0,0,PackedDibGetWidth
(pPackedDib),

PackedDibGetHeight (pPackedDib),
```

```
                                0,0,0,PackedDibGetHeight (pPackedDib),
                                      PackedDibGetBitsPtr (pPackedDib),
                                      pPackedDib,
                    DIB_RGB_COLORS) ;
                          }
                          EndPaint (hwnd, &ps) ;
                          return 0 ;

     case  WM_QUERYNEWPALETTE:
                          hdc = GetDC (hwnd) ;
                          SelectPalette (hdc, hPalette, FALSE) ;
                          RealizePalette (hdc) ;
                          InvalidateRect (hwnd, NULL, TRUE) ;

                          ReleaseDC (hwnd, hdc) ;
                          return TRUE ;

     case  WM_PALETTECHANGED:
                          if ((HWND) wParam != hwnd)

                          hdc = GetDC (hwnd) ;
                          SelectPalette (hdc, hPalette, FALSE) ;
                          RealizePalette (hdc) ;
                          UpdateColors (hdc) ;

                          ReleaseDC (hwnd, hdc) ;
                          break ;

     case  WM_DESTROY:
                          if (pPackedDib)
                                         free (pPackedDib) ;

                          DeleteObject (hPalette) ;

                          PostQuitMessage (0) ;
                          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB4.RC （摘录）

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
/
// Menu
SHOWDIB4 MENU DISCARDABLE
BEGIN
```

```
      POPUP "&Open"
      BEGIN
                        MENUITEM "&File",                   IDM_FILE_OPEN
      END
END
```

RESOURCE.H （摘录）

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib4.rc

#define IDM_FILE_OPEN           40001
```

在处理 WM_CREATE 讯息时，SHOWDIB4 将呼叫 CreateAllPurposePalette，并在程式中保留该调色盘，而在 WM_DESTROY 讯息处理期间删除它。因为程式知道调色盘一定存在，所以在处理 WM_PAINT、WM_QUERYNEWPALETTE 或 WM_PALETTECHANGED 讯息时，不必检查调色盘的存在。

CreateAllPurposePalette 函式似乎是用 247 个项目来建立逻辑调色盘，它超出了系统调色盘中允许程式正常存取的 236 个项目。的确如此，不过这样做很方便。这些项目中有 15 个被复制或者映射到 20 种标准的保留颜色中。

CreateAllPurposePalette 从建立 31 种灰阶开始，即 0x00、0x09、0x11、0x1A、0x22、0x2B、0x33、0x3C、0x44、0x4D、0x55、0x5E、0x66、0x6F、0x77、0x80、0x88、0x91、0x99、0xA2、0xAA、0xB3、0xBB、0xC4、0xCC、0xD5、0xDD、0xE6、0xEE、0xF9 和 0xFF 的红色、绿色和蓝色值。注意，第一个、最後一个和中间的项目都在标准的 20 种保留颜色中。下一个函式用红色、绿色和蓝色值的所有组合建立了颜色 0x00、0x33、0x66、0x99、0xCC 和 0xFF。这样就共有 216 种颜色，但是其中 8 种颜色复制了标准的 20 种保留颜色，而另外 4 个复制了前面计算的灰阶。如果将 PALETTEENTRY 结构的 peFlags 栏位设为 0，则 Windows 将不把复制的项目放进系统调色盘。

显然地，实际的程式不希望计算 16 位元、24 位元或者 32 位元 DIB 的最佳调色盘，程式将继续使用 DIB 颜色表来显示 8 位元 DIB。SHOWDIB4 不完成这项工作，它只对每件事都使用通用调色盘。因为 SHOWDIB4 是一个展示程式，而且您可以与 SHOWDIB3 显示的 8 位元 DIB 进行比较。如果看一些人像的彩色 DIB，那么您可能会得出这样的结论：SHOWDIB4 没有足够的颜色来精确地表示鲜艳的色调。

如果用 SHOWDIB4 中的 CreateAllPurposePalette 函式来试验（可能是通过将逻辑调色盘的大小减少到只有几个项目的方法），您将发现当调色盘选进装置内容时，Windows 将只使用调色盘中的颜色，而不使用标准的 20 种颜色调色盘的颜色。

中间色调色盘

Windows API 包括一个通用调色盘，程式可以通过呼叫 CreateHalftonePalette 来获得该调色盘。使用此调色盘的方法与使用从 SHOWDIB4 中的 CreateAllPurposePalette 获得调色盘的方法相同，或者您也可以与点阵图缩放模式中的 HALFTONE 设定——用 SetStretchBltMode 设定——一起使用。程式 16-15 所示的 SHOWDIB5 程式展示了使用中间色调色盘的方法。

程式 16-15 SHOWDIB5

```
SHOWDIB5.C
/*------------------------------------------------------------------------
      SHOWDIB5.C --     Displays DIB with halftone palette
                                             (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib5") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                    PSTR    szCmdLine,    int
iCmdShow)
{
      HWND                  hwnd ;
      MSG                   msg ;
      WNDCLASS          wndclass ;

      wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc              = WndProc ;
      wndclass.cbClsExtra               = 0 ;
      wndclass.cbWndExtra               = 0 ;
      wndclass.hInstance                = hInstance ;
      wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground      = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
      wndclass.lpszMenuName       = szAppName ;
      wndclass.lpszClassName      = szAppName ;

      if (!RegisterClass (&wndclass))
      {
                        MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                                              szAppName,
MB_ICONERROR) ;
```

```
                    return 0 ;
    }

     hwnd = CreateWindow (  szAppName, TEXT ("Show DIB #5: Halftone Palette"),
                            WS_OVERLAPPEDWINDOW,
                            CW_USEDEFAULT, CW_USEDEFAULT,
                            CW_USEDEFAULT, CW_USEDEFAULT,
                            NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
    {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
    }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (   HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
     static     BITMAPINFO *     pPackedDib ;
     static     HPALETTE         hPalette ;
     static     int         cxClient, cyClient ;
     static     OPENFILENAME     ofn ;
     static     TCHAR            szFileName     [MAX_PATH],     szTitleName
[MAX_PATH] ;
     static     TCHAR            szFilter[] =     TEXT    ("Bitmap    Files
(*.BMP)\0*.bmp\0")
      TEXT ("All Files (*.*)\0*.*\0\0") ;
     HDC                                    hdc ;
     PAINTSTRUCT                                ps ;

     switch (message)
    {
     case  WM_CREATE:
                    ofn.lStructSize             = sizeof (OPENFILENAME) ;
                    ofn.hwndOwner               = hwnd ;
                    ofn.hInstance               = NULL ;
                    ofn.lpstrFilter             = szFilter ;
                    ofn.lpstrCustomFilter       = NULL ;
                    ofn.nMaxCustFilter          = 0 ;
                    ofn.nFilterIndex            = 0 ;
                    ofn.lpstrFile               = szFileName ;
                    ofn.nMaxFile                = MAX_PATH ;
                    ofn.lpstrFileTitle          = szTitleName ;
```

```
                    ofn.nMaxFileTitle           = MAX_PATH ;
                    ofn.lpstrInitialDir         = NULL ;
                    ofn.lpstrTitle              = NULL ;
                    ofn.Flags                   = 0 ;
                    ofn.nFileOffset             = 0 ;
                    ofn.nFileExtension          = 0 ;
                    ofn.lpstrDefExt             = TEXT ("bmp") ;
                    ofn.lCustData               = 0 ;
                    ofn.lpfnHook                = NULL ;
                    ofn.lpTemplateName          = NULL ;

                              // Create the All-Purpose Palette

                    hdc = GetDC (hwnd) ;
                    hPalette = CreateHalftonePalette (hdc) ;
                    ReleaseDC (hwnd, hdc) ;
                    return 0 ;

case  WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

case  WM_COMMAND:
                    switch (LOWORD (wParam))
                    {
                    case  IDM_FILE_OPEN:

                                        // Show the File Open dialog box

                                   if (!GetOpenFileName (&ofn))
                                              return 0 ;

               // If there's an existing packed DIB, free the memory

                                   if (pPackedDib)
                                   {
                                         free (pPackedDib) ;
                                         pPackedDib = NULL ;
                                   }

                                   // Load the packed DIB into memory

                                   SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                                   ShowCursor (TRUE) ;

                                   pPackedDib = PackedDibLoad (szFileName) ;
```

```
                              ShowCursor (FALSE) ;
                              SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                                   if (!pPackedDib)
                                   {
          MessageBox ( hwnd, TEXT ("Cannot load DIB file"),
                     szAppName, 0) ;
                                   }
                                   InvalidateRect (hwnd, NULL, TRUE) ;
                                   return 0 ;
                    }
                    break ;

case  WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;

                    if (pPackedDib)
                    {
                                   // Set halftone stretch mode

                              SetStretchBltMode (hdc, HALFTONE) ;
                              SetBrushOrgEx (hdc, 0, 0, NULL) ;

                                   // Select and realize halftone palette

                              SelectPalette (hdc, hPalette, FALSE) ;
                              RealizePalette (hdc) ;

               // StretchDIBits rather than SetDIBitsToDevice
               StretchDIBits (  hdc,0,0,PackedDibGetWidth (pPackedDib),
                              PackedDibGetHeight (pPackedDib),
                              0,0,PackedDibGetWidth (pPackedDib),
                              PackedDibGetHeight (pPackedDib),
                              PackedDibGetBitsPtr (pPackedDib),
                              pPackedDib,
                              DIB_RGB_COLORS,
                              SRCCOPY) ;
                    }
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

case  WM_QUERYNEWPALETTE:
                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    InvalidateRect (hwnd, NULL, TRUE) ;

                    ReleaseDC (hwnd, hdc) ;
```

```
                    return TRUE ;

     case  WM_PALETTECHANGED:
                    if ((HWND) wParam != hwnd)

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    UpdateColors (hdc) ;

                    ReleaseDC (hwnd, hdc) ;
                    break ;

     case  WM_DESTROY:
                    if (pPackedDib)
                                   free (pPackedDib) ;

                    DeleteObject (hPalette) ;

                    PostQuitMessage (0) ;
                    return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB5.RC （摘录）

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
/
// Menu
SHOWDIB5 MENU DISCARDABLE
BEGIN
     POPUP "&Open"
     BEGIN
                    MENUITEM "&File",            IDM_FILE_OPEN
     END
END
```

RESOURCE.H （摘录）

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib5.rc

#define IDM_FILE_OPEN          40001
```

SHOWDIB5 程式类似於 SHOWDIB4，SHOWDIB4 中不使用 DIB 中的颜色表，而使用适用於图像范围更大的调色盘。为此，SHOWDIB5 使用了由 Windows 支援的逻辑调色盘，其代号可以从 CreateHalftonePalette 函式获得。

中间色调色盘并不比 SHOWDIB4 中的 CreateAllPurposePalette 函式所建立的调色盘更复杂。的确，如果只是拿来自用，结果是相似的。然而，如果您呼叫下面两个函式：

```
SetStretchBltMode (hdc, HALFTONE) ;
                              SetBrushOrgEx (hdc, x, y, NULL) ;
```

其中，x 和 y 是 DIB 左上角的装置座标，并且如果您用 StretchDIBits 而不是 SetDIBitsToDevice 来显示 DIB，那么结果会让您吃惊：颜色色调要比不设定点阵图缩放模式来使用 CreateAllPurposePalette 或者 CreateHalftonePalette 更精确。Windows 使用一种混色图案来处理中间色调色盘上的颜色，以使其更接近 8 位元显示卡上原始图像的颜色。与您所想像的一样，这样做的缺点是需要更多的处理时间。

索引调色盘颜色

现在开始处理 SetDIBitsToDevice、StretchDIBits、CreateDIBitmap、SetDIBits、GetDIBits 和 CreateDIBSection 的 fClrUse 参数。通常，您将这个参数设定为 DIB_RGB_COLORS(等於 0)。不过，您也能将它设定为 DIB_PAL_COLORS。在这种情况下，假定 BITMAPINFO 结构中的颜色表不包括 RGB 颜色值，而是包括逻辑调色盘中颜色项目的 16 位元索引。逻辑调色盘是作为第一个参数传递给函式的装置内容中目前选择的那个。实际上，在 CreateDIBSection 中，之所以需要指定一个非 NULL 的装置内容代号作为第一个参数，只是因为使用了 DIB_PAL_COLORS。

DIB_PAL_COLORS 能为您做些什么呢？它可能提高一些性能。考虑一下在 8 位元显示模式下呼叫 SetDIBitsToDevice 显示的 8 位元 DIB。Windows 首先必须在 DIB 颜色表的所有颜色中搜索与设备可用颜色最接近的颜色。然後设定一个小表，以便将 DIB 图素值映射到设备图素。也就是说，最多需要搜索 256 次最接近的颜色。但是如果 DIB 颜色表中含有从装置内容中选择颜色的逻辑调色盘项目索引，那么就可能跳过搜索。

除了使用调色盘索引以外，程式 16-16 所示的 SHOWDIB6 程式与 SHOWDIB3 相似。

程式 16-16　SHOWDIB6

```
SHOWDIB6.C
/*------------------------------------------------------------------------
-
      SHOWDIB6.C --     Display DIB with palette indices
                                             (c) Charles Petzold, 1998
------------------------------------------------------------------------
*/
```

```
#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib6") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                    PSTR    szCmdLine,    int
iCmdShow)
{
      HWND                  hwnd ;
      MSG                   msg ;
      WNDCLASS         wndclass ;

      wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc              = WndProc ;
      wndclass.cbClsExtra               = 0 ;
      wndclass.cbWndExtra               = 0 ;
      wndclass.hInstance                = hInstance ;
      wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground      = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
      wndclass.lpszMenuName       = szAppName ;
      wndclass.lpszClassName      = szAppName ;

      if (!RegisterClass (&wndclass))
      {
                        MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                                                szAppName,
MB_ICONERROR) ;
                        return 0 ;
    }

      hwnd = CreateWindow (  szAppName, TEXT ("Show DIB #6: Palette Indices"),
                        WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        NULL, NULL, hInstance, NULL) ;

      ShowWindow (hwnd, iCmdShow) ;
      UpdateWindow (hwnd) ;

      while (GetMessage (&msg, NULL, 0, 0))
      {
                  TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
```

```
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (  HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
     static     BITMAPINFO *     pPackedDib ;
     static     HPALETTE         hPalette ;
     static     int         cxClient, cyClient ;
     static     OPENFILENAME     ofn ;
     static     TCHAR            szFileName     [MAX_PATH],     szTitleName
[MAX_PATH] ;
     static     TCHAR            szFilter[] =     TEXT    ("Bitmap    Files
(*.BMP)\0*.bmp\0")
      TEXT ("All Files (*.*)\0*.*\0\0") ;
     HDC                   hdc ;
     int                   i, iNumColors ;
     PAINTSTRUCT                ps ;
     WORD                  *          pwIndex ;

     switch (message)
     {
     case  WM_CREATE:
                       ofn.lStructSize             = sizeof (OPENFILENAME) ;
                       ofn.hwndOwner               = hwnd ;
                       ofn.hInstance               = NULL ;
                       ofn.lpstrFilter             = szFilter ;
                       ofn.lpstrCustomFilter       = NULL ;
                       ofn.nMaxCustFilter          = 0 ;
                       ofn.nFilterIndex            = 0 ;
                       ofn.lpstrFile               = szFileName ;
                       ofn.nMaxFile                = MAX_PATH ;
                       ofn.lpstrFileTitle          = szTitleName ;
                       ofn.nMaxFileTitle           = MAX_PATH ;
                       ofn.lpstrInitialDir         = NULL ;
                       ofn.lpstrTitle              = NULL ;
                       ofn.Flags                   = 0 ;
                       ofn.nFileOffset             = 0 ;
                       ofn.nFileExtension          = 0 ;
                       ofn.lpstrDefExt             = TEXT ("bmp") ;
                       ofn.lCustData               = 0 ;
                       ofn.lpfnHook                = NULL ;
                       ofn.lpTemplateName          = NULL ;

                       return 0 ;

     case  WM_SIZE:
```

```
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

     case  WM_COMMAND:
                    switch (LOWORD (wParam))
                    {
                    case  IDM_FILE_OPEN:

                                        // Show the File Open dialog box
                                  if (!GetOpenFileName (&ofn))
                                              return 0 ;

                    // If there's an existing packed DIB, free the memory

                          if (pPackedDib)
                          {
                                        free (pPackedDib) ;
                                        pPackedDib = NULL ;
                    }

               // If there's an existing logical palette, delete it

                          if (hPalette)
                          {
                                        DeleteObject (hPalette) ;
                                        hPalette = NULL ;
                          }

                                  // Load the packed DIB into memory

                          SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                          ShowCursor (TRUE) ;

                          pPackedDib = PackedDibLoad (szFileName) ;

                          ShowCursor (FALSE) ;
                          SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                          if (pPackedDib)
                          {
                    // Create the palette from the DIB color table

               hPalette = PackedDibCreatePalette (pPackedDib) ;

                          // Replace DIB color table with indices

                                  if (hPalette)
```

```
                              {
                        iNumColors = PackedDibGetNumColors (pPackedDib) ;
                              pwIndex = (WORD *)
                              PackedDibGetColorTablePtr (pPackedDib) ;

                              for (i = 0 ; i < iNumColors ; i++)
                                          pwIndex[i] = (WORD) i ;
                                    }
                              }
                              else
                              {
    MessageBox (  hwnd, TEXT ("Cannot load DIB file"),
                              szAppName, 0) ;
                              }
                              InvalidateRect (hwnd, NULL, TRUE) ;
                              return 0 ;
                        }
                        break ;

case  WM_PAINT:
                        hdc = BeginPaint (hwnd, &ps) ;

                        if (hPalette)
                        {
                              SelectPalette (hdc, hPalette, FALSE) ;
                              RealizePalette (hdc) ;
                        }

                        if (pPackedDib)
                              SetDIBitsToDevice (hdc,0,0,
                               PackedDibGetWidth  (pPackedDib),
                              PackedDibGetHeight (pPackedDib),
                              0,0,0,PackedDibGetHeight (pPackedDib),
                              PackedDibGetBitsPtr (pPackedDib),
                              pPackedDib,
                              DIB_PAL_COLORS) ;
                        EndPaint (hwnd, &ps) ;
                        return 0 ;

case  WM_QUERYNEWPALETTE:
                        if (!hPalette)
                                          return FALSE ;

                        hdc = GetDC (hwnd) ;
                        SelectPalette (hdc, hPalette, FALSE) ;
                        RealizePalette (hdc) ;
                        InvalidateRect (hwnd, NULL, TRUE) ;
```

```
                    ReleaseDC (hwnd, hdc) ;
                    return TRUE ;

     case  WM_PALETTECHANGED:
                    if (!hPalette || (HWND) wParam == hwnd)
                                   break ;

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    UpdateColors (hdc) ;

                    ReleaseDC (hwnd, hdc) ;
                    break ;


     case  WM_DESTROY:
                    if (pPackedDib)
                                   free (pPackedDib) ;

                    if (hPalette)
                                   DeleteObject (hPalette) ;

                    PostQuitMessage (0) ;
                    return 0 ;
    }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB6.RC （摘录）

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
/
// Menu
SHOWDIB6 MENU DISCARDABLE
BEGIN
     POPUP "&File"
     BEGIN
                    MENUITEM "&Open",                      IDM_FILE_OPEN
     END
END
```

RESOURCE.H （摘录）

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib6.rc
//
#define IDM_FILE_OPEN         40001
```

SHOWDIB6 将 DIB 载入到记忆体并由此建立了调色盘以後，SHOWDIB6 简单地用以 0 开始的 WORD 索引替换了 DIB 颜色表中的颜色。PackedDibGetNumColors 函式将表示有多少种颜色，而 PackedDibGetColorTablePtr 函式传回指向 DIB 颜色表起始位置的指标。

注意，只有直接从 DIB 颜色表来建立调色盘时，此技术才可行。如果使用通用调色盘，则必须搜索最接近的颜色，以获得放入 DIB 的索引。

如果要使用调色盘索引，那么请在将 DIB 储存到磁片之前，确实替换掉 DIB 中的颜色表。另外，不要将包含调色盘索引的 DIB 放入剪贴簿。实际上，在显示之前，将调色盘索引放入 DIB，然後将 RGB 颜色值放回，会更安全一些。

调色盘和点阵图物件

程式 16-17 中的 SHOWDIB7 程式显示了如何使用与 DIB 相关联的调色盘，这些 DIB 是使用 CreateDIBitmap 函式转换成 GDI 点阵图物件的。

程式 16-17　SHOWDIB7

```
SHOWDIB7.C
/*-----------------------------------------------------------------------
      SHOWDIB7.C --          Shows DIB converted to DDB
                                                  (c) Charles Petzold, 1998
-----------------------------------------------------------------------
*/

#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib7") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                  PSTR    szCmdLine,    int
iCmdShow)
{
      HWND                  hwnd ;
      MSG                   msg ;
      WNDCLASS         wndclass ;

      wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc              = WndProc ;
      wndclass.cbClsExtra               = 0 ;
      wndclass.cbWndExtra               = 0 ;
      wndclass.hInstance                = hInstance ;
      wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
```

```
     wndclass.hbrBackground       = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName        = szAppName ;
     wndclass.lpszClassName       = szAppName ;

     if (!RegisterClass (&wndclass))
     {
                    MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                                                szAppName,
MB_ICONERROR) ;
                    return 0 ;
    }

     hwnd = CreateWindow (  szAppName, TEXT ("Show DIB #7: Converted to DDB"),
                  WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
    }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HBITMAP                    hBitmap ;
     static HPALETTE             hPalette ;
     static int                  cxClient, cyClient ;
     static OPENFILENAME       ofn ;
     static TCHAR                szFileName     [MAX_PATH],     szTitleName
[MAX_PATH] ;
     static TCHAR                szFilter[] =     TEXT    ("Bitmap    Files
(*.BMP)\0*.bmp\0")
      TEXT ("All Files (*.*)\0*.*\0\0") ;
     BITMAP                      bitmap ;
     BITMAPINFO               *  pPackedDib ;
     HDC                         hdc, hdcMem ;
     PAINTSTRUCT                 ps ;

     switch (message)
    {
```

```
     case  WM_CREATE:
                    ofn.lStructSize      = sizeof (OPENFILENAME) ;
                    ofn.hwndOwner        = hwnd ;
                    ofn.hInstance        = NULL ;
                    ofn.lpstrFilter      = szFilter ;
                    ofn.lpstrCustomFilter = NULL ;
                    ofn.nMaxCustFilter    = 0 ;
                    ofn.nFilterIndex     = 0 ;
                    ofn.lpstrFile        = szFileName ;
                    ofn.nMaxFile         = MAX_PATH ;
                    ofn.lpstrFileTitle    = szTitleName ;
                    ofn.nMaxFileTitle     = MAX_PATH ;
                    ofn.lpstrInitialDir   = NULL ;
                    ofn.lpstrTitle       = NULL ;
                    ofn.Flags           = 0 ;
                    ofn.nFileOffset      = 0 ;
                    ofn.nFileExtension    = 0 ;
                    ofn.lpstrDefExt      = TEXT ("bmp") ;
                    ofn.lCustData        = 0 ;
                    ofn.lpfnHook         = NULL ;
                    ofn.lpTemplateName    = NULL ;

                    return 0 ;

     case  WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

     case  WM_COMMAND:
                    switch (LOWORD (wParam))
                    {
                    case IDM_FILE_OPEN:

                                // Show the File Open dialog box

                                if (!GetOpenFileName (&ofn))
                          return 0 ;

               // If there's an existing packed DIB, free the memory

                if (hBitmap)
                                {
                                          DeleteObject (hBitmap) ;
                                          hBitmap = NULL ;
                          }

               // If there's an existing logical palette, delete it
```

```
          if (hPalette)
                    {
                              DeleteObject (hPalette) ;
                              hPalette = NULL ;
                    }

                         // Load the packed DIB into memory

          SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
          ShowCursor (TRUE) ;

                    pPackedDib = PackedDibLoad (szFileName) ;

                    ShowCursor (FALSE) ;
                    SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                    if (pPackedDib)
                    {
             // Create palette from the DIB and select it into DC

          hPalette = PackedDibCreatePalette (pPackedDib) ;

                         hdc = GetDC (hwnd) ;

                         if (hPalette)
                         {
               SelectPalette (hdc, hPalette, FALSE) ;
                                        RealizePalette (hdc) ;
                              }
               // Create the DDB from the DIB
          hBitmap = CreateDIBitmap(hdc,(PBITMAPINFOHEADER) pPackedDib,
                    CBM_INIT,PackedDibGetBitsPtr (pPackedDib),
                    pPackedDib, DIB_RGB_COLORS) ;
                              ReleaseDC (hwnd, hdc) ;

               // Free the packed-DIB memory

                              free (pPackedDib) ;
                         }
                         else
                         {
     MessageBox (  hwnd, TEXT ("Cannot load DIB file"),
                         szAppName, 0) ;
}
                              InvalidateRect (hwnd, NULL, TRUE) ;
                              return 0 ;
               }
```

```
                    break ;
     case  WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;

                    if (hPalette)
                    {
                                        SelectPalette (hdc, hPalette, FALSE) ;
                                        RealizePalette (hdc) ;
                    }
                    if (hBitmap)
                    {
                          GetObject (hBitmap, sizeof (BITMAP), &bitmap) ;
                          hdcMem = CreateCompatibleDC (hdc) ;
                          SelectObject (hdcMem, hBitmap) ;
                          BitBlt (hdc,0,0,bitmap.bmWidth, bitmap.bmHeight,
                                        hdcMem,     0,    0,    SRCCOPY) ;

                                DeleteDC (hdcMem) ;
                    }
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

     case  WM_QUERYNEWPALETTE:
                    if (!hPalette)
                                        return FALSE ;

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    InvalidateRect (hwnd, NULL, TRUE) ;

                    ReleaseDC (hwnd, hdc) ;
                    return TRUE ;

     case  WM_PALETTECHANGED:
                    if (!hPalette || (HWND) wParam == hwnd)
                                        break ;

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    UpdateColors (hdc) ;

                    ReleaseDC (hwnd, hdc) ;
                    break ;


     case  WM_DESTROY:
```

```
                        if (hBitmap)
                                        DeleteObject (hBitmap) ;

                        if (hPalette)
                                        DeleteObject (hPalette) ;

                                    PostQuitMessage (0) ;
                                    return 0 ;
      }
      return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB7.RC （摘录）

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
/
// Menu
SHOWDIB7 MENU DISCARDABLE
BEGIN
      POPUP "&File"
      BEGIN
                        MENUITEM "&Open",                   IDM_FILE_OPEN
      END
END
```

RESOURCE.H （摘录）

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib7.rc

#define IDM_FILE_OPEN     40001
```

与前面的程式一样，SHOWDIB7 获得了一个指向 packed DIB 的指标，该 DIB 回应功能表的「File」、「Open」命令。程式从 packed DIB 建立了调色盘，然後——还是在 WM_COMMAND 讯息的处理过程中——获得了用於视讯显示的装置内容，并选进调色盘，显现调色盘。然後 SHOWDIB7 呼叫 CreateDIBitmap 以便从 DIB 建立 DDB。如果调色盘没有选进装置内容并显现，那么 CreateDIBitmap 建立的 DDB 将不使用逻辑调色盘中的附加颜色。

呼叫 CreateDIBitmap 以後，该程式将释放 packed DIB 占用的记忆体空间。pPackedDib 变数不是静态变数。相反的，SHOWDIB7 按静态变数保留了点阵图代号（hBitmap）和逻辑调色盘代号（hPalette）。

在 WM_PAINT 讯息处理期间，调色盘再次选进装置内容并显现。GetObject 函式可获得点阵图的宽度和高度。然後，程式通过建立相容的记忆体装置内容在显示区域显示点阵图，选进点阵图，并执行 BitBlt。显示 DDB 时所用的调色

盘，必须与从 CreateDIBitmap 呼叫建立时所用的一样。

如果将点阵图复制到剪贴簿，则最好使用 packed DIB 格式。然後 Windows 可以将点阵图物件提供给希望使用这些点阵图的程式。然而，如果需要将点阵图物件复制到剪贴簿，则首先要获得视讯装置内容并显现调色盘。这允许 Windows 依据目前的系统调色盘将 DDB 转换为 DIB。

调色盘和 DIB 区块

最後,程式 16-18 所示的 SHOWDIB8 说明了如何使用带有 DIB 区块的调色盘。

程式 16-18　SHOWDIB8

```
SHOWDIB8.C
/*------------------------------------------------------------------------
      SHOWDIB8.C --          Shows DIB converted to DIB section
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*
/

#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib8") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                PSTR    szCmdLine,    int
iCmdShow)
{
      HWND                  hwnd ;
      MSG                   msg ;
      WNDCLASS         wndclass ;

      wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc              = WndProc ;
      wndclass.cbClsExtra               = 0 ;
      wndclass.cbWndExtra               = 0 ;
      wndclass.hInstance                = hInstance ;
      wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground       = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
      wndclass.lpszMenuName        = szAppName ;
      wndclass.lpszClassName       = szAppName ;

      if (!RegisterClass (&wndclass))
      {
            MessageBox (     NULL, TEXT ("This program requires Windows NT!"),
```

```
                                szAppName, MB_ICONERROR) ;
                    return 0 ;
     }

     hwnd = CreateWindow (  szAppName, TEXT ("Show DIB #8: DIB Section"),
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HBITMAP         hBitmap ;
     static HPALETTE        hPalette ;
     static int             cxClient, cyClient ;
     static OPENFILENAME    ofn ;
     static PBYTE           pBits ;
     static TCHAR           szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
     static TCHAR       szFilter[] =    TEXT ("Bitmap Files (*.BMP)\0*.bmp\0")
      TEXT ("All Files (*.*)\0*.*\0\0") ;
     BITMAP                     bitmap ;
     BITMAPINFO                 *     pPackedDib ;
        HDC                     hdc, hdcMem ;
     PAINTSTRUCT                ps ;

     switch (message)
     {
     case  WM_CREATE:
                    ofn.lStructSize        = sizeof (OPENFILENAME) ;
                    ofn.hwndOwner          = hwnd ;
                    ofn.hInstance          = NULL ;
                    ofn.lpstrFilter        = szFilter ;
                    ofn.lpstrCustomFilter  = NULL ;
                    ofn.nMaxCustFilter     = 0 ;
                    ofn.nFilterIndex       = 0 ;
                    ofn.lpstrFile          = szFileName ;
                    ofn.nMaxFile           = MAX_PATH ;
```

```
                    ofn.lpstrFileTitle     = szTitleName ;
                    ofn.nMaxFileTitle      = MAX_PATH ;
                    ofn.lpstrInitialDir    = NULL ;
                    ofn.lpstrTitle         = NULL ;
                    ofn.Flags              = 0 ;
                    ofn.nFileOffset        = 0 ;
                    ofn.nFileExtension     = 0 ;
                    ofn.lpstrDefExt        = TEXT ("bmp") ;
                    ofn.lCustData          = 0 ;
                    ofn.lpfnHook           = NULL ;
                    ofn.lpTemplateName     = NULL ;

                    return 0 ;

     case  WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

     case  WM_COMMAND:
                    switch (LOWORD (wParam))
                    {
                    case  IDM_FILE_OPEN:

               // Show the File Open dialog box

                         if (!GetOpenFileName (&ofn))
                                   return 0 ;

               // If there's an existing packed DIB, free the memory

                         if (hBitmap)
                         {
                                   DeleteObject (hBitmap) ;
                                   hBitmap = NULL ;
                         }

               // If there's an existing logical palette, delete it

                         if (hPalette)
                         {
                                   DeleteObject (hPalette) ;
                                   hPalette = NULL ;
                         }
               // Load the packed DIB into memory

                         SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                         ShowCursor (TRUE) ;
```

```
                                    pPackedDib = PackedDibLoad (szFileName) ;
                                    ShowCursor (FALSE) ;
                                    SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                                          if (pPackedDib)
                                          {
                          // Create the DIB section from the DIB

hBitmap = CreateDIBSection (NULL,pPackedDib,DIB_RGB_COLORS,&pBits,NULL, 0) ;

                    // Copy the bits

                    CopyMemory (pBits, PackedDibGetBitsPtr  (pPackedDib),
                                       PackedDibGetBitsSize (pPackedDib)) ;

                      // Create palette from the DIB

                     hPalette = PackedDibCreatePalette (pPackedDib) ;

                        // Free the packed-DIB memory

                                                free (pPackedDib) ;
                                    }
                                    else
                                    {
                  MessageBox (   hwnd, TEXT ("Cannot load DIB file"),
                                 szAppName, 0) ;
                                                }
                                    InvalidateRect (hwnd, NULL, TRUE) ;
                                    return 0 ;
                          }
                          break ;
          case  WM_PAINT:
                          hdc = BeginPaint (hwnd, &ps) ;

                          if (hPalette)
                          {
                                    SelectPalette (hdc, hPalette, FALSE) ;
                                    RealizePalette (hdc) ;
                          }
                          if (hBitmap)
                          {
                                    GetObject   (hBitmap,   sizeof   (BITMAP),
&bitmap) ;

                                    hdcMem = CreateCompatibleDC (hdc) ;
                                    SelectObject (hdcMem, hBitmap) ;
```

```
                BitBlt (    hdc,    0, 0, bitmap.bmWidth, bitmap.bmHeight,
                            hdcMem, 0, 0, SRCCOPY) ;

                                DeleteDC (hdcMem) ;
                        }
                        EndPaint (hwnd, &ps) ;
                        return 0 ;

      case  WM_QUERYNEWPALETTE:
                        if (!hPalette)
                                        return FALSE ;

                        hdc = GetDC             (hwnd) ;
                        SelectPalette           (hdc, hPalette, FALSE) ;
                        RealizePalette          (hdc) ;
                        InvalidateRect          (hwnd, NULL, TRUE) ;

                        ReleaseDC (hwnd, hdc) ;
                        return TRUE ;

      case  WM_PALETTECHANGED:
                        if (!hPalette || (HWND) wParam == hwnd)
                                        break ;

                        hdc = GetDC (hwnd) ;
                        SelectPalette (hdc, hPalette, FALSE) ;
                        RealizePalette (hdc) ;
                        UpdateColors (hdc) ;

                        ReleaseDC (hwnd, hdc) ;
                        break ;


      case  WM_DESTROY:
                        if (hBitmap)
                                        DeleteObject (hBitmap) ;

                        if (hPalette)
                                        DeleteObject (hPalette) ;

                        PostQuitMessage (0) ;
                        return 0 ;
    }
      return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB8.RC （摘录）

```
//Microsoft Developer Studio generated resource script.
```

```
#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
/
// Menu
SHOWDIB8 MENU DISCARDABLE
BEGIN
      POPUP "&File"
      BEGIN
                        MENUITEM "&Open",                   IDM_FILE_OPEN
      END
END
 RESOURCE.H （摘录）
// Microsoft Developer Studio generated include file.
// Used by ShowDib8.rc

#define IDM_FILE_OPEN           40001
```

在 SHOWDIB7 和 SHOWDIB8 中的 WM_PAINT 处理是一样的：两个程式都将点阵图代号（hBitmap）和逻辑调色盘代号（hPalette）作为静态变数。调色盘被选进装置内容并显现，点阵图的宽度和高度从 GetObject 函式获得，程式建立记忆体装置内容并选进点阵图，然後通过呼叫 BitBlt 将点阵图显示到显示区域。

两个程式之间最大的差别在於处理「File」、「Open」功能表命令的程序。在获得指向 packed DIB 的指标并建立了调色盘以後，SHOWDIB7 必须将调色盘选进视讯装置内容，并在呼叫 CreateDIBitmap 之前显现。SHOWDIB8 在获得 packed DIB 指标以後呼叫 CreateDIBSection。不必将调色盘选进装置内容，这是因为 CreateDIBSection 不将 DIB 转换成设备相关的格式。的确，CreateDIBSection 的第一个参数（即装置内容代号）的唯一用途在於您是否使用 DIB_PAL_COLORS 旗标。

呼叫 CreateDIBSection 以後，SHOWDIB8 将图素位元从 packed DIB 复制到从 CreateDIBSection 函式传回的记忆体位置，然後呼叫 PackedDibCreatePalette。尽管此函式便於程式使用，但是 SHOWDIB8 将依据从 GetDIBColorTable 函式传回的资讯建立调色盘。

DIB 处理程式库

就是现在——经过我们长时间地学习 GDI 点阵图物件、装置无关点阵图、DIB 区块和 Windows 调色盘管理器之後——我们才做好了开发一套有助於处理点阵图的函式的准备。

前面的 PACKEDIB 档案展示了一种可能的方法：记忆体中的 packed DIB 只

用指向它的指标表示。程式所的有关 DIB 的全部资讯都可以从存取表头资讯结构的函式获得。然而，实际上到「get pixel」和「set pixel」常式时，这种方法就会产生严重的执行问题。图像处理任务当然需要存取点阵图位元，并且这些函式也应该尽可能地快。

可能的 C++的解决方式中包括建立 DIB 类别，这时指向 packed DIB 的指标正好是一个成员变数。其他成员变数和成员函式有助於更快地执行获得和设定 DIB 中的图素的常式。不过，因为我在第一章已经指出，对於本书您只需要了解 C，使用 C++将是其他书的范围。

当然，用 C++能做的事情用 C 也能做。一个好的例子就是许多 Windows 函式都使用代号。除了将代号当作数值以外，应用程式对它还了解什么呢？程式知道代号引用特殊的函式物件，还知道函式用於处理现存的物件。显然，作业系统按某种方式用代号来引用物件的内部资讯。代号可以与结构指标一样简单。

例如，假设有一个函式集，这些函式都使用一个称为 HDIB 的代号。HDIB 是什么呢？它可能在某个表头档案中定义如下：

```
typedef void * HDIB ;
```

此定义用「不关您的事」回答了「HDIB 是什么」这个问题。

然而，实际上 HDIB 可能是结构指标，该结构不仅包括指向 packed DIB 的指标，还包括其他资讯：

```
typedef struct
{
      BITMAPINFO  *     pPackedDib ;
      int                           cx, cy, cBitsPerPixel, cBytesPerRow ;
      BYTE                    * pBits ;
{
DIBSTRUCTURE, * PDIBSTRUCTURE ;
```

此结构的其他五个栏位包括从 packed DIB 中引出的资讯。当然，结构中这些值允许更快速地存取它们。不同的 DIB 程式库函式都可以处理这个结构，而不是 pPackedDib 指标。可以按下面的方法来执行 DibGetPixelPointer 函式：

```
BYTE * DibGetPixelPointer (HDIB hdib, int x, int y)
{
      PDIBSTRUCTURE pdib = hdib ;
      return pdib->pBits +   y * pdib->cBytesPerRow +
                                  x * pdib->cBitsPerPixel / 8 ;
}
```

当然，这种方法可能要比 PACKEDIB.C 中执行「get pixel」常式快。

由於这种方法非常合理，所以我决定放弃 packed DIB，并改用处理 DIB 区块的 DIB 程式库。这实际上使我们对 packed DIB 的处理有更大的弹性（也就是说，能够在装置无关的方式下操纵 DIB 图素位元），而且在 Windows NT 下执行

时将更有效。

DIBSTRUCT 结构

DIBHELP.C 档案——如此命名是因为对处理 DIB 提供帮助——有上千行，并在几个小部分中显示。但是首先让我们看一下 DIBHELP 函式所处理的结构，该结构在 DIBHELP.C 中定义如下：

```
typedef struct
{
      PBYTE       * ppRow ;      // array of row pointers
      int         iSignature ;     // = "Dib "
      HBITMAP     hBitmap ;        // handle returned from CreateDIBSection
      BYTE        *     pBits ;    // pointer to bitmap bits
      DIBSECTION  ds ;             // DIBSECTION structure
      int         iRShift[3] ;     // right-shift values for color masks
      int         iLShift[3] ;     // left-shift values for color masks
}
DIBSTRUCT, * PDIBSTRUCT ;
```

现在跳过第一个栏位。它之所以为第一个栏位是因为它使某些巨集更易於使用——在讨论完其他栏位以後再来理解第一个栏位就更容易了。

在 DIBHELP.C 中，当 DIB 建立的函式首先设定了此结构时，第二个栏位就设定为文字字串「Dib」的二进位值。通过一些 DIBHELP 函式，第二个栏位将用於结构有效指标的一个标记。

第三个栏位，即 hBitmap，是从 CreateDIBSection 函式传回的点阵图代号。您将想起该代号可有多种使用方式，它与我们在第十四章遇到的 GDI 点阵图物件的代号用法一样。不过，从 CreateDIBSection 传回的代号将涉及按装置无关格式储存的点阵图，该点阵图格式一直储存到通过呼叫 BitBlt 和 StretchBlt 来将位元图画到输出设备。

DIBSTRUCT 的第四个栏位是指向点阵图位元的指标。此值也可由 CreateDIBSection 函式设定。您将想起，作业系统将控制这个记忆体块，但应用程式有存取它的许可权。在删除点阵图代号时，记忆体块将自动释放。

DIBSTRUCT 的第五个栏位是 DIBSECTION 结构。如果您有从 CreateDIBSection 传回的点阵图代号，那么您可以将代号传递给 GetObject 函式以获得有关 DIBSECTION 结构中的点阵图资讯：

```
GetObject (hBitmap, sizeof (DIBSECTION), &ds) ;
```

作为提示，DIBSECTION 结构在 WINGDI.H 中定义如下：

```
typedef struct tagDIBSECTION {
      BITMAP                       dsBm ;
      BITMAPINFOHEADER dsBmih ;
      DWORD            dsBitfields[3] ;    // Color masks
```

```
      HANDLE         dshSection ;
      DWORD          dsOffset ;
}
DIBSECTION, * PDIBSECTION ;
```

第一个栏位是 BITMAP 结构，它与 CreateBitmapIndirect 一起建立点阵图物件，与 GetObject 一起传回关於 DDB 的资讯。第二个栏位是 BITMAPINFOHEADER 结构。不管点阵图资讯结构是否传递给 CreateDIBSection 函式，DIBSECTION 结构总有 BITMAPINFOHEADER 结构而不是其他结构，例如 BITMAPCOREHEADER 结构。这意味著在存取此结构时，DIBHELP.C 中的许多函式都不必检查与 OS/2 相容的 DIB。

对於 16 位元和 32 位元的 DIB，如果 BITMAPINFOHEADER 结构的 biCompression 栏位是 BI_BITFIELDS，那么在资讯表头结构後面通常有三个遮罩值。这些遮罩值决定如何将 16 位元和 32 位图素值转换成 RGB 颜色。遮罩储存在 DIBSECTION 结构的第三个栏位中。

DIBSECTION 结构的最後两个栏位指的是 DIB 区块，此区块由档案映射建立。DIBHELP 不使用 CreateDIBSection 的这个特性，因此可以忽略这些栏位。

DIBSTRUCT 的最後两个栏位储存左右移位值，这些值用於处理 16 位元和 32 位元 DIB 的颜色遮罩。我们将在第十五章讨论这些移位值。

让我们再回来看一下 DIBSTRUCT 的第一个栏位。正如我们所看到的一样，在开始建立 DIB 时，此栏位设定为指向一个指标阵列的指标，该阵列中的每个指标都指向 DIB 中的一行图素。这些指标允许以更快的方式来获得 DIB 图素位元，同时也被定义，以便顶行可以首先引用 DIB 图素位元。此阵列的最後一个元素——引用 DIB 图像的最底行——通常等於 DIBSTRUCT 的 pBits 栏位。

资讯函式

DIBHELP.C 以定义 DIBSTRUCT 结构开始，然後提供一个函式集，此函式集允许应用程式获得有关 DIB 区块的资讯。程式 16-19 显示了 DIBHELP.C 的第一部分。

程式 16-19　DIBHELP.C 档案的第一部分

```
DIBHELP.C （第一部分）
/*-----------------------------------------------------------------------
      DIBHELP.C --      DIB Section Helper Routines
                                              (c) Charles Petzold, 1998
-----------------------------------------------------------------------
*/

#include <windows.h>
#include "dibhelp.h"
```

```
#define HDIB_SIGNATURE (* (int *) "Dib ")
typedef struct
{
      PBYTE                  *     ppRow ;    // must be first field for macros!
      int                    iSignature ;
      HBITMAP                hBitmap ;
      BYTE                   *     pBits ;
      DIBSECTION       ds ;
      int                          iRShift[3] ;
      int                          iLShift[3] ;
}
DIBSTRUCT, * PDIBSTRUCT ;

/*-------------------------------------------------------------------------
--
      DibIsValid:  Returns TRUE if hdib points to a valid DIBSTRUCT
-------------------------------------------------------------------------
-*/

BOOL DibIsValid (HDIB hdib)
{
      PDIBSTRUCT pdib = hdib ;
      if (pdib == NULL)
                        return FALSE ;
      if (IsBadReadPtr (pdib, sizeof (DIBSTRUCT)))
                        return FALSE ;
      if (pdib->iSignature != HDIB_SIGNATURE)
                        return FALSE ;
      return TRUE ;
}

/*-------------------------------------------------------------------------
--
      DibBitmapHandle: Returns the handle to the DIB section bitmap object
-------------------------------------------------------------------------
-*/

HBITMAP DibBitmapHandle (HDIB hdib)
{
      if (!DibIsValid (hdib))
                        return NULL ;
      return ((PDIBSTRUCT) hdib)->hBitmap ;
}

/*-------------------------------------------------------------------------
-
      DibWidth:  Returns the bitmap pixel width
```

```
-------------------------------------------------------------------------
-*/

int DibWidth (HDIB hdib)
{
      if (!DibIsValid (hdib))
                        return 0 ;
      return ((PDIBSTRUCT) hdib)->ds.dsBm.bmWidth ;
}

/*-------------------------------------------------------------------------
-
      DibHeight:  Returns the bitmap pixel height
-------------------------------------------------------------------------
*/

int DibHeight (HDIB hdib)
{
      if (!DibIsValid (hdib))
                        return 0 ;
      return ((PDIBSTRUCT) hdib)->ds.dsBm.bmHeight ;
}

/*-------------------------------------------------------------------------
-
   DibBitCount:   Returns the number of bits per pixel
-------------------------------------------------------------------------
*/

int DibBitCount (HDIB hdib)
{
      if (!DibIsValid (hdib))
                        return 0 ;

      return ((PDIBSTRUCT) hdib)->ds.dsBm.bmBitsPixel ;
}

/*-------------------------------------------------------------------------
--
      DibRowLength:    Returns the number of bytes per row of pixels
-------------------------------------------------------------------------
-*/

int DibRowLength (HDIB hdib)
{
      if (!DibIsValid (hdib))
                        return 0 ;
      return 4 * ((DibWidth (hdib) * DibBitCount (hdib) + 31) / 32) ;
```

```
}

/*-------------------------------------------------------------------------
-
   DibNumColors:        Returns the number of colors in the color table
-------------------------------------------------------------------------
*/

int DibNumColors (HDIB hdib)
{
      PDIBSTRUCT pdib = hdib ;

      if (!DibIsValid (hdib))
                        return 0 ;

      if (pdib->ds.dsBmih.biClrUsed != 0)
     {
                        return pdib->ds.dsBmih.biClrUsed ;
     }
      else if (DibBitCount (hdib) <= 8)
     {
                  return 1 << DibBitCount (hdib) ;
     }
      return 0 ;
}
/*-------------------------------------------------------------------------
-
      DibMask:  Returns one of the color masks
-------------------------------------------------------------------------*
/

DWORD       DibMask (HDIB hdib, int i)
{
      PDIBSTRUCT pdib = hdib ;

      if (!DibIsValid (hdib) || i < 0 || i > 2)
                        return 0 ;

      return pdib->ds.dsBitfields[i] ;
}

/*-------------------------------------------------------------------------
--
      DibRShift:  Returns one of the right-shift values
-------------------------------------------------------------------------
-*/

int   DibRShift (HDIB hdib, int i)
```

```
{
      PDIBSTRUCT pdib = hdib ;

      if (!DibIsValid (hdib) || i < 0 || i > 2)
                        return 0 ;

      return pdib->iRShift[i] ;
}

/*-------------------------------------------------------------------------
--
      DibLShift:  Returns one of the left-shift values
-------------------------------------------------------------------------
*/

int DibLShift (HDIB hdib, int i)
{
      PDIBSTRUCT pdib = hdib ;

      if (!DibIsValid (hdib) || i < 0 || i > 2)
                        return 0 ;

      return pdib->iLShift[i] ;
}

/*-------------------------------------------------------------------------
-
      DibCompression:  Returns the value of the biCompression field
-------------------------------------------------------------------------
*/
int DibCompression (HDIB hdib)
{
      if (!DibIsValid (hdib))
                        return 0 ;

      return ((PDIBSTRUCT) hdib)->ds.dsBmih.biCompression ;
}

/*-------------------------------------------------------------------------
-
      DibIsAddressable:  Returns TRUE if the DIB is not compressed
-------------------------------------------------------------------------
*/

BOOL  DibIsAddressable (HDIB hdib)
{
      int iCompression ;
```

```
     if (!DibIsValid (hdib))
                    return FALSE ;

     iCompression = DibCompression (hdib) ;

     if (  iCompression == BI_RGB || iCompression == BI_BITFIELDS)
                    return TRUE ;

     return FALSE ;
}

/*-------------------------------------------------------------------------
-
     These functions return the sizes of various components of the DIB section
     AS THEY WOULD APPEAR in a packed DIB. These functions aid in converting
     the DIB section to a packed DIB and in saving DIB files.
-------------------------------------------------------------------------
-*/

DWORD DibInfoHeaderSize (HDIB hdib)
{
     if (!DibIsValid (hdib))
                    return 0 ;
     return ((PDIBSTRUCT) hdib)->ds.dsBmih.biSize ;
}

DWORD      DibMaskSize (HDIB hdib)
{
     PDIBSTRUCT pdib = hdib ;
     if (!DibIsValid (hdib))
                    return 0 ;

     if (pdib->ds.dsBmih.biCompression == BI_BITFIELDS)
                    return 3 * sizeof (DWORD) ;

     return 0 ;
}

DWORD DibColorSize (HDIB hdib)
{
     return DibNumColors (hdib) * sizeof (RGBQUAD) ;
}

DWORD DibInfoSize (HDIB hdib)
{
     return DibInfoHeaderSize(hdib) + DibMaskSize(hdib) + DibColorSize(hdib) ;
}
```

```
DWORD DibBitsSize (HDIB hdib)
{
      PDIBSTRUCT pdib = hdib ;

      if (!DibIsValid (hdib))
                        return 0 ;

      if (pdib->ds.dsBmih.biSizeImage != 0)
     {
                        return pdib->ds.dsBmih.biSizeImage ;
     }
      return DibHeight (hdib) * DibRowLength (hdib) ;
}

DWORD DibTotalSize (HDIB hdib)
{
      return DibInfoSize (hdib) + DibBitsSize (hdib) ;
}

/*-------------------------------------------------------------------------
-
      These functions return pointers to the various components of the DIB
      section.
-------------------------------------------------------------------------
-*/
BITMAPINFOHEADER * DibInfoHeaderPtr (HDIB hdib)
{
      if (!DibIsValid (hdib))
                        return NULL ;
      return & (((PDIBSTRUCT) hdib)->ds.dsBmih) ;
}

DWORD * DibMaskPtr (HDIB hdib)
{
      PDIBSTRUCT pdib = hdib ;
      if (!DibIsValid (hdib))
                        return 0 ;
      return pdib->ds.dsBitfields ;
}

void * DibBitsPtr (HDIB hdib)
{
      if (!DibIsValid (hdib))
                        return NULL ;
      return ((PDIBSTRUCT) hdib)->pBits ;
}

/*-------------------------------------------------------------------------
```

```
-
      DibSetColor:  Obtains entry from the DIB color table
------------------------------------------------------------------------------
-*/

BOOL DibGetColor (HDIB hdib, int index, RGBQUAD * prgb)
{
      PDIBSTRUCT pdib = hdib ;
      HDC                          hdcMem ;
      int                          iReturn ;

      if (!DibIsValid (hdib))
                        return 0 ;
      hdcMem = CreateCompatibleDC (NULL) ;
      SelectObject (hdcMem, pdib->hBitmap) ;
      iReturn = GetDIBColorTable (hdcMem, index, 1, prgb) ;
      DeleteDC (hdcMem) ;
      return iReturn ? TRUE : FALSE ;
}

/*------------------------------------------------------------------------------
--
      DibGetColor:  Sets an entry in the DIB color table
------------------------------------------------------------------------------
*/

BOOL DibSetColor (HDIB hdib, int index, RGBQUAD * prgb)
{
      PDIBSTRUCT  pdib = hdib ;
      HDC                     hdcMem ;
      int                     iReturn ;

      if (!DibIsValid (hdib))
                        return 0 ;
      hdcMem = CreateCompatibleDC (NULL) ;
      SelectObject (hdcMem, pdib->hBitmap) ;
      iReturn = SetDIBColorTable (hdcMem, index, 1, prgb) ;
      DeleteDC (hdcMem) ;

      return iReturn ? TRUE : FALSE ;
}
```

DIBHELP.C 中的大部分函式是不用解释的。DibIsValid 函式能有助於保护整个系统。在试图引用 DIBSTRUCT 中的资讯之前，其他函式都呼叫 DibIsValid。所有这些函式都有（而且通常是只有）HDIB 型态的第一个参数，（ 我们将立即看到）该参数在 DIBHELP.H 中定义为空指标。这些函式可以将此参数储存到 PDIBSTRUCT，然後再存取结构中的栏位。

注意传回 BOOL 值的 DibIsAddressable 函式。DibIsNotCompressed 函式也可以呼叫此函式。传回值表示独立的 DIB 图素能否定址。

以 DibInfoHeaderSize 开始的函式集将取得 DIB 区块中不同元件出现在 packed DIB 中的大小。与我们所看到的一样，这些函式有助於将 DIB 区块转换成 packed DIB，并储存 DIB 档案。这些函式的後面是获得指向不同 DIB 元件的指标的函式集。

尽管 DIBHELP.C 包括名称为 DibInfoHeaderPtr 的函式，而且该函式将获得指向 BITMAPINFOHEADER 结构的指标，但还是没有函式可以获得 BITMAPINFO 结构指标——即接在 DIB 颜色表後面的资讯结构。这是因为在处理 DIB 区块时，应用程式并不直接存取这种型态的结构。BITMAPINFOHEADER 结构和颜色遮罩都在 DIBSECTION 结构中有效，而且从 CreateDIBSection 函式传回指向图素位元的指标，这时通过呼叫 GetDIBColorTable 和 SetDIBColorTable，就只能间接存取 DIB 颜色表。这些功能都封装到 DIBHELP 的 DibGetColor 和 DibSetColor 函式里头了。

在 DIBHELP.C 的後面，档案 DibCopyToInfo 配置一个指向 BITMAPINFO 结构的指标，并填充资讯，但是那与获得指向记忆体中现存结构的指标不完全相同。

读、写图素

应用程式维护 packed DIB 或 DIB 区块的一个引人注目的优点是能够直接操作 DIB 图素位元。程式 16-20 所示的 DIBHELP.C 第二部分列出了提供此功能的函式。

程式 16-20　DIBHELP.C 档案的第二部分

```
DIBHELP.C （第二部分）
/*-------------------------------------------------------------------------
--
      DibPixelPtr:     Returns a pointer to the pixel at position (x, y)
-------------------------------------------------------------------------
-*/

BYTE * DibPixelPtr (HDIB hdib, int x, int y)
{
      if (!DibIsAddressable (hdib))
                        return NULL ;
      if (x < 0 || x >= DibWidth (hdib) || y < 0 || y >= DibHeight (hdib))
                        return NULL ;
      return (((PDIBSTRUCT) hdib)->ppRow)[y] + (x * DibBitCount (hdib) >> 3) ;
}

/*-------------------------------------------------------------------------
```

```
-
      DibGetPixel:  Obtains a pixel value at (x, y)
-------------------------------------------------------------------------
-*/

DWORD DibGetPixel (HDIB hdib, int x, int y)
{
      PBYTE pPixel ;
      if (!(pPixel = DibPixelPtr (hdib, x, y)))
                  return 0 ;
      switch (DibBitCount (hdib))
      {
      case        1:  return 0x01 & (* pPixel >> (7 - (x & 7))) ;
      case        4:  return 0x0F & (* pPixel >> (x & 1 ? 0 : 4)) ;
      case        8:  return * pPixel ;
      case        16: return * (WORD *) pPixel ;
      case        24: return 0x00FFFFFF & * (DWORD *) pPixel ;
      case        32: return * (DWORD *) pPixel ;
      }
      return 0 ;
}

/*-------------------------------------------------------------------------
      DibSetPixel:  Sets a pixel value at (x, y)
-------------------------------------------------------------------------*
/

BOOL DibSetPixel (HDIB hdib, int x, int y, DWORD dwPixel)
{
      PBYTE pPixel ;
      if (!(pPixel = DibPixelPtr (hdib, x, y)))
                        return FALSE ;
      switch (DibBitCount (hdib))
     {
      case  1:    *     pPixel &= ~(1    << (7 - (x & 7))) ;
                  *     pPixel |= dwPixel       << (7 - (x & 7)) ;
                                    break ;
      case  4:  * pPixel &= 0x0F   << (x & 1 ? 4 : 0) ;
                  *     pPixel |= dwPixel       << (x & 1 ? 0 : 4) ;
                                    break ;
      case 8:     *     pPixel = (BYTE) dwPixel ;
                                    break ;

      case 16:    *     (WORD *) pPixel = (WORD) dwPixel ;
                                    break ;

      case 24:    *     (RGBTRIPLE *) pPixel = * (RGBTRIPLE *) &dwPixel ;
                                    break ;
```

```
      case 32:    *     (DWORD *) pPixel = dwPixel ;
                                        break ;
      default:
                        return FALSE ;
      }
      return TRUE ;
}

/*-------------------------------------------------------------------------
-
      DibGetPixelColor:  Obtains the pixel color at (x, y)
-------------------------------------------------------------------------
*/

BOOL DibGetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb)
{
      DWORD                   dwPixel ;
      int                     iBitCount ;
      PDIBSTRUCT  pdib = hdib ;

                        // Get bit count; also use this as a validity check

      if (0 == (iBitCount = DibBitCount (hdib)))
                        return FALSE ;
                        // Get the pixel value
      dwPixel = DibGetPixel (hdib, x, y) ;
                        // If the bit-count is 8 or less, index the color table
      if (iBitCount <= 8)
                        return DibGetColor (hdib, (int) dwPixel, prgb) ;
                        // If the bit-count is 24, just use the pixel
      else  if (iBitCount == 24)
     {
                  *     (RGBTRIPLE *) prgb = * (RGBTRIPLE *) & dwPixel ;
                        prgb->rgbReserved = 0 ;
      }

                  // If the bit-count is 32 and the biCompression field is BI_RGB,
                  //   just use the pixel

      else  if (iBitCount == 32 &&
                        pdib->ds.dsBmih.biCompression == BI_RGB)
      {
                  *     prgb = * (RGBQUAD *) & dwPixel ;
      }

                  // Otherwise, use the mask and shift values
                  //   (for best performance, don't use DibMask and DibShift
```

```
functions)
      else
      {
            prgb->rgbRed = (BYTE)(((pdib->ds.dsBitfields[0] & dwPixel)
                  >> pdib->iRShift[0]) << pdib->iLShift[0]) ;

            prgb->rgbGreen=(BYTE((pdib->ds.dsBitfields[1] & dwPixel)
                  >> pdib->iRShift[1]) << pdib->iLShift[1]) ;

            prgb->rgbBlue=(BYTE)(((pdib->ds.dsBitfields[2] & dwPixel)
                  >> pdib->iRShift[2]) << pdib->iLShift[2]) ;
      }
      return TRUE ;
}

/*-------------------------------------------------------------------------
---
      DibSetPixelColor:  Sets the pixel color at (x, y)
-------------------------------------------------------------------------
-*/

BOOL DibSetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb)
{
      DWORD                   dwPixel ;
      int                     iBitCount ;
      PDIBSTRUCT  pdib = hdib ;

                  // Don't do this function for DIBs with color tables

      iBitCount = DibBitCount (hdib) ;
      if (iBitCount <= 8)
                        return FALSE ;
                  // The rest is just the opposite of DibGetPixelColor
      else if (iBitCount == 24)
      {
                        * (RGBTRIPLE *) & dwPixel = * (RGBTRIPLE *) prgb ;
                        dwPixel &= 0x00FFFFFF ;
      }
      else if (iBitCount == 32 &&
                        pdib->ds.dsBmih.biCompression == BI_RGB)
    {
                        * (RGBQUAD *) & dwPixel = * prgb ;
    }

      else
    {
       dwPixel  =       (((DWORD)   prgb->rgbRed >> pdib->iLShift[0])
       << pdib->iRShift[0]) ;
```

```
        dwPixel |=      (((DWORD)   prgb->rgbGreen >> pdib->iLShift[1])
        << pdib->iRShift[1]) ;

        dwPixel |=      (((DWORD)   prgb->rgbBlue >> pdib->iLShift[2])
        << pdib->iRShift[2]) ;
       }

      DibSetPixel (hdib, x, y, dwPixel) ;
      return TRUE ;
}
```

这部分 DIBHELP.C 从 DibPixelPtr 函式开始，该函式获得指向储存（或部分储存）有特殊图素的位元组的指标。回想一下 DIBSTRUCT 结构的 ppRow 栏位，那是个指向 DIB 中由顶行开始排列的图素行位址的指标。这样，

```
((PDIBSTRUCT) hdib)->pprow)[0]
```

就是指向 DIB 顶行最左端图素的指标，而

```
(((PDIBSTRUCT) hdib)->ppRow)[y] + (x * DibBitCount (hdib) >> 3)
```

是指向位於(x,y)的图素的指标。注意，如果 DIB 中的图素不可被定址（即如果已压缩），或者如果函式的 x 和 y 参数是负数或相对位於 DIB 外面的区域，则函式将传回 NULL。此检查降低了函式（和所有依赖於 DibPixelPtr 的函式）的执行速度，下面我将讲述一些更快的常式。

档案後面的 DibGetPixel 和 DibSetPixel 函式利用了 DibPixelPtr。对於 8 位元、16 位元和 32 位元 DIB，这些函式只记录指向合适资料尺寸的指标，并存取图素值。对於 1 位元和 4 位元的 DIB，则需要遮罩和移位角度。

DibGetColor 函式按 RGBQUAD 结构获得图素颜色。对於 1 位元、4 位元和 8 位元 DIB，这包括使用图素值来从 DIB 颜色表获得颜色。对於 16 位元、24 位元和 32 位元 DIB，通常必须将图素值遮罩和移位以得到 RGB 颜色。DibSetPixel 函式则相反，它允许从 RGBQUAD 结构设定图素值。该函式只为 16 位元、24 位元和 32 位元 DIB 定义。

建立和转换

程式 16-21 所示的 DIBHELP 第三部分和最後部分展示了如何建立 DIB 区块，以及如何将 DIB 区块与 packed DIB 相互转换。

程式 16-21　DIBHELP.C 档案的第三部分和最後部分

```
DIBHELP.C （第三部分）
/*-------------------------------------------------------------------------
      Calculating shift values from color masks is required by the
      DibCreateFromInfo function.
-------------------------------------------------------------------------
```

```
*/

static int MaskToRShift (DWORD dwMask)
{
      int iShift ;
      if (dwMask == 0)
                        return 0 ;
      for (iShift = 0 ; !(dwMask & 1) ; iShift++)
                        dwMask >>= 1 ;
      return iShift ;
}

static int MaskToLShift (DWORD dwMask)
{
      int iShift ;
      if (dwMask == 0)
                        return 0 ;
      while (!(dwMask & 1))
                        dwMask >>= 1 ;
      for (iShift = 0 ; dwMask & 1 ; iShift++)
                        dwMask >>= 1 ;
      return 8 - iShift ;
}
/*-------------------------------------------------------------------------
--
      DibCreateFromInfo: All DIB creation functions ultimately call this one.
      This function is responsible for calling CreateDIBSection, allocating
      memory for DIBSTRUCT, and setting up the row pointer.
-------------------------------------------------------------------------
-*/

HDIB DibCreateFromInfo (BITMAPINFO * pbmi)
{
      BYTE                    *     pBits ;
      DIBSTRUCT   *     pdib ;
      HBITMAP                 hBitmap ;
      int                           i, iRowLength, cy, y ;

      hBitmap = CreateDIBSection (NULL, pbmi, DIB_RGB_COLORS, &pBits, NULL, 0) ;
      if (hBitmap == NULL)
                        return NULL ;
      if (NULL == (pdib = malloc (sizeof (DIBSTRUCT))))
      {
                        DeleteObject (hBitmap) ;
                        return NULL ;
      }

      pdib->iSignature = HDIB_SIGNATURE ;
```

```
     pdib->hBitmap          = hBitmap ;
     pdib->pBits      = pBits ;

     GetObject (hBitmap, sizeof (DIBSECTION), &pdib->ds) ;
                // Notice that we can now use the DIB information functions
                //   defined above.

                // If the compression is BI_BITFIELDS, calculate shifts from
masks

     if (DibCompression (pdib) == BI_BITFIELDS)
     {
                for (i = 0 ; i < 3 ; i++)
                {
                           pdib->iLShift[i]          =          MaskToLShift
(pdib->ds.dsBitfields[i]) ;
                           pdib->iRShift[i]          =          MaskToRShift
(pdib->ds.dsBitfields[i]) ;
                }
     }

                // If the compression is BI_RGB, but bit-count is 16 or 32,
                //   set the bitfields and the masks
     else if (DibCompression (pdib) == BI_RGB)
     {
                if (DibBitCount (pdib) == 16)
                {
                           pdib->ds.dsBitfields[0] = 0x00007C00 ;
                           pdib->ds.dsBitfields[1] = 0x000003E0 ;
                           pdib->ds.dsBitfields[2] = 0x0000001F ;

                           pdib->iRShift    [0]   =          10    ;
                           pdib->iRShift    [1]   =          5     ;
                           pdib->iRShift    [2]   =          0     ;

                           pdib->iLShift    [0]   =          3     ;
                           pdib->iLShift    [1]   =          3     ;
                           pdib->iLShift    [2]   =          3     ;
                }
                else if (DibBitCount (pdib) == 24 || DibBitCount (pdib) == 32)
                {
                           pdib->ds.dsBitfields[0] = 0x00FF0000 ;
                           pdib->ds.dsBitfields[1] = 0x0000FF00 ;
                           pdib->ds.dsBitfields[2] = 0x000000FF ;

                           pdib->iRShift    [0]   =          16    ;
                           pdib->iRShift    [1]   =          8     ;
                           pdib->iRShift    [2]   =          0     ;
```

```
                              pdib->iLShift    [0]   =          0     ;
                              pdib->iLShift    [1]   =          0     ;
                              pdib->iLShift    [2]   =          0     ;
                    }
      }
                    // Allocate an array of pointers to each row in the DIB
      cy = DibHeight (pdib) ;
      if (NULL == (pdib->ppRow = malloc (cy * sizeof (BYTE *))))
      {
                        free (pdib) ;
                        DeleteObject (hBitmap) ;
                        return NULL ;
      }

                    // Initialize them.
      iRowLength = DibRowLength (pdib) ;
      if (pbmi->bmiHeader.biHeight > 0)                    // ie, bottom up
      {
                        for (y = 0 ; y < cy ; y++)
                              pdib->ppRow[y] = pBits + (cy - y - 1) * iRowLength ;
      }
      else

// top down
     {
                        for (y = 0 ; y < cy ; y++)
                              pdib->ppRow[y] = pBits + y * iRowLength ;
      }
      return pdib ;
 }

/*-------------------------------------------------------------------------
      DibDelete:  Frees all memory for the DIB section
-------------------------------------------------------------------------
*/

BOOL DibDelete (HDIB hdib)
{
      DIBSTRUCT * pdib = hdib ;
      if (!DibIsValid (hdib))
                        return FALSE ;
      free (pdib->ppRow) ;
      DeleteObject (pdib->hBitmap) ;
      free (pdib) ;
      return TRUE ;
}
```

```
/*-------------------------------------------------------------------------
--
      DibCreate: Creates an HDIB from explicit arguments
-------------------------------------------------------------------------
-*/

HDIB DibCreate (int cx, int cy, int cBits, int cColors)
{
      BITMAPINFO *     pbmi ;
      DWORD                              dwInfoSize ;
      HDIB                               hDib ;
      int                          cEntries ;

      if (cx <= 0 || cy <= 0 ||
                       ((cBits !=  1) && (cBits !=  4) && (cBits !=  8) &&
                             (cBits != 16) && (cBits != 24) && (cBits != 32)))
      {
                  return NULL ;
      }

      if (  cColors != 0)
                       cEntries = cColors ;
      else  if (cBits <= 8)
                       cEntries = 1 << cBits ;

      dwInfoSize = sizeof (BITMAPINFOHEADER) + (cEntries - 1) * sizeof (RGBQUAD);

      if (NULL == (pbmi = malloc (dwInfoSize)))
      {
                  return NULL ;
      }

      ZeroMemory (pbmi, dwInfoSize) ;

      pbmi->bmiHeader.biSize                  = sizeof (BITMAPINFOHEADER) ;
      pbmi->bmiHeader.biWidth                 = cx ;
      pbmi->bmiHeader.biHeight                = cy ;
      pbmi->bmiHeader.biPlanes                = 1 ;
      pbmi->bmiHeader.biBitCount              = cBits ;
      pbmi->bmiHeader.biCompression           = BI_RGB ;
      pbmi->bmiHeader.biSizeImage             = 0 ;
      pbmi->bmiHeader.biXPelsPerMeter         = 0 ;
      pbmi->bmiHeader.biYPelsPerMeter         = 0 ;
      pbmi->bmiHeader.biClrUsed               = cColors ;
      pbmi->bmiHeader.biClrImportant          = 0 ;

      hDib = DibCreateFromInfo (pbmi) ;
      free (pbmi) ;
```

```
     return hDib ;
}

/*-------------------------------------------------------------------------
--
     DibCopyToInfo:         Builds BITMAPINFO structure.
                                                  Used   by   DibCopy   and
DibCopyToDdb
-------------------------------------------------------------------------
-*/

static BITMAPINFO * DibCopyToInfo (HDIB hdib)
{
     BITMAPINFO       *     pbmi ;
     int                              i, iNumColors ;
     RGBQUAD                *     prgb ;
     if (!DibIsValid (hdib))
                      return NULL ;
                      // Allocate the memory
     if (NULL == (pbmi = malloc (DibInfoSize (hdib))))
                      return NULL ;
                      // Copy the information header
     CopyMemory (pbmi, DibInfoHeaderPtr(hdib), sizeof (BITMAPINFOHEADER));

                      // Copy the possible color masks

     prgb = (RGBQUAD *) ((BYTE *) pbmi + sizeof (BITMAPINFOHEADER)) ;
     if (DibMaskSize (hdib))
     {
                      CopyMemory (prgb, DibMaskPtr (hdib), 3 * sizeof (DWORD)) ;
                      prgb = (RGBQUAD *) ((BYTE *) prgb + 3 * sizeof (DWORD)) ;
     }
                      // Copy the color table
     iNumColors = DibNumColors (hdib) ;
     for (i = 0 ; i < iNumColors ; i++)
                      DibGetColor (hdib, i, prgb + i) ;
     return pbmi ;
}

/*-------------------------------------------------------------------------
     DibCopy:  Creates a new DIB section from an existing DIB section,
          possibly swapping the DIB width and height.
-------------------------------------------------------------------------*
/

HDIB DibCopy (HDIB hdibSrc, BOOL fRotate)
{
```

```
     BITMAPINFO       *     pbmi ;
     BYTE             *     pBitsSrc, * pBitsDst ;
     HDIB             hdibDst ;

     if (!DibIsValid (hdibSrc))
                      return NULL ;
     if (NULL == (pbmi = DibCopyToInfo (hdibSrc)))
                      return NULL ;
     if (fRotate)
     {
                      pbmi->bmiHeader.biWidth = DibHeight (hdibSrc) ;
                      pbmi->bmiHeader.biHeight = DibWidth (hdibSrc) ;
     }

     hdibDst = DibCreateFromInfo (pbmi) ;
     free (pbmi) ;

     if (  hdibDst == NULL)
                      return NULL ;

                           // Copy the bits

     if (!fRotate)
     {
                           pBitsSrc = DibBitsPtr (hdibSrc) ;
                           pBitsDst = DibBitsPtr (hdibDst) ;

                           CopyMemory   (pBitsDst,   pBitsSrc,   DibBitsSize
(hdibSrc)) ;
     }
     return hdibDst ;
}

/*-------------------------------------------------------------------------
--
     DibCopyToPackedDib is generally used for saving DIBs and for
     transferring DIBs to the clipboard. In the second case, the second
     argument should be set to TRUE so that the memory is allocated
     with the GMEM_SHARE flag.
-------------------------------------------------------------------------
-*/

BITMAPINFO * DibCopyToPackedDib (HDIB hdib, BOOL fUseGlobal)
{
     BITMAPINFO       *     pPackedDib ;
     BYTE             *     pBits ;
     DWORD                  dwDibSize ;
     HDC              hdcMem ;
```

```
     HGLOBAL               hGlobal ;
     int              iNumColors ;
     PDIBSTRUCT            pdib = hdib ;
     RGBQUAD          *    prgb ;

     if (!DibIsValid (hdib))
                      return NULL ;
                           // Allocate memory for packed DIB
     dwDibSize = DibTotalSize (hdib) ;
     if (fUseGlobal)
     {
                      hGlobal = GlobalAlloc (GHND | GMEM_SHARE, dwDibSize) ;
                      pPackedDib = GlobalLock (hGlobal) ;
     }
     else
     {
                      pPackedDib = malloc (dwDibSize) ;
     }

     if (pPackedDib == NULL)
                      return NULL ;
                      // Copy the information header
     CopyMemory (pPackedDib, &pdib->ds.dsBmih, sizeof (BITMAPINFOHEADER)) ;
     prgb = (RGBQUAD *) ((BYTE *) pPackedDib + sizeof (BITMAPINFOHEADER)) ;
                      // Copy the possible color masks
     if (pdib->ds.dsBmih.biCompression == BI_BITFIELDS)
     {
                CopyMemory (prgb, pdib->ds.dsBitfields, 3 * sizeof (DWORD)) ;
                prgb = (RGBQUAD *) ((BYTE *) prgb + 3 * sizeof (DWORD)) ;
     }
                      // Copy the color table
     if (iNumColors = DibNumColors (hdib))
     {
                      hdcMem = CreateCompatibleDC (NULL) ;
                      SelectObject (hdcMem, pdib->hBitmap) ;
                      GetDIBColorTable (hdcMem, 0, iNumColors, prgb) ;
                      DeleteDC (hdcMem) ;
     }

     pBits = (BYTE *) (prgb + iNumColors) ;
                // Copy the bits
     CopyMemory (pBits, pdib->pBits, DibBitsSize (pdib)) ;
                // If last argument is TRUE, unlock global memory block and
                //   cast it to pointer in preparation for return

     if (fUseGlobal)
     {
                      GlobalUnlock (hGlobal) ;
```

```
                        pPackedDib = (BITMAPINFO *) hGlobal ;
      }
      return pPackedDib ;
}

/*-------------------------------------------------------------------------
   DibCopyFromPackedDib is generally used for pasting DIBs from the
     clipboard.
  -----------------------------------------------------------------------*/

HDIB DibCopyFromPackedDib (BITMAPINFO * pPackedDib)
{
      BYTE                    *     pBits ;
      DWORD                         dwInfoSize, dwMaskSize, dwColorSize ;
      int                           iBitCount ;
      PDIBSTRUCT        pdib ;

                        // Get the size of the information header and do validity
check

      dwInfoSize = pPackedDib->bmiHeader.biSize ;
      if (  dwInfoSize != sizeof (BITMAPCOREHEADER) &&
                        dwInfoSize != sizeof (BITMAPINFOHEADER) &&
                        dwInfoSize != sizeof (BITMAPV4HEADER) &&
                        dwInfoSize != sizeof (BITMAPV5HEADER))
      {
                        return NULL ;
      }
                        // Get the possible size of the color masks

      if (dwInfoSize == sizeof (BITMAPINFOHEADER) &&
                        pPackedDib->bmiHeader.biCompression == BI_BITFIELDS)
      {
                        dwMaskSize = 3 * sizeof (DWORD) ;
      }
      else
      {
                        dwMaskSize = 0 ;
      }
                        // Get the size of the color table
      if (dwInfoSize == sizeof (BITMAPCOREHEADER))
      {
                  iBitCount = ((BITMAPCOREHEADER *) pPackedDib)->bcBitCount ;

                        if (iBitCount <= 8)
                        {
                        dwColorSize = (1 << iBitCount) * sizeof (RGBTRIPLE) ;
                        }
```

```
                        else
                                        dwColorSize = 0 ;
      }
      else                              // all non-OS/2 compatible DIBs
      {
                        if (pPackedDib->bmiHeader.biClrUsed > 0)
                        {
            dwColorSize = pPackedDib->bmiHeader.biClrUsed * sizeof (RGBQUAD);
                        }
                        else if (pPackedDib->bmiHeader.biBitCount <= 8)
                        {
                                        dwColorSize       =       (1       <<
pPackedDib->bmiHeader.biBitCount) * sizeof (RGBQUAD) ;
                        }
                        else
                        {
                                        dwColorSize = 0 ;
                        }
      }
                        // Finally, get the pointer to the bits in the packed DIB
      pBits = (BYTE *) pPackedDib + dwInfoSize + dwMaskSize + dwColorSize ;
                        // Create the HDIB from the packed-DIB pointer
      pdib = DibCreateFromInfo (pPackedDib) ;
                        // Copy the pixel bits
      CopyMemory (pdib->pBits, pBits, DibBitsSize (pdib)) ;
      return pdib ;
}

/*-------------------------------------------------------------------------
--
      DibFileLoad:  Creates a DIB section from a DIB file
-------------------------------------------------------------------------
-*/
HDIB DibFileLoad (const TCHAR * szFileName)
{
      BITMAPFILEHEADER       bmfh ;
      BITMAPINFO                    *     pbmi ;
      BOOL                                        bSuccess ;
      DWORD                                       dwInfoSize,   dwBitsSize,
dwBytesRead ;
      HANDLE                                hFile ;
      HDIB                                        hDib ;

            // Open the file: read access, prohibit write access

      hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHARE_READ, NULL,
                        OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
      if (hFile == INVALID_HANDLE_VALUE)
```

```
                    return NULL ;
                    // Read in the BITMAPFILEHEADER
     bSuccess = ReadFile (  hFile, &bmfh, sizeof (BITMAPFILEHEADER),
                                 &dwBytesRead, NULL) ;

     if (!bSuccess    || (dwBytesRead != sizeof (BITMAPFILEHEADER))
                                 || (bmfh.bfType != * (WORD *) "BM"))
     {
                    CloseHandle (hFile) ;
                    return NULL ;
    }
                // Allocate memory for the information structure & read it in
     dwInfoSize = bmfh.bfOffBits - sizeof (BITMAPFILEHEADER) ;
     if (NULL == (pbmi = malloc (dwInfoSize)))
     {
                    CloseHandle (hFile) ;
                    return NULL ;
     }

     bSuccess = ReadFile (hFile, pbmi, dwInfoSize, &dwBytesRead, NULL) ;
     if (!bSuccess || (dwBytesRead != dwInfoSize))
    {
                CloseHandle (hFile) ;
                free (pbmi) ;
                return NULL ;
     }
                // Create the DIB
     hDib = DibCreateFromInfo (pbmi) ;
     free (pbmi) ;

     if (hDib == NULL)
     {
                CloseHandle (hFile) ;
                return NULL ;
     }
                // Read in the bits
     dwBitsSize = bmfh.bfSize - bmfh.bfOffBits ;
     bSuccess = ReadFile (  hFile, ((PDIBSTRUCT) hDib)->pBits,
                           dwBitsSize, &dwBytesRead, NULL) ;
     CloseHandle (hFile) ;
     if (!bSuccess || (dwBytesRead != dwBitsSize))
     {
                DibDelete (hDib) ;
                return NULL ;
     }
     return hDib ;
}
```

```
/*------------------------------------------------------------------------
      DibFileSave:  Saves a DIB section to a file
------------------------------------------------------------------------
*/

BOOL DibFileSave (HDIB hdib, const TCHAR * szFileName)
{
      BITMAPFILEHEADER       bmfh ;
      BITMAPINFO                   *     pbmi ;
      BOOL                   bSuccess ;
      DWORD                  dwTotalSize, dwBytesWritten ;
      HANDLE                 hFile ;

      hFile = CreateFile (szFileName, GENERIC_WRITE, 0, NULL,
                              CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL) ;
      if (hFile == INVALID_HANDLE_VALUE)
                        return FALSE ;
      dwTotalSize                          = DibTotalSize (hdib) ;
      bmfh.bfType                 = * (WORD *) "BM" ;
      bmfh.bfSize                 = sizeof (BITMAPFILEHEADER) + dwTotalSize ;
      bmfh.bfReserved1       = 0 ;
      bmfh.bfReserved2       = 0 ;
      bmfh.bfOffBits         = bmfh.bfSize - DibBitsSize (hdib) ;

                        // Write the BITMAPFILEHEADER

      bSuccess = WriteFile ( hFile, &bmfh, sizeof (BITMAPFILEHEADER),
                              &dwBytesWritten, NULL) ;

      if (!bSuccess || (dwBytesWritten != sizeof (BITMAPFILEHEADER)))
      {
                        CloseHandle (hFile) ;
                        DeleteFile (szFileName) ;
                        return FALSE ;
      }
                        // Get entire DIB in packed-DIB format
      if (NULL == (pbmi = DibCopyToPackedDib (hdib, FALSE)))
     {
                        CloseHandle (hFile) ;
                        DeleteFile (szFileName) ;
                        return FALSE ;
      }
                        // Write out the packed DIB
      bSuccess = WriteFile (hFile, pbmi, dwTotalSize, &dwBytesWritten, NULL) ;
      CloseHandle (hFile) ;
      free (pbmi) ;

      if (!bSuccess || (dwBytesWritten != dwTotalSize))
```

```
      {
                        DeleteFile (szFileName) ;
                        return FALSE ;
      }
      return TRUE ;
}

/*-------------------------------------------------------------------------
-
      DibCopyToDdb:  For more efficient screen displays
-------------------------------------------------------------------------*
/
HBITMAP DibCopyToDdb (HDIB hdib, HWND hwnd, HPALETTE hPalette)
{
      BITMAPINFO        *     pbmi ;
      HBITMAP                           hBitmap ;
      HDC                               hdc ;

      if (!DibIsValid (hdib))
                        return NULL ;
      if (NULL == (pbmi = DibCopyToInfo (hdib)))
                        return NULL ;
      hdc = GetDC (hwnd) ;
      if (hPalette)
      {
                        SelectPalette (hdc, hPalette, FALSE) ;
                        RealizePalette (hdc) ;
      }

      hBitmap = CreateDIBitmap (hdc, DibInfoHeaderPtr (hdib), CBM_INIT,
                                                      DibBitsPtr(hdib),pbmi,
DIB_RGB_COLORS) ;

      ReleaseDC (hwnd, hdc) ;
      free (pbmi) ;

      return hBitmap ;
}
```

这部分的 DIBHELP.C 档案从两个小函式开始，这两个函式根据 16 位元和 32 位元 DIB 的颜色遮罩得到左、右移位值。这些函式在第十五章「颜色遮罩」一节说明。

DibCreateFromInfo 函式是 DIBHELP 中唯一呼叫 CreateDIBSection 并为 DIBSTRUCT 结构配置记忆体的函式。其他所有建立和复制函式都重复此函式。DibCreateFromInfo 唯一的参数是指向 BITMAPINFO 结构的指标。此结构的颜色表必须存在，但是它不必用有效的值填充。呼叫 CreateDIBSection 之後，该函

式将初始化 DIBSTRUCT 结构的所有栏位。注意，在设定 DIBSTRUCT 结构的 ppRow 栏位的值时（指向 DIB 行位址的指标），DIB 有由下而上和由上而下的不同储存方式。ppRow 开头的元素就是 DIB 的顶行。

DibDelete 删除 DibCreateFromInfo 中建立的点阵图，同时释放在该函式中配置的记忆体。

DibCreate 可能比 DibCreateFromInfo 更像一个从应用程式呼叫的函式。前三个参数提供图素的宽度、高度和每图素的位数。最後一个参数可以设定为 0（用於颜色表的内定尺寸），或者设定为非 0（表示比每图素位元数所需要的颜色表更小的颜色表）。

DibCopy 函式根据现存的 DIB 区块建立新的 DIB 区块，并用 DibCreateInfo 函式为 BITMAPINFO 结构配置了记忆体，还填了所有的资料。DibCopy 函式的一个 BOOL 参数指出是否在建立新的 DIB 时交换了 DIB 的宽度和高度。我们将在後面看到此函式的用法。

DibCopyToPackedDib 和 DibCopyFromPackedDib 函式的使用通常与透过剪贴簿传递 DIB 相关。DibFileLoad 函式从 DIB 档案建立 DIB 区块；DibFileSave 函式将资料储存到 DIB 档案。

最後，DibCopyToDdb 函式根据 DIB 建立 GDI 点阵图物件。注意，该函式需要目前调色盘的代号和程式视窗的代号。程式视窗代号用於获得选进并显现调色盘的装置内容。只有这样，函式才可以呼叫 CreateDIBitmap。这曾在本章前面的 SHOWDIB7 中展示。

DIBHELP 表头档案和巨集

DIBHELP.H 表头档案如程式 16-22 所示。

程式 16-22　DIBHELP.H 档案

```
DIBHELP.H
/*------------------------------------------------------------------------
      DIBHELP.H header file for DIBHELP.C
------------------------------------------------------------------------
*/

typedef void * HDIB ;
      // Functions in DIBHELP.C
BOOL DibIsValid (HDIB hdib) ;
HBITMAP DibBitmapHandle (HDIB hdib) ;
int DibWidth (HDIB hdib) ;
int DibHeight (HDIB hdib) ;
int DibBitCount (HDIB hdib) ;
int DibRowLength (HDIB hdib) ;
```

```
int DibNumColors (HDIB hdib) ;
DWORD DibMask (HDIB hdib, int i) ;
int DibRShift (HDIB hdib, int i) ;
int DibLShift (HDIB hdib, int i) ;
int DibCompression (HDIB hdib) ;
BOOL DibIsAddressable (HDIB hdib) ;
DWORD DibInfoHeaderSize (HDIB hdib) ;
DWORD DibMaskSize (HDIB hdib) ;
DWORD DibColorSize (HDIB hdib) ;
DWORD DibInfoSize (HDIB hdib) ;
DWORD DibBitsSize (HDIB hdib) ;
DWORD DibTotalSize (HDIB hdib) ;
BITMAPINFOHEADER * DibInfoHeaderPtr (HDIB hdib) ;
DWORD * DibMaskPtr (HDIB hdib) ;
void * DibBitsPtr (HDIB hdib) ;
BOOL DibGetColor (HDIB hdib, int index, RGBQUAD * prgb) ;
BOOL DibSetColor (HDIB hdib, int index, RGBQUAD * prgb) ;
BYTE * DibPixelPtr (HDIB hdib, int x, int y) ;
DWORD DibGetPixel (HDIB hdib, int x, int y) ;
BOOL DibSetPixel (HDIB hdib, int x, int y, DWORD dwPixel) ;
BOOL DibGetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb) ;
BOOL DibSetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb) ;
HDIB DibCreateFromInfo (BITMAPINFO * pbmi) ;
BOOL DibDelete (HDIB hdib) ;
HDIB DibCreate (int cx, int cy, int cBits, int cColors) ;
HDIB DibCopy (HDIB hdibSrc, BOOL fRotate) ;
BITMAPINFO * DibCopyToPackedDib (HDIB hdib, BOOL fUseGlobal) ;
HDIB DibCopyFromPackedDib (BITMAPINFO * pPackedDib) ;
HDIB DibFileLoad (const TCHAR * szFileName) ;
BOOL DibFileSave (HDIB hdib, const TCHAR * szFileName) ;
HBITMAP DibCopyToDdb (HDIB hdib, HWND hwnd, HPALETTE hPalette) ;
HDIB DibCreateFromDdb (HBITMAP hBitmap) ;

/*-------------------------------------------------------------------------
-
      Quickie no-bounds-checked pixel gets and sets
-------------------------------------------------------------------------
-*/

#define     DibPixelPtr1(hdib, x, y)     (((* (PBYTE **) hdib) [y]) + ((x) >> 3))
#define     DibPixelPtr4(hdib, x, y)     (((* (PBYTE **) hdib) [y]) + ((x) >> 1))
#define     DibPixelPtr8(hdib, x, y)     (((* (PBYTE **) hdib) [y]) +  (x)
      )
#define     DibPixelPtr16(hdib, x, y)  \
                                          ((WORD *) (((* (PBYTE **) hdib) [y])
+  (x) *  2))

#define DibPixelPtr24(hdib, x, y)  \
```

```
                                                ((RGBTRIPLE *) (((* (PBYTE **)
hdib) [y]) +  (x) *  3))
#define DibPixelPtr32(hdib, x, y)  \
                                                      ((DWORD *) (((* (PBYTE **)
hdib) [y]) +  (x) *  4))

#define DibGetPixel1(hdib, x, y)   \
                                    (0x01 & (* DibPixelPtr1 (hdib, x, y) >> (7 -
((x) & 7))))

#define DibGetPixel4(hdib, x, y)   \
                                    (0x0F & (* DibPixelPtr4 (hdib, x, y) >> ((x)
& 1 ? 0 : 4)))

#define     DibGetPixel8(hdib, x, y)                (* DibPixelPtr8
      (hdib, x, y))
#define     DibGetPixel16(hdib, x, y)    (* DibPixelPtr16 (hdib, x, y))
#define     DibGetPixel24(hdib, x, y)    (* DibPixelPtr24 (hdib, x, y))
#define     DibGetPixel32(hdib, x, y)    (* DibPixelPtr32 (hdib, x, y))

#define DibSetPixel1(hdib, x, y, p)
                                        \
                  ((* DibPixelPtr1 (hdib, x, y) &= ~( 1   << (7 - ((x) & 7)))),
                  \
                  (* DibPixelPtr1 (hdib, x, y) |=    ((p) << (7 - ((x) & 7)))))

#define DibSetPixel4(hdib, x, y, p)
                                        \
                  ((* DibPixelPtr4 (hdib, x, y) &= (0x0F <<     ((x) & 1 ? 4 :
0))), \
                  (* DibPixelPtr4 (hdib, x, y) |= ((p)    <<    ((x) & 1 ? 0 :
4))))

#define DibSetPixel8(hdib, x, y, p)  (* DibPixelPtr8 (hdib, x, y) = p)
#define DibSetPixel16(hdib, x, y, p) (* DibPixelPtr16 (hdib, x, y) = p)
#define DibSetPixel24(hdib, x, y, p) (* DibPixelPtr24 (hdib, x, y) = p)
#define DibSetPixel32(hdib, x, y, p) (* DibPixelPtr32 (hdib, x, y) = p)
```

这个表头档案将 HDIB 定义为空指标(void*)。应用程式的确不需要了解 HDIB 所指结构的内部结构。此表头档案还包括 DIBHELP.C 中所有函式的说明，还有一些巨集——非常特殊的巨集。

如果再看一看 DIBHELP.C 中的 DibPixelPtr、DibGetPixel 和 DibSetPixel 函式，并试图提高它们的执行速度表现，那么您将看到两种可能的解决方法。第一种，可以删除所有的检查保护，并相信应用程式不会使用无效参数呼叫函式。还可以删除一些函式呼叫，例如 DibBitCount，并使用指向 DIBSTRUCT 结构内部的指标来直接获得资讯。

提高执行速度表现另一项较不明显的方法是删除所有对每图素位元数的处理方式，同时分离出处理不同 DIB 函式——例如 DibGetPixel1、DibGetPixel4、DibGetPixel8 等等。下一个最佳化步骤是删除整个函式呼叫，将其处理动作透过 inline function 或巨集中进行合并。

DIBHELP.H 采用巨集的方法。它依据 DibPixelPtr、DibGetPixel 和 DibSetPixel 函式提出了三套巨集。这些巨集都明确对应於特殊的图素位元数。

DIBBLE 程式

DIBBLE 程式，如程式 16-23 所示，使用 DIBHELP 函式和巨集工作。尽管 DIBBLE 是本书中最长的程式，它确实只是一些作业的粗略范例，这些作业可以在简单的数位影像处理程式中找到。对 DIBBLE 的明显改进是转换成了多重文件介面（MDI：multiple document interface），我们将在第十九章学习有关多重文件介面的知识。

程式 16-23 DIBBLE

```
DIBBLE.C
/*-----------------------------------------------------------------------
-
      DIBBLE.C --      Bitmap and Palette Program
                                          (c) Charles Petzold, 1998
-----------------------------------------------------------------------
-*/

#include <windows.h>
#include "dibhelp.h"
#include "dibpal.h"
#include "dibconv.h"
#include "resource.h"

#define WM_USER_SETSCROLLS              (WM_USER + 1)
#define WM_USER_DELETEDIB               (WM_USER + 2)
#define WM_USER_DELETEPAL               (WM_USER + 3)
#define WM_USER_CREATEPAL               (WM_USER + 4)

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("Dibble") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                PSTR    szCmdLine,    int
iCmdShow)
{
      HACCEL            hAccel ;
      HWND              hwnd ;
      MSG               msg ;
```

```
     WNDCLASS         wndclass ;

     wndclass.style                    = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor                  = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground       = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName        = szAppName ;
     wndclass.lpszClassName       = szAppName ;

     if (!RegisterClass (&wndclass))
     {
                    MessageBox (     NULL,  TEXT  ("This  program  requires
Windows NT!"),
                                                              szAppName,
MB_ICONERROR) ;
                    return 0 ;
     }

     hwnd  =     CreateWindow (   szAppName, szAppName,
            WS_OVERLAPPEDWINDOW | WM_VSCROLL | WM_HSCROLL,
            CW_USEDEFAULT, CW_USEDEFAULT,
            CW_USEDEFAULT, CW_USEDEFAULT,
            NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     hAccel = LoadAccelerators (hInstance, szAppName) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
               if (!TranslateAccelerator (hwnd, hAccel, &msg))
               {
                              TranslateMessage (&msg) ;
                              DispatchMessage (&msg) ;
               }
     }
     return msg.wParam ;
}

/*---------------------------------------------------------------------------
--
     DisplayDib:           Displays or prints DIB actual size or stretched
                                        depending on menu selection
```

```
------------------------------------------------------------------------
-*/

int DisplayDib ( HDC hdc, HBITMAP hBitmap, int x, int y,
                                          int cxClient, int cyClient,
                                          WORD        wShow,        BOOL
fHalftonePalette)
{
      BITMAP           bitmap ;
      HDC                    hdcMem ;
      int                    cxBitmap, cyBitmap, iReturn ;
      GetObject   (hBitmap, sizeof (BITMAP), &bitmap) ;
      cxBitmap         =     bitmap.bmWidth ;
      cyBitmap         =     bitmap.bmHeight ;

      SaveDC (hdc) ;
      if (fHalftonePalette)
                       SetStretchBltMode (hdc, HALFTONE) ;
      else
                       SetStretchBltMode (hdc, COLORONCOLOR) ;
      hdcMem = CreateCompatibleDC (hdc) ;
      SelectObject (hdcMem, hBitmap) ;

      switch (wShow)
      {
      case  IDM_SHOW_NORMAL:
            if (fHalftonePalette)
                  iReturn = StretchBlt (hdc,   0, 0,
                              min (cxClient, cxBitmap - x),
                              min (cyClient, cyBitmap - y),
                              hdcMem, x, y, min (cxClient, cxBitmap - x),
                              min (cyClient, cyBitmap - y),
                              SRCCOPY);
            else
                  iReturn = BitBlt (hdc,0, 0, min (cxClient,
                              cxBitmap - x),
                              min (cyClient, cyBitmap - y),
                              hdcMem, x, y, SRCCOPY) ;
                       break ;

      case  IDM_SHOW_CENTER:
                       if (fHalftonePalette)
                       iReturn = StretchBlt ( hdc, (cxClient - cxBitmap) / 2,
                                (cyClient - cyBitmap) / 2,
                                 cxBitmap, cyBitmap,
                                 hdcMem, 0, 0, cxBitmap, cyBitmap, SRCCOPY);
                       else
                              iReturn = BitBlt (hdc, (cxClient - cxBitmap) / 2,
```

```
                        cyClient - cyBitmap) / 2,
                        cxBitmap, cyBitmap,
                        hdcMem, 0, 0, SRCCOPY) ;
                       break ;
     case  IDM_SHOW_STRETCH:
                        iReturn = StretchBlt (hdc,   0, 0, cxClient, cyClient,
               hdcMem,     0, 0, cxBitmap, cyBitmap, SRCCOPY) ;
                        break ;

     case  IDM_SHOW_ISOSTRETCH:
                        SetMapMode (hdc, MM_ISOTROPIC) ;
                        SetWindowExtEx (hdc, cxBitmap, cyBitmap, NULL) ;
                        SetViewportExtEx (hdc, cxClient, cyClient, NULL) ;
                        SetWindowOrgEx (hdc, cxBitmap / 2, cyBitmap / 2, NULL) ;
                        SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NULL) ;

                        iReturn = StretchBlt (hdc,   0, 0, cxBitmap, cyBitmap,
               hdcMem,     0, 0, cxBitmap, cyBitmap, SRCCOPY) ;
                        break ;
     }
     DeleteDC (hdcMem) ;
     RestoreDC (hdc, -1) ;
     return iReturn ;
}

/*--------------------------------------------------------------------------
-
     DibFlipHorizontal: Calls non-optimized DibSetPixel and DibGetPixel
--------------------------------------------------------------------------
*/

HDIB DibFlipHorizontal (HDIB hdibSrc)
{
     HDIB hdibDst ;
     int  cx, cy, x, y ;

     if (!DibIsAddressable (hdibSrc))
                        return NULL ;
     if (NULL == (hdibDst = DibCopy (hdibSrc, FALSE)))
                        return NULL ;
     cx = DibWidth    (hdibSrc) ;
     cy = DibHeight   (hdibSrc) ;

     for (x = 0 ; x < cx ; x++)
     for (y = 0 ; y < cy ; y++)
     {
          DibSetPixel (hdibDst, x, cy - 1 - y, DibGetPixel (hdibSrc, x, y)) ;
     }
```

```
     return hdibDst ;
}

/*-------------------------------------------------------------------------
-
     DibRotateRight: Calls optimized DibSetPixelx and DibGetPixelx
-------------------------------------------------------------------------
-*/

HDIB DibRotateRight (HDIB hdibSrc)
{
     HDIB  hdibDst ;
     int   cx, cy, x, y ;

     if (!DibIsAddressable (hdibSrc))
                    return NULL ;

     if (NULL == (hdibDst = DibCopy (hdibSrc, TRUE)))
                    return NULL ;

     cx = DibWidth (hdibSrc) ;
     cy = DibHeight (hdibSrc) ;

     switch (DibBitCount (hdibSrc))
    {
     case       1:
                         for ( x = 0 ; x < cx ; x++)
                         for ( y = 0 ; y < cy ; y++)
                                   DibSetPixel1 (hdibDst, cy - y - 1, x,
                                   DibGetPixel1 (hdibSrc, x, y)) ;
                         break ;

     case       4:
                         for ( x = 0 ; x < cx ; x++)
                         for ( y = 0 ; y < cy ; y++)
                                   DibSetPixel4 (hdibDst, cy - y - 1, x,
                                   DibGetPixel4 (hdibSrc, x, y)) ;
                         break ;

     case       8:
                         for ( x = 0 ; x < cx ; x++)
                         for ( y = 0 ; y < cy ; y++)
                               DibSetPixel8 (hdibDst, cy - y - 1, x,
                               DibGetPixel8 (hdibSrc, x, y)) ;
                         break ;
     case       16:
                         for (x = 0 ; x < cx ; x++)
```

```
                              for (y = 0 ; y < cy ; y++)
                                        DibSetPixel16 (hdibDst, cy - y - 1, x,
                                        DibGetPixel16 (hdibSrc, x, y)) ;
                              break ;

     case       24:
                              for (x = 0 ; x < cx ; x++)
                              for (y = 0 ; y < cy ; y++)
                                        DibSetPixel24 (hdibDst, cy - y - 1, x,
                                        DibGetPixel24 (hdibSrc, x, y)) ;
                              break ;

     case       32:
                              for (x = 0 ; x < cx ; x++)
                              for (y = 0 ; y < cy ; y++)
                                        DibSetPixel32 (hdibDst, cy - y - 1, x,
                                        DibGetPixel32 (hdibSrc, x, y)) ;
                              break ;
     }
     return hdibDst ;
}

/*------------------------------------------------------------------------
     PaletteMenu: Uncheck and check menu item on palette menu
------------------------------------------------------------------------*/

void PaletteMenu (HMENU hMenu, WORD wItemNew)
{
     static WORD wItem = IDM_PAL_NONE ;
     CheckMenuItem (hMenu, wItem, MF_UNCHECKED) ;
     wItem = wItemNew ;
     CheckMenuItem (hMenu, wItem, MF_CHECKED) ;
}

LRESULT CALLBACK WndProc (   HWND  hwnd,  UINT  message,  WPARAM  wParam,LPARAM
lParam)
{
     static     BOOL                        fHalftonePalette ;
     static          DOCINFO                di                              =
{sizeof(DOCINFO),TEXT("Dibble:Printing")} ;
     static     HBITMAP               hBitmap ;
     static     HDIB                        hdib ;
     static     HMENU                            hMenu ;
     static     HPALETTE              hPalette ;
     static     int                         cxClient,  cyClient,  iVscroll,
iHscroll ;
     static     OPENFILENAME     ofn ;
     static     PRINTDLG                    printdlg = { sizeof (PRINTDLG) } ;
```

```
     static     TCHAR                              szFileName    [MAX_PATH],
szTitleName [MAX_PATH] ;
     static     TCHAR                              szFilter[]= TEXT  ("Bitmap
Files (*.BMP)\0*.bmp\0")
            TEXT ("All Files (*.*)\0*.*\0\0") ;
     static TCHAR          *          szCompression[] = {
                  TEXT("BI_RGB"),TEXT("BI_RLE8"),TEXT("BI_RLE4"),
      TEXT("BI_BITFIELDS"),TEXT("Unknown")} ;
     static WORD           wShow = IDM_SHOW_NORMAL ;
     BOOL                  fSuccess ;
     BYTE                  *          pGlobal ;
     HDC                   hdc, hdcPrn ;
     HGLOBAL               hGlobal ;
     HDIB                  hdibNew ;
     int                   iEnable, cxPage, cyPage, iConvert ;
     PAINTSTRUCT           ps ;
     SCROLLINFO            si ;
     TCHAR                 szBuffer [256] ;

     switch (message)
     {
     case  WM_CREATE:

                              // Save the menu handle in a static variable

                    hMenu = GetMenu (hwnd) ;

                    // Initialize the OPENFILENAME structure for the File Open
                    //   and File Save dialog boxes.

                    ofn.lStructSize       = sizeof (OPENFILENAME) ;
                    ofn.hwndOwner         = hwnd ;
                    ofn.hInstance         = NULL ;
                    ofn.lpstrFilter       = szFilter ;
                    ofn.lpstrCustomFilter = NULL ;
                    ofn.nMaxCustFilter    = 0 ;
                    ofn.nFilterIndex      = 0 ;
                    ofn.lpstrFile         = szFileName ;
                    ofn.nMaxFile          = MAX_PATH ;
                    ofn.lpstrFileTitle    = szTitleName ;
                    ofn.nMaxFileTitle     = MAX_PATH ;
                    ofn.lpstrInitialDir   = NULL ;
                    ofn.lpstrTitle        = NULL ;
                    ofn.Flags             = OFN_OVERWRITEPROMPT ;
                    ofn.nFileOffset       = 0 ;
                    ofn.nFileExtension    = 0 ;
                    ofn.lpstrDefExt       = TEXT ("bmp") ;
                    ofn.lCustData         = 0 ;
```

```
                    ofn.lpfnHook          = NULL ;
                    ofn.lpTemplateName    = NULL ;
                    return 0 ;

     case  WM_DISPLAYCHANGE:
                    SendMessage (hwnd, WM_USER_DELETEPAL, 0, 0) ;
                    SendMessage (hwnd, WM_USER_CREATEPAL, TRUE, 0) ;
                    return 0 ;

     case  WM_SIZE:
                         // Save the client area width and height in static
variables.

                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;

                    wParam = FALSE ;
               // fall through
               // WM_USER_SETSCROLLS:  Programmer-defined Message!
               // Set the scroll bars. If the display mode is not normal,
               //   make them invisible. If wParam is TRUE, reset the
               //   scroll bar position.

     case  WM_USER_SETSCROLLS:
                    if (hdib == NULL || wShow != IDM_SHOW_NORMAL)
                    {
                                   si.cbSize        =            sizeof
(SCROLLINFO) ;
                                   si.fMask         = SIF_RANGE ;
                                   si.nMin          = 0 ;
                                   si.nMax          = 0 ;
                              SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
                              SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
                    }
                    else
                    {
                                   // First the vertical scroll

                              si.cbSize        = sizeof (SCROLLINFO) ;
                              si.fMask               = SIF_ALL ;
                              GetScrollInfo (hwnd, SB_VERT, &si) ;
                              si.nMin                = 0 ;
                              si.nMax                = DibHeight (hdib) ;
                              si.nPage               = cyClient ;
                         if ((BOOL) wParam)
                              si.nPos = 0 ;
                              SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
                                   GetScrollInfo (hwnd, SB_VERT, &si) ;
```

```
                                    iVscroll = si.nPos ;

                                          // Then the horizontal scroll

                                    GetScrollInfo (hwnd, SB_HORZ, &si) ;
                                    si.nMin          = 0 ;
                                    si.nMax          = DibWidth (hdib) ;
                                    si.nPage         = cxClient ;

                              if ((BOOL) wParam)
                                    si.nPos = 0 ;
                              SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
                              GetScrollInfo (hwnd, SB_HORZ, &si) ;
                              iHscroll = si.nPos ;
                        }
                        return 0 ;

                        // WM_VSCROLL: Vertically scroll the DIB

case  WM_VSCROLL:
                        si.cbSize = sizeof (SCROLLINFO) ;
                        si.fMask  = SIF_ALL ;
                        GetScrollInfo (hwnd, SB_VERT, &si) ;

                        iVscroll = si.nPos ;

                  switch (LOWORD (wParam))
                  {
                  case SB_LINEUP:   si.nPos - = 1 ;            break ;
                  case SB_LINEDOWN:  si.nPos + = 1 ;      break ;
                  case SB_PAGEUP:   si.nPos - = si.nPage ;break ;
                  case SB_PAGEDOWN:  si.nPos + = si.nPage ;break ;
                  case SB_THUMBTRACK:si.nPos        = si.nTrackPos ;break ;
                        default:
            break ;
                        }
                  si.fMask = SIF_POS ;
                  SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
                  GetScrollInfo (hwnd, SB_VERT, &si) ;
                  if (si.nPos != iVscroll)
                  {
                        ScrollWindow (hwnd, 0, iVscroll - si.nPos, NULL, NULL) ;
                              iVscroll = si.nPos ;
                              UpdateWindow (hwnd) ;
                  }
                        return 0 ;
```

```
                    // WM_HSCROLL: Horizontally scroll the DIB

     case  WM_HSCROLL:
                    si.cbSize = sizeof (SCROLLINFO) ;
                    si.fMask  = SIF_ALL ;
                    GetScrollInfo (hwnd, SB_HORZ, &si) ;

                    iHscroll = si.nPos ;

                    switch (LOWORD (wParam))
                    {
                    case SB_LINELEFT:  si.nPos -=1 ;   break ;
                    case SB_LINERIGHT: si.nPos +=1 ;   break ;
                    case SB_PAGELEFT:  si.nPos -=si.nPage ;break ;
                    case SB_PAGERIGHT: si.nPos +=si.nPage ;break ;
                    case SB_THUMBTRACK:si.nPos  =si.nTrackPos ;break ;
                    default: break ;
                    }

                    si.fMask = SIF_POS ;
                    SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
                    GetScrollInfo (hwnd, SB_HORZ, &si) ;

               if (si.nPos != iHscroll)
               {
                    ScrollWindow (hwnd, iHscroll - si.nPos, 0, NULL, NULL) ;
                    iHscroll = si.nPos ;
                    UpdateWindow (hwnd) ;
               }
               return 0 ;

               // WM_INITMENUPOPUP:  Enable or Gray menu items

     case  WM_INITMENUPOPUP:
                    if (hdib)
                                   iEnable = MF_ENABLED ;
                    else
                                   iEnable = MF_GRAYED ;
     EnableMenuItem (hMenu, IDM_FILE_SAVE,              iEnable) ;
     EnableMenuItem (hMenu, IDM_FILE_PRINT,             iEnable) ;
     EnableMenuItem (hMenu, IDM_FILE_PROPERTIES,        iEnable) ;
     EnableMenuItem (hMenu, IDM_EDIT_CUT,               iEnable) ;
     EnableMenuItem (hMenu, IDM_EDIT_COPY,              iEnable) ;
     EnableMenuItem (hMenu, IDM_EDIT_DELETE,            iEnable) ;

                    if (DibIsAddressable (hdib))
                                   iEnable = MF_ENABLED ;
                    else
```

```
                                    iEnable = MF_GRAYED ;

     EnableMenuItem (hMenu, IDM_EDIT_ROTATE,    iEnable) ;
     EnableMenuItem (hMenu, IDM_EDIT_FLIP,      iEnable) ;
     EnableMenuItem (hMenu, IDM_CONVERT_01,     iEnable) ;
     EnableMenuItem (hMenu, IDM_CONVERT_04,     iEnable) ;
     EnableMenuItem (hMenu, IDM_CONVERT_08,     iEnable) ;
     EnableMenuItem (hMenu, IDM_CONVERT_16,     iEnable) ;
     EnableMenuItem (hMenu, IDM_CONVERT_24,     iEnable) ;
     EnableMenuItem (hMenu, IDM_CONVERT_32,     iEnable) ;

     switch (DibBitCount (hdib))
     {
        case  1:    EnableMenuItem (hMenu, IDM_CONVERT_01, MF_GRAYED) ;
                    break ;
        case  4:    EnableMenuItem (hMenu, IDM_CONVERT_04, MF_GRAYED) ;
                     break ;
        case  8:    EnableMenuItem (hMenu, IDM_CONVERT_08, MF_GRAYED) ;
                    break ;
        case  16: EnableMenuItem (hMenu, IDM_CONVERT_16, MF_GRAYED) ;
                    break ;
        case  24: EnableMenuItem (hMenu, IDM_CONVERT_24, MF_GRAYED) ;
                   break ;
        case  32: EnableMenuItem (hMenu, IDM_CONVERT_32, MF_GRAYED) ;
                    break ;
     }

                    if (hdib && DibColorSize (hdib) > 0)
                                    iEnable = MF_ENABLED ;
                    else
                                    iEnable = MF_GRAYED ;

                    EnableMenuItem (hMenu, IDM_PAL_DIBTABLE,    iEnable) ;
               if (DibIsAddressable (hdib) && DibBitCount (hdib) > 8)
                                    iEnable = MF_ENABLED ;
                    else
                                    iEnable = MF_GRAYED ;

          EnableMenuItem         (hMenu, IDM_PAL_OPT_POP4,
iEnable) ;
          EnableMenuItem         (hMenu, IDM_PAL_OPT_POP5,
iEnable) ;
          EnableMenuItem         (hMenu, IDM_PAL_OPT_POP6,
iEnable) ;
          EnableMenuItem         (hMenu, IDM_PAL_OPT_MEDCUT,
iEnable) ;
          EnableMenuItem         (hMenu, IDM_EDIT_PASTE,
               IsClipboardFormatAvailable (CF_DIB) ? MF_ENABLED : MF_GRAYED) ;
```

```
                    return 0 ;

                    // WM_COMMAND:  Process all menu commands.

     case  WM_COMMAND:
                    iConvert = 0 ;

                    switch (LOWORD (wParam))
                    {
                    case  IDM_FILE_OPEN:

               // Show the File Open dialog box

                                        if (!GetOpenFileName (&ofn))
                                                    return 0 ;

               // If there's an existing DIB and palette, delete them

                                        SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0) ;

               // Load the DIB into memory

                                        SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                                        ShowCursor (TRUE) ;

                                        hdib = DibFileLoad (szFileName) ;

                                        ShowCursor (FALSE) ;
                                        SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                                                    // Reset the scroll bars

                                   SendMessage (hwnd, WM_USER_SETSCROLLS, TRUE, 0) ;

                                                    // Create the palette and DDB

                                   SendMessage (hwnd, WM_USER_CREATEPAL, TRUE, 0) ;

                                        if (!hdib)
                                        {
                         MessageBox (hwnd, TEXT ("Cannot load DIB file!"),
                                   szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                                        }
                                        InvalidateRect (hwnd, NULL, TRUE) ;
                                        return 0 ;

          case  IDM_FILE_SAVE:
```

```
                              // Show the File Save dialog box

                                        if    (!    GetSaveFileName (&ofn))
                                              return 0 ;

                              // Save the DIB to memory

                                    SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                                    ShowCursor (TRUE) ;

                                    fSuccess = DibFileSave (hdib, szFileName) ;

                                    ShowCursor (FALSE) ;
                                    SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                                    if (!fSuccess)
                  MessageBox ( hwnd, TEXT ("Cannot save DIB file!"),
                            szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                                    return 0 ;

                        case  IDM_FILE_PRINT:
                                    if (!hdib)
                                              return 0 ;

                                              // Get printer DC

                        printdlg.Flags   =   PD_RETURNDC   |   PD_NOPAGENUMS   |
PD_NOSELECTION ;

                                    if (!PrintDlg (&printdlg))
                                              return 0 ;

                                    if (NULL == (hdcPrn = printdlg.hDC))
                                    {
                   MessageBox( hwnd, TEXT ("Cannot obtain Printer DC"),
                             szAppName, MB_ICONEXCLAMATION | MB_OK) ;
                                              return 0 ;
                                    }
                        // Check if the printer can print bitmaps

           if (!(RC_BITBLT & GetDeviceCaps (hdcPrn, RASTERCAPS)))
                                    {
                                          DeleteDC (hdcPrn) ;
           MessageBox (   hwnd, TEXT ("Printer cannot print bitmaps"),
                        szAppName, MB_ICONEXCLAMATION | MB_OK) ;
                                          return 0 ;
                                    }
```

```
                                    // Get size of printable area of page

                              cxPage = GetDeviceCaps (hdcPrn, HORZRES) ;
                              cyPage = GetDeviceCaps (hdcPrn, VERTRES) ;

                                    fSuccess = FALSE ;

                                          // Send the DIB to the printer

                              SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                              ShowCursor (TRUE) ;

          if ((StartDoc (hdcPrn, &di) > 0) && (StartPage (hdcPrn) > 0))
          {
                DisplayDib (hdcPrn, DibBitmapHandle (hdib), 0, 0,
                            cxPage, cyPage, wShow, FALSE) ;

                                    if (EndPage (hdcPrn) > 0)
                                          {
                                                fSuccess = TRUE ;
                                                EndDoc (hdcPrn) ;
                                          }
                              }
                              ShowCursor (FALSE) ;
                              SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                              DeleteDC (hdcPrn) ;

                              if (  !fSuccess)
          MessageBox ( hwnd, TEXT ("Could not print bitmap"),
                      szAppName, MB_ICONEXCLAMATION | MB_OK) ;
                              return 0 ;

          case  IDM_FILE_PROPERTIES:
                              if (!hdib)
                                          return 0 ;

                    wsprintf (szBuffer, TEXT ("Pixel width:\t%i\n")
                           TEXT ("Pixel height:\t%i\n")
                              TEXT ("Bits per pixel:\t%i\n")
                              TEXT ("Number of colors:\t%i\n")
                              TEXT ("Compression:\t%s\n"),
             DibWidth (hdib), DibHeight (hdib),
             DibBitCount (hdib), DibNumColors (hdib),
             szCompression [min (3, DibCompression (hdib))]) ;

                     MessageBox (     hwnd, szBuffer, szAppName,
             MB_ICONEXCLAMATION | MB_OK) ;
```

```
                    return 0 ;

          case  IDM_APP_EXIT:
                    SendMessage (hwnd, WM_CLOSE, 0, 0) ;
                    return 0 ;

          case  IDM_EDIT_COPY:
          case  IDM_EDIT_CUT:
                if (!(hGlobal = DibCopyToPackedDib (hdib, TRUE)))
                              return 0 ;

                    OpenClipboard (hwnd) ;
                    EmptyClipboard () ;
                    SetClipboardData (CF_DIB, hGlobal) ;
                    CloseClipboard () ;

                    if (LOWORD (wParam) == IDM_EDIT_COPY)
               return 0 ;
               // fall through for IDM_EDIT_CUT
          case  IDM_EDIT_DELETE:
                    SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0) ;
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;

          case  IDM_EDIT_PASTE:
                    OpenClipboard (hwnd) ;

                    hGlobal = GetClipboardData (CF_DIB) ;
                    pGlobal = GlobalLock (hGlobal) ;

               // If there's an existing DIB and palette,delete them.
               // Then convert the packed DIB to an HDIB.

                    if (pGlobal)
                    {
               SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0) ;
               hdib = DibCopyFromPackedDib ((BITMAPINFO *) pGlobal) ;
               SendMessage (hwnd, WM_USER_CREATEPAL, TRUE, 0) ;
                    }

                    GlobalUnlock (hGlobal) ;
                    CloseClipboard () ;
                          // Reset the scroll bars

                SendMessage (hwnd, WM_USER_SETSCROLLS, TRUE, 0) ;
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;
```

```
                case  IDM_EDIT_ROTATE:
                                if (hdibNew = DibRotateRight (hdib))
                                {
                           DibDelete (hdib) ;
                           DeleteObject (hBitmap) ;
                           hdib = hdibNew ;
                           hBitmap = DibCopyToDdb (hdib, hwnd, hPalette) ;
                           SendMessage (hwnd, WM_USER_SETSCROLLS, TRUE, 0) ;
                           InvalidateRect (hwnd, NULL, TRUE) ;
                                }
                                else
                                {
               MessageBox (     hwnd, TEXT ("Not enough memory"),
                           szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                                }
                                return 0 ;

                case  IDM_EDIT_FLIP:
                                if (hdibNew = DibFlipHorizontal (hdib))
                                {
                           DibDelete (hdib) ;
                           DeleteObject (hBitmap) ;
                           hdib = hdibNew ;
                           hBitmap = DibCopyToDdb (hdib, hwnd, hPalette) ;
                           InvalidateRect (hwnd, NULL, TRUE) ;
                                }
                                else
                                {
               MessageBox (     hwnd, TEXT ("Not enough memory"),
                           szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                                }
                                return 0 ;

                case  IDM_SHOW_NORMAL:
                case  IDM_SHOW_CENTER:
                case  IDM_SHOW_STRETCH:
                case  IDM_SHOW_ISOSTRETCH:
                                CheckMenuItem (hMenu, wShow, MF_UNCHECKED) ;
                                wShow = LOWORD (wParam) ;
                                CheckMenuItem (hMenu, wShow, MF_CHECKED) ;
                           SendMessage (hwnd, WM_USER_SETSCROLLS, FALSE, 0) ;

                                InvalidateRect (hwnd, NULL, TRUE) ;
                                return 0 ;

                case  IDM_CONVERT_32:  iConvert += 8 ;
                case  IDM_CONVERT_24:  iConvert += 8 ;
                case  IDM_CONVERT_16:  iConvert += 8 ;
```

```
                case  IDM_CONVERT_08:  iConvert += 4 ;
                case  IDM_CONVERT_04:  iConvert += 3 ;
                case  IDM_CONVERT_01:  iConvert += 1 ;
                                       SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                                       ShowCursor (TRUE) ;

                                       hdibNew = DibConvert (hdib, iConvert) ;

                                       ShowCursor (FALSE) ;
                                       SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                                       if (hdibNew)
                                       {
                SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0) ;
                                       hdib = hdibNew ;
                SendMessage (hwnd, WM_USER_CREATEPAL, TRUE, 0) ;
                InvalidateRect (hwnd, NULL, TRUE) ;
                                       }
                                       else
                                       {
                MessageBox (     hwnd, TEXT ("Not enough memory"),
                           szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                                            }
                                            return 0 ;

                     case  IDM_APP_ABOUT:
           MessageBox (  hwnd, TEXT ("Dibble (c) Charles Petzold, 1998"),
                      szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                                       return 0 ;
                     }

                // All the other WM_COMMAND messages are from the palette
                //     items. Any existing palette is deleted, and the cursor
                //     is set to the hourglass.

                     SendMessage (hwnd, WM_USER_DELETEPAL, 0, 0) ;
                     SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                     ShowCursor (TRUE) ;
                // Notice that all messages for palette items are ended
                //     with break rather than return. This is to allow
                //     additional processing later on.

                     switch (LOWORD (wParam))
                     {
                     case  IDM_PAL_DIBTABLE:
                                       hPalette = DibPalDibTable (hdib) ;
                                       break ;
```

```
                    case  IDM_PAL_HALFTONE:
                                    hdc = GetDC (hwnd) ;

                                if (hPalette = CreateHalftonePalette (hdc))
                                                fHalftonePalette = TRUE ;

                                    ReleaseDC (hwnd, hdc) ;
                                    break ;

                    case  IDM_PAL_ALLPURPOSE:
                                    hPalette = DibPalAllPurpose () ;
                                    break ;

case IDM_PAL_GRAY2:hPalette     = DibPalUniformGrays   (2)      ;
break;
case IDM_PAL_GRAY3:hPalette     = DibPalUniformGrays   (3)      ;
break;
case IDM_PAL_GRAY4:hPalette     = DibPalUniformGrays   (4)      ;
break;
case IDM_PAL_GRAY8:hPalette     = DibPalUniformGrays   (8)      ;
break;
case  IDM_PAL_GRAY16:hPalette      = DibPalUniformGrays  (16)   ;
break;
case IDM_PAL_GRAY32:hPalette      = DibPalUniformGrays  (32)   ;
break;
case IDM_PAL_GRAY64:hPalette      = DibPalUniformGrays  (64)   ;
break;
case IDM_PAL_GRAY128:hPalette     = DibPalUniformGrays (128)   ;
break;
case IDM_PAL_GRAY256:hPalette     = DibPalUniformGrays (256)   ;
break;
case IDM_PAL_RGB222:hPalette      =    DibPalUniformColors    (2,2,2);
break;
case IDM_PAL_RGB333:hPalette      =    DibPalUniformColors    (3,3,3);
break;
case IDM_PAL_RGB444:hPalette      =    DibPalUniformColors    (4,4,4);
break;
case IDM_PAL_RGB555:hPalette      =    DibPalUniformColors    (5,5,5);
break;
case IDM_PAL_RGB666:hPalette      =    DibPalUniformColors    (6,6,6);
break;
case IDM_PAL_RGB775:hPalette      =    DibPalUniformColors    (7,7,5);
break;
case IDM_PAL_RGB757:hPalette      =    DibPalUniformColors    (7,5,7);
break;
case IDM_PAL_RGB577:hPalette      =    DibPalUniformColors    (5,7,7);
break;
case IDM_PAL_RGB884:hPalette      =    DibPalUniformColors    (8,8,4);
```

```
     break;
     case IDM_PAL_RGB848:hPalette      =    DibPalUniformColors    (8,4,8);
     break;
     case IDM_PAL_RGB488:hPalette      =    DibPalUniformColors    (4,8,8);
     break;
     case IDM_PAL_OPT_POP4:hPalette   = DibPalPopularity (hdib, 4) ;  break ;
     case IDM_PAL_OPT_POP5:hPalette   = DibPalPopularity (hdib, 5) ;  break ;
     case IDM_PAL_OPT_POP6:hPalette   = DibPalPopularity (hdib, 6) ;  break ;
     case IDM_PAL_OPT_MEDCUT:hPalette = DibPalMedianCut  (hdib, 6) ;  break ;
                     }

                // After processing Palette items from the menu, the cursor
                //    is restored to an arrow, the menu item is checked, and
                //    the window is invalidated.

                     hBitmap = DibCopyToDdb (hdib, hwnd, hPalette) ;

                     ShowCursor (FALSE) ;
                     SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                     if (hPalette)
                                PaletteMenu (hMenu, (LOWORD (wParam))) ;

                     InvalidateRect (hwnd, NULL, TRUE) ;
                     return 0 ;

                // This programmer-defined message deletes an existing DIB
                //   in preparation for getting a new one.  Invoked during
                //   File Open command, Edit Paste command, and others.

     case  WM_USER_DELETEDIB:
                     if (hdib)
                     {
                                     DibDelete (hdib) ;
                                     hdib = NULL ;
                     }
                     SendMessage (hwnd, WM_USER_DELETEPAL, 0, 0) ;
                     return 0 ;

                // This programmer-defined message deletes an existing palette
                //          in preparation for defining a new one.

     case  WM_USER_DELETEPAL:
                     if (hPalette)
                     {
                                     DeleteObject (hPalette) ;
                                     hPalette = NULL ;
                                     fHalftonePalette = FALSE ;
```

```
                                        PaletteMenu (hMenu, IDM_PAL_NONE) ;
                    }
                    if (hBitmap)
                                        DeleteObject (hBitmap) ;

                         return 0 ;

               // Programmer-defined message to create a new palette based on
               //          a new DIB.  If wParam == TRUE, create a DDB as well.

     case  WM_USER_CREATEPAL:
                    if (hdib)
                    {
                                        hdc = GetDC (hwnd) ;

                    if (!(RC_PALETTE & GetDeviceCaps (hdc, RASTERCAPS)))
                                        {
               PaletteMenu (hMenu, IDM_PAL_NONE) ;
                                        }
                                   else if (hPalette = DibPalDibTable (hdib))
                                        {
               PaletteMenu (hMenu, IDM_PAL_DIBTABLE) ;
                                        }
                                   else if (hPalette = CreateHalftonePalette (hdc))
                                        {
                                                  fHalftonePalette = TRUE ;
               PaletteMenu (hMenu, IDM_PAL_HALFTONE) ;
                                        }
                                        ReleaseDC (hwnd, hdc) ;

                                        if ((BOOL) wParam)
               hBitmap = DibCopyToDdb (hdib, hwnd, hPalette) ;
                    }
                    return 0 ;

     case  WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;

                    if (hPalette)
                    {
                                        SelectPalette (hdc, hPalette, FALSE) ;
                                        RealizePalette (hdc) ;
                    }
                    if (hBitmap)
               {
                                   DisplayDib (     hdc,
                    fHalftonePalette ? DibBitmapHandle (hdib) : hBitmap,
                    iHscroll, iVscroll,
```

```
                    cxClient, cyClient,
                    wShow, fHalftonePalette) ;
                    }
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

     case  WM_QUERYNEWPALETTE:
                    if (!hPalette)
                                    return FALSE ;

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    InvalidateRect (hwnd, NULL, TRUE) ;

                    ReleaseDC (hwnd, hdc) ;
                    return TRUE ;

     case  WM_PALETTECHANGED:
                    if (!hPalette || (HWND) wParam == hwnd)
                                    break ;

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    UpdateColors (hdc) ;

                    ReleaseDC (hwnd, hdc) ;
                    break ;

     case  WM_DESTROY:
                    if (hdib)
                                    DibDelete (hdib) ;

                    if (hBitmap)
                                    DeleteObject (hBitmap) ;

                    if (hPalette)
                                    DeleteObject (hPalette) ;

                    PostQuitMessage (0) ;
                    return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

DIBBLE.RC （摘录）

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
```

```
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////
/
// Menu
DIBBLE MENU DISCARDABLE BEGIN POPUP "&File"
      BEGIN
            MENUITEM "&Open...\tCtrl+O",       IDM_FILE_OPEN
            MENUITEM "&Save...\tCtrl+S",       IDM_FILE_SAVE
            MENUITEM SEPARATOR
            MENUITEM "&Print...\tCtrl+P",      IDM_FILE_PRINT
            MENUITEM SEPARATOR
            MENUITEM "Propert&ies...",         IDM_FILE_PROPERTIES
            MENUITEM SEPARATOR
            MENUITEM "E&xit",                        IDM_APP_EXIT
      END
      POPUP "&Edit"
      BEGIN
            MENUITEM "Cu&t\tCtrl+X",       IDM_EDIT_CUT
            MENUITEM "&Copy\tCtrl+C",      IDM_EDIT_COPY
      MENUITEM "&Paste\tCtrl+V",    IDM_EDIT_PASTE
            MENUITEM "&Delete\tDelete",   IDM_EDIT_DELETE
            MENUITEM SEPARATOR
       MENUITEM "&Flip",                  IDM_EDIT_FLIP
      MENUITEM "&Rotate",           IDM_EDIT_ROTATE
      END
      POPUP "&Show"
      BEGIN
       MENUITEM "&Actual Size",    IDM_SHOW_NORMAL, CHECKED
       MENUITEM "&Center",   IDM_SHOW_CENTER
       MENUITEM "&Stretch to Window",         IDM_SHOW_STRETCH
       MENUITEM "Stretch &Isotropically",     IDM_SHOW_ISOSTRETCH
      END
      POPUP "&Palette"
      BEGIN
       MENUITEM "&None",                      IDM_PAL_NONE, CHECKED
       MENUITEM "&Dib ColorTable",       IDM_PAL_DIBTABLE
       MENUITEM "&Halftone",       IDM_PAL_HALFTONE
       MENUITEM "&All-Purpose",    IDM_PAL_ALLPURPOSE
      POPUP "&Gray Shades"
            BEGIN
       MENUITEM "&1. 2 Grays",           IDM_PAL_GRAY2
       MENUITEM "&2. 3 Grays",      IDM_PAL_GRAY3
       MENUITEM "&3. 4 Grays",      IDM_PAL_GRAY4
       MENUITEM "&4. 8 Grays",      IDM_PAL_GRAY8
       MENUITEM "&5. 16 Grays",     IDM_PAL_GRAY16
       MENUITEM "&6. 32 Grays",     IDM_PAL_GRAY32
       MENUITEM "&7. 64 Grays",       IDM_PAL_GRAY64
       MENUITEM "&8. 128 Grays",      IDM_PAL_GRAY128
```

```
      MENUITEM "&9. 256 Grays",      IDM_PAL_GRAY256
           END
           POPUP "&Uniform Colors"
      BEGIN
       MENUITEM "&1. 2R x 2G x 2B (8)",            IDM_PAL_RGB222
       MENUITEM "&2. 3R x 3G x 3B (27)",           IDM_PAL_RGB333
       MENUITEM "&3. 4R x 4G x 4B (64)",         IDM_PAL_RGB444
       MENUITEM "&4. 5R x 5G x 5B (125)",     IDM_PAL_RGB555
       MENUITEM "&5. 6R x 6G x 6B (216)",     IDM_PAL_RGB666
       MENUITEM "&6. 7R x 7G x 5B (245)",     IDM_PAL_RGB775
          MENUITEM "&7. 7R x 5B x 7B (245)",        IDM_PAL_RGB757
       MENUITEM "&8. 5R x 7G x 7B (245)",     IDM_PAL_RGB577
       MENUITEM "&9. 8R x 8G x 4B (256)",     IDM_PAL_RGB884
       MENUITEM "&A. 8R x 4G x 8B (256)",     IDM_PAL_RGB848
      MENUITEM "&B. 4R x 8G x 8B (256)",       IDM_PAL_RGB488
           END
           POPUP "&Optimized"
           BEGIN
       MENUITEM "&1. Popularity Algorithm (4 bits)"IDM_PAL_OPT_POP4
       MENUITEM "&2. Popularity Algorithm (5 bits)"IDM_PAL_OPT_POP5
       MENUITEM "&3. Popularity Algorithm (6 bits)"IDM_PAL_OPT_POP6
       MENUITEM "&4. Median Cut Algorithm ", IDM_PAL_OPT_MEDCUT
           END
      END
      POPUP "Con&vert"
      BEGIN
       MENUITEM "&1. to 1 bit per pixel",     IDM_CONVERT_01
       MENUITEM "&2. to 4 bits per pixel",          IDM_CONVERT_04
       MENUITEM "&3. to 8 bits per pixel",          IDM_CONVERT_08
       MENUITEM "&4. to 16 bits per pixel",         IDM_CONVERT_16
       MENUITEM "&5. to 24 bits per pixel",         IDM_CONVERT_24
       MENUITEM "&6. to 32 bits per pixel",         IDM_CONVERT_32
      END
      POPUP "&Help"
      BEGIN
           MENUITEM "&About",
                IDM_APP_ABOUT
      END
END
//////////////////////////////////////////////////////////////////////////
/
// Accelerator
DIBBLE ACCELERATORS DISCARDABLE
BEGIN
      "C",         IDM_EDIT_COPY,         VIRTKEY, CONTROL, NOINVERT
      "O",      IDM_FILE_OPEN,         VIRTKEY, CONTROL, NOINVERT
      "P",      IDM_FILE_PRINT,        VIRTKEY, CONTROL, NOINVERT
      "S",      IDM_FILE_SAVE,         VIRTKEY, CONTROL, NOINVERT
```

```
      "V",      IDM_EDIT_PASTE,        VIRTKEY, CONTROL, NOINVERT
      VK_DELETE,        IDM_EDIT_DELETE,       VIRTKEY, NOINVERT
      "X",      IDM_EDIT_CUT,          VIRTKEY, CONTROL, NOINVERT
END
```

RESOURCE.H （摘录）

```
// Microsoft Developer Studio generated include file.
// Used by Dibble.rc

#define IDM_FILE_OPEN             40001
#define IDM_FILE_SAVE             40002
#define IDM_FILE_PRINT            40003
#define IDM_FILE_PROPERTIES        40004
#define IDM_APP_EXIT              40005
#define IDM_EDIT_CUT              40006
#define IDM_EDIT_COPY             40007
#define IDM_EDIT_PASTE            40008
#define IDM_EDIT_DELETE           40009
#define IDM_EDIT_FLIP             40010
#define IDM_EDIT_ROTATE           40011
#define IDM_SHOW_NORMAL           40012
#define IDM_SHOW_CENTER           40013
#define IDM_SHOW_STRETCH          40014
#define IDM_SHOW_ISOSTRETCH        40015
#define IDM_PAL_NONE              40016
#define IDM_PAL_DIBTABLE          40017
#define IDM_PAL_HALFTONE          40018
#define IDM_PAL_ALLPURPOSE         40019
#define IDM_PAL_GRAY2             40020
#define IDM_PAL_GRAY3             40021
#define IDM_PAL_GRAY4             40022
#define IDM_PAL_GRAY8             40023
#define IDM_PAL_GRAY16            40024
#define IDM_PAL_GRAY32            40025
#define IDM_PAL_GRAY64            40026
#define IDM_PAL_GRAY128           40027
#define IDM_PAL_GRAY256           40028
#define IDM_PAL_RGB222            40029
#define IDM_PAL_RGB333            40030
#define IDM_PAL_RGB444            40031
#define IDM_PAL_RGB555            40032
#define IDM_PAL_RGB666            40033
#define IDM_PAL_RGB775            40034
#define IDM_PAL_RGB757            40035
#define IDM_PAL_RGB577            40036
#define IDM_PAL_RGB884            40037
#define IDM_PAL_RGB848            40038
#define IDM_PAL_RGB488            40039
#define IDM_PAL_OPT_POP4          40040
```

```
#define IDM_PAL_OPT_POP5          40041
#define IDM_PAL_OPT_POP6          40042
#define IDM_PAL_OPT_MEDCUT        40043
#define IDM_CONVERT_01            40044
#define IDM_CONVERT_04            40045
#define IDM_CONVERT_08            40046
#define IDM_CONVERT_16            40047
#define IDM_CONVERT_24            40048
#define IDM_CONVERT_32            40049
#define IDM_APP_ABOUT             40050
```

DIBBLE 使用了两个其他档案，我将简要地说明它们。DIBCONV 档案（DIBCONV.C 和 DIBCONV.H）在两种不同格式之间转换——例如，从每图素 24 位元转换成每图素 8 位元。DIBPAL 档案（DIBPAL.C 和 DIBPAL.H）建立调色盘。

DIBBLE 维护 WndProc 中的三个重要的静态变数。这些是呼叫 hdib 的 HDIB 代号、呼叫 hPalette 的 HPALETTE 代号和呼叫 hBitmap 的 HBITMAP 代号。HDIB 来自 DIBHELP 中的不同函式；HPALETTE 来自 DIBPAL 中的不同函式或 CreateHalftonePalette 函式；而 HBITMAP 代号来自 DIBHELP.C 中的 DibCopyToDdb 函式并帮助加速萤幕显示，特别是在 256 色显示模式下。不过，无论在程式建立新的「DIB Section」（显而易见地）或在程式建立不同的调色盘（不很明显）时，这个代号都必须重新建立。

让我们从功能上而非循序渐进地来介绍一下 DIBBLE。

档案载入和储存

DIBBLE 可以在回应 IDM_FILE_LOAD 和 IDM_FILE_SAVE 的 WM_COMMAND 讯息处理过程中载入 DIB 档案并储存这些档案。在处理这些讯息处理期间，DIBBLE 通过分别呼叫 GetOpenFileName 和 GetSaveFileName 来启动公用档案对话方块。

对於「File」、「Save」功能表命令，DIBBLE 只需要呼叫 DibFileSave。对於「File」、「Open」功能表命令，DIBBLE 必须首先删除前面的 HDIB、调色盘和点阵图物件。它透过发送一个 WM_USER_DELETEDIB 讯息来完成这件事，此讯息通过呼叫 DibDelete 和 DeleteObject 来处理。然後 DIBBLE 呼叫 DIBHELP 中的 DibFileLoad 函式，发送 WM_USER_SETSCROLLS 和 WM_USER_CREATEPAL 讯息来重新设定卷动列并建立调色盘。WM_USER_CREATEPAL 讯息也位於程式从 DIB 区块建立的新的 DDB 位置。

显示、卷动和列印

DIBBLE 的功能表允许它以实际尺寸在显示区域左上角显示 DIB，或在显示区域中间显示 DIB，或伸展到填充显示区域，或者在保持纵横比的情况下尽量填

充显示区域。您可以在 DIBBLE 的「Show」功能表上来选择需要的选项。注意，这些与上一章的 SHOWDIB2 程式中四个选项相同。

在 WM_PAINT 讯息处理期间——也是处理「File」、「Print」命令的过程中——DIBBLE 呼叫 DisplayDib 函式。注意，DisplayDib 使用 BitBlt 和 StretchBlt，而不是使用 SetDIBitsToDevice 和 StretchDIBits。在 WM_PAINT 讯息处理期间，传递给函式的点阵图代号由 DibCopyToDdb 函式建立，并在 WM_USER_CREATEPAL 讯息处理期间呼叫。其中 DDB 与视讯装置内容相容。当处理「File」、「Print」命令时，DIBBLE 呼叫 DisplayDib，其中可用的 DIB 区块代号来自 DIBHELP.C 中的 DibBitmapHandle 函式。

另外要注意，DIBBLE 保留一个称作 fHalftonePalette 的静态 BOOL 变数，如果从 CreateHalftonePalette 函式中获得 hPalette，则此变数设定为 TRUE。这将迫使 DisplayDib 函式呼叫 StretchBlt 而不是呼叫 BitBlt，即使 DIB 被指定按实际尺寸显示。fHalftonePalette 变数也导致 WM_PAINT 处理程式将 DIB 区块代号传递给 DisplayDib 函式，而不是由 DibCopyToDdb 函式建立的点阵图代号。本章前面讨论过中间色调色盘的使用，并在 SHOWDIB5 程式中进行了展示。

第一次使用范例程式时，DIBBLE 允许在显示区域中卷动 DIB。只有按实际尺寸显示 DIB 时，才显示卷动列。在处理 WM_PAINT 时，WndProc 简单地将卷动列的目前位置传递给 DisplayDib 函式。

剪贴簿

对於「Cut」和「Copy」功能表项，DIBBLE 呼叫 DIBHELP 中的 DibCopyToPackedDib 函式。该函式将获得所有的 DIB 元件并将它们放入大的记忆体块中。

对於第一次使用本书中的某些范例程式来说，DIBBLE 从剪贴簿中粘贴 DIB。这包括呼叫 DibCopyFromPackedDib 函式，并替换视窗讯息处理程式前面储存的 HDIB、调色盘和点阵图。

翻转和旋转

DIBBLE 中的「Edit」功能表中除了常见的「Cut」、「Copy」、「Paste」和「Delete」选项之外，还包括两个附加项——「Flip」和「Rotate」。「Flip」选项使点阵图绕水平轴翻转——即上下颠倒翻转。「Rotate」选项使点阵图顺时针旋转 90 度。这两个函式都需要透过将它们从一个 DIB 复制到另一个来存取所有的 DIB 图素（因为这两个函式不需要建立新的调色盘，所以不删除和重新建立调色盘）。

「Flip」功能表选项使用 DibFlipHorizontal 函式，此函式也位於 DIBBLE.C 档案。此函式呼叫 DibCopy 来获得 DIB 精确的副本。然後，进入将原 DIB 中的图素复制到新 DIB 的回圈，但是复制这些图素是为了上下翻转图像。注意，此函式呼叫 DibGetPixel 和 DibSetPixel。这些是 DIBHELP.C 中的通用（但不像我们所希望的那么快）函式。

为了说明 DibGetPixel 和 DibSetPixel 函式与 DIBHELP.H 中执行更快的 DibGetPixel 和 DibSetPixel 巨集之间的区别，DibRotateRight 函式使用了巨集。然而，首先要注意的是，该函式呼叫 DibCopy 时，第二个参数设定为 TRUE。这导致 DibCopy 翻转原 DIB 的宽度和高度来建立新的 DIB。另外，图素位元不能由 DibCopy 函式复制。但是，DibRotateRight 函式有六个不同的回圈将图素位元从原 DIB 复制到新的 DIB——每一个都对应不同的 DIB 图素宽度（1 位元、4 位元、8 位元、16 位元、24 位元和 32 位元）。虽然包括了更多的程式码，但是函式更快了。

尽管可以使用「Flip Horizontal」和「Rotate Right」选项来产生「Flip Vertical」、「Rotate Left」和「Rotate 180 啊构δ埽　　ǔ 3淌浇　苯又葱兴　醒∠睢 1 暇梗琐 IBBLE 只是个展示程式而已。

简单调色盘；最佳化调色盘

在 DIBBLE 中，您可以在 256 色视讯显示器上选择不同的调色盘来显示 DIB。这些都在 DIBBLE 的「Palette」功能表中列出。除了中间色调色盘以外，其余的都直接由 Windows 函式呼叫建立，建立不同调色盘的所有函式都由程式 16-24 所示的 DIBPAL 档案提供。

程式 16-24　DIBPAL 档案

```
DIBPAL.H
/*------------------------------------------------------------------------
      DIBPAL.H header file for DIBPAL.C
------------------------------------------------------------------------
*/

HPALETTE DibPalDibTable (HDIB hdib) ;
HPALETTE DibPalAllPurpose (void) ;
HPALETTE DibPalUniformGrays (int iNum) ;
HPALETTE DibPalUniformColors (int iNumR, int iNumG, int iNumB) ;
HPALETTE DibPalVga (void) ;
HPALETTE DibPalPopularity (HDIB hdib, int iRes) ;
HPALETTE DibPalMedianCut (HDIB hdib, int iRes) ;
DIBPAL.C
/*------------------------------------------------------------------------
--
```

```
      DIBPAL.C --      Palette-Creation Functions
                                          (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>
#include "dibhelp.h"
#include "dibpal.h"

/*------------------------------------------------------------------------
-
      DibPalDibTable: Creates a palette from the DIB color table
------------------------------------------------------------------------
-*/

HPALETTE DibPalDibTable (HDIB hdib)
{
      HPALETTE                    hPalette ;
      int                         i, iNum ;
      LOGPALETTE *     plp ;
      RGBQUAD                     rgb ;

      if (0 == (iNum = DibNumColors (hdib)))
                        return NULL ;
      plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (PALETTEENTRY)) ;
      plp->palVersion             = 0x0300 ;
      plp->palNumEntries     = iNum ;

      for (i = 0 ; i < iNum ; i++)
      {
                        DibGetColor (hdib, i, &rgb) ;
                        plp->palPalEntry[i].peRed   = rgb.rgbRed ;
                        plp->palPalEntry[i].peGreen = rgb.rgbGreen ;
                        plp->palPalEntry[i].peBlue  = rgb.rgbBlue ;
                        plp->palPalEntry[i].peFlags = 0 ;
      }
      hPalette = CreatePalette (plp) ;
      free (plp) ;
      return hPalette ;
}
/*------------------------------------------------------------------------
-
      DibPalA    llPurpose: Creates a palette suitable for a wide variety
                            of images; the palette has 247 entries, but 15 of
them are
                            duplicates or match the standard 20 colors.
------------------------------------------------------------------------
*/
```

```
HPALETTE DibPalAllPurpose (void)
{
      HPALETTE                    hPalette ;
      int                         i, incr, R, G, B ;
      LOGPALETTE *      plp ;

      plp = malloc (sizeof (LOGPALETTE) + 246 * sizeof (PALETTEENTRY)) ;
      plp->palVersion             = 0x0300 ;
      plp->palNumEntries     = 247 ;

                        // The following loop calculates 31 gray shades, but 3
of them
                        //               will match the standard 20 colors

      for (i = 0, G = 0, incr = 8 ; G <= 0xFF ; i++, G += incr)
     {
                        plp->palPalEntry[i].peRed         = (BYTE) G ;
                        plp->palPalEntry[i].peGreen  = (BYTE) G ;
                        plp->palPalEntry[i].peBlue   = (BYTE) G ;
                        plp->palPalEntry[i].peFlags  = 0 ;

                        incr = (incr == 9 ? 8 : 9) ;
      }

            // The following loop is responsible for 216 entries, but 8 of
                  //                    them will match the standard 20 colors,
and another
                  //                    4 of them will match the gray shades
above.

      for (R = 0 ; R <= 0xFF ; R += 0x33)
      for (G = 0 ; G <= 0xFF ; G += 0x33)
      for (B = 0 ; B <= 0xFF ; B += 0x33)
      {
                        plp->palPalEntry[i].peRed               = (BYTE) R ;
                        plp->palPalEntry[i].peGreen       = (BYTE) G ;
                        plp->palPalEntry[i].peBlue        = (BYTE) B ;
                        plp->palPalEntry[i].peFlags       = 0 ;

                        i++ ;
      }
      hPalette = CreatePalette (plp) ;
      free (plp) ;
      return hPalette ;
}

/*------------------------------------------------------------------------
```

```
-
      DibPalUniformGrays:  Creates a palette of iNum grays, uniformly spaced
----------------------------------------------------------------------------
*/

HPALETTE DibPalUniformGrays (int iNum)
{
      HPALETTE                    hPalette ;
      int                         i ;
      LOGPALETTE *     plp ;

      plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (PALETTEENTRY)) ;
      plp->palVersion                  = 0x0300 ;
      plp->palNumEntries          = iNum ;

      for (i = 0 ; i < iNum ; i++)
      {
                        plp->palPalEntry[i].peRed         =
                        plp->palPalEntry[i].peGreen  =
                        plp->palPalEntry[i].peBlue   = (BYTE) (i * 255 / (iNum -
1)) ;
                        plp->palPalEntry[i].peFlags = 0 ;
      }
      hPalette = CreatePalette (plp) ;
      free (plp) ;
      return hPalette ;
}

/*----------------------------------------------------------------------------
      DibPalUniformColors: Creates a palette of iNumR x iNumG x iNumB colors
----------------------------------------------------------------------------
*/

HPALETTE DibPalUniformColors (int iNumR, int iNumG, int iNumB)
{
      HPALETTE                    hPalette ;
      int                         i, iNum, R, G, B ;
      LOGPALETTE *     plp ;

      iNum = iNumR * iNumG * iNumB ;
      plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (PALETTEENTRY)) ;
      plp->palVersion    = 0x0300 ;
      plp->palNumEntries = iNumR * iNumG * iNumB ;

      i = 0 ;
      for (R = 0 ; R < iNumR ; R++)
      for (G = 0 ; G < iNumG ; G++)
      for (B = 0 ; B < iNumB ; B++)
```

```
     {
                plp->palPalEntry[i].peRed    = (BYTE) (R * 255 / (iNumR - 1)) ;
                plp->palPalEntry[i].peGreen  = (BYTE) (G * 255 / (iNumG - 1)) ;
                plp->palPalEntry[i].peBlue   = (BYTE) (B * 255 / (iNumB - 1)) ;
                plp->palPalEntry[i].peFlags  = 0 ;

                     i++ ;
     }
     hPalette = CreatePalette (plp) ;
     free (plp) ;
     return hPalette ;
}

/*----------------------------------------------------------------------------
-
     DibPalVga:  Creates a palette based on standard 16 VGA colors
----------------------------------------------------------------------------
*/

HPALETTE DibPalVga (void)
{
     static RGBQUAD rgb [16] = { 0x00, 0x00, 0x00, 0x00,
                           0x00, 0x00, 0x80,  0x00,
                           0x00, 0x80, 0x00,  0x00,
                           0x00, 0x80, 0x80,  0x00,
                           0x80, 0x00, 0x00,  0x00,
                           0x80, 0x00, 0x80,  0x00,
                           0x80, 0x80, 0x00,  0x00,
                           0x80, 0x80, 0x80,  0x00,
                           0xC0, 0xC0, 0xC0,  0x00,
                           0x00, 0x00, 0xFF,  0x00,
                           0x00, 0xFF, 0x00,  0x00,
                           0x00, 0xFF, 0xFF,  0x00,
                           0xFF, 0x00, 0x00,  0x00,
                           0xFF, 0x00, 0xFF,  0x00,
                           0xFF, 0xFF, 0x00,  0x00,
                           0xFF, 0xFF, 0xFF,  0x00 } ;
     HPALETTE                    hPalette ;
     int                         i ;
     LOGPALETTE                * plp ;

     plp = malloc (sizeof (LOGPALETTE) + 15 * sizeof (PALETTEENTRY)) ;
     plp->palVersion             = 0x0300 ;
     plp->palNumEntries     = 16 ;

     for (i = 0 ; i < 16 ; i++)
     {
                    plp->palPalEntry[i].peRed    = rgb[i].rgbRed ;
```

```
                        plp->palPalEntry[i].peGreen  = rgb[i].rgbGreen ;
                        plp->palPalEntry[i].peBlue   = rgb[i].rgbBlue ;
                        plp->palPalEntry[i].peFlags  = 0 ;
      }
      hPalette = CreatePalette (plp) ;
      free (plp) ;
      return hPalette ;
}

/*-------------------------------------------------------------------------
-
      Macro used in palette optimization routines

-------------------------------------------------------------------------*/

#define PACK_RGB(R,G,B,iRes) ((int) (R) |     ((int) (G) <<  (iRes)) |     \
                         ((int) (B) << ((iRes) + (iRes))))

/*-------------------------------------------------------------------------
--
      AccumColorCounts: Fills up piCount (indexed by a packed RGB color)
      with counts of pixels of that color.
-------------------------------------------------------------------------
-*/

static void AccumColorCounts (HDIB hdib, int * piCount, int iRes)
{
      int         x, y, cx, cy ;
      RGBQUAD     rgb ;

      cx = DibWidth (hdib) ;
      cy = DibHeight (hdib) ;

      for (y = 0 ; y < cy ; y++)
      for (x = 0 ; x < cx ; x++)
      {
                        DibGetPixelColor (hdib, x, y, &rgb) ;

                        rgb.rgbRed       >>= (8 - iRes) ;
                        rgb.rgbGreen     >>= (8 - iRes) ;
                        rgb.rgbBlue      >>= (8 - iRes) ;

                        ++piCount    [PACK_RGB    (rgb.rgbRed,    rgb.rgbGreen,
rgb.rgbBlue, iRes)] ;
      }
}

/*-------------------------------------------------------------------------
```

```
-
      DibPalPopularity:  Popularity algorithm for optimized colors
------------------------------------------------------------------------------
-*/

HPALETTE DibPalPopularity (HDIB hdib, int iRes)
{
      HPALETTE                    hPalette ;
      int                         i, iArraySize, iEntry, iCount, iIndex, iMask,
R, G, B ;
      int                   *     piCount ;
      LOGPALETTE  *     plp ;

                        // Validity checks

      if (DibBitCount (hdib) < 16)
                        return NULL ;
      if (iRes < 3 || iRes > 8)
                  return NULL ;
                  // Allocate array for counting pixel colors
      iArraySize = 1 << (3 * iRes) ;
      iMask = (1 << iRes) - 1 ;

      if (NULL == (piCount = calloc (iArraySize, sizeof (int))))
                        return NULL ;
                        // Get the color counts
      AccumColorCounts (hdib, piCount, iRes) ;
                        // Set up a palette
      plp = malloc (sizeof (LOGPALETTE) + 235 * sizeof (PALETTEENTRY)) ;
      plp->palVersion = 0x0300 ;
      for (iEntry = 0 ; iEntry < 236 ; iEntry++)
      {
                        for (i = 0, iCount = 0 ; i < iArraySize ; i++)
                                          if (piCount[i] > iCount)

                                          {
                                                      iCount = piCount[i] ;
                                                      iIndex = i ;
                                          }
                        if (iCount == 0)
                                          break ;
                        R = (iMask &  iIndex) << (8 - iRes) ;
              G = (iMask & (iIndex >> iRes )) << (8 - iRes) ;
              B = (iMask & (iIndex >> (iRes + iRes)))<< (8 - iRes) ;

                        plp->palPalEntry[iEntry].peRed    = (BYTE) R ;
                        plp->palPalEntry[iEntry].peGreen  = (BYTE) G ;
                        plp->palPalEntry[iEntry].peBlue   = (BYTE) B ;
```

```
                        plp->palPalEntry[iEntry].peFlags  = 0 ;

                        piCount [iIndex] = 0 ;
     }
                  // On exit from the loop iEntry will be the number of stored
entries
      plp->palNumEntries = iEntry ;
                  // Create the palette, clean up, and return the palette handle
      hPalette = CreatePalette (plp) ;
      free (piCount) ;
      free (plp) ;

      return hPalette ;
}

/*------------------------------------------------------------------------
      Structures used for implementing median cut algorithm
------------------------------------------------------------------------
*/

typedef struct                                  // defines dimension of a box
{
      int Rmin, Rmax, Gmin, Gmax, Bmin, Bmax ;
}
MINMAX ;
typedef struct                                  // for Compare routine for qsort
{
      int               iBoxCount ;
      RGBQUAD     rgbBoxAv ;
}

BOXES ;
/*------------------------------------------------------------------------
--
      FindAverageColor: In a box
------------------------------------------------------------------------
-*/

static int FindAverageColor (     int * piCount, MINMAX mm,
                              int iRes, RGBQUAD * prgb)
{
      int R, G, B, iR, iG, iB, iTotal, iCount ;
                        // Initialize some variables
      iTotal = iR = iG = iB = 0 ;
                        // Loop through all colors in the box
      for (R = mm.Rmin ; R <= mm.Rmax ; R++)
      for (G = mm.Gmin ; G <= mm.Gmax ; G++)
      for (B = mm.Bmin ; B <= mm.Bmax ; B++)
```

```
     {
                                   // Get the number of pixels of that color
                         iCount = piCount [PACK_RGB (R, G, B, iRes)] ;
                                   // Weight the pixel count by the color value
                         iR += iCount * R ;
                         iG += iCount * G ;
                         iB += iCount * B ;

                         iTotal += iCount ;
     }
                         // Find the average color
     prgb->rgbRed           = (BYTE) ((iR / iTotal) << (8 - iRes)) ;
     prgb->rgbGreen         = (BYTE) ((iG / iTotal) << (8 - iRes)) ;
     prgb->rgbBlue          = (BYTE) ((iB / iTotal) << (8 - iRes)) ;

                         // Return the total number of pixels in the box

     return iTotal ;
}

/*-------------------------------------------------------------------------
-
   CutBox:  Divide a box in two
-------------------------------------------------------------------------
*/
static void CutBox (int * piCount, int iBoxCount, MINMAX mm,
                 int iRes, int iLevel, BOXES * pboxes, int * piEntry)
{
     int               iCount, R, G, B ;
     MINMAX      mmNew ;

                         // If the box is empty, return

     if (iBoxCount == 0)
                         return ;

                         // If the nesting level is 8, or the box is one pixel,
we're ready
                         // to find the average color in the box and save it along
with
                         // the number of pixels of that color

     if (iLevel == 8 || (   mm.Rmin == mm.Rmax &&
                            mm.Gmin == mm.Gmax &&
                            mm.Bmin == mm.Bmax))
    {
                pboxes[*piEntry].iBoxCount =
                         FindAverageColor       (piCount,       mm,       iRes,
```

```
&pboxes[*piEntry].rgbBoxAv) ;
                        (*piEntry) ++ ;
     }
                        // Otherwise, if blue is the largest side, split it
     else if ((mm.Bmax - mm.Bmin > mm.Rmax - mm.Rmin) &&
                                          (mm.Bmax - mm.Bmin > mm.Gmax - mm.Gmin))
     {
                        // Initialize a counter and loop through the blue side
                iCount = 0 ;
                for (B = mm.Bmin ; B < mm.Bmax ; B++)
                {
                        // Accumulate all the pixels for each successive blue value
                              for ( R = mm.Rmin ; R <= mm.Rmax ; R++)
                              for ( G = mm.Gmin ; G <= mm.Gmax ; G++)
                              iCount += piCount [PACK_RGB (R, G, B, iRes)] ;

                              // If it's more than half the box count, we're there

                              if (i Count >= iBoxCount / 2)
                                          break ;

                        //    If the next blue value will be the max, we're there
                                                      if (  B == mm.Bmax - 1)
                                                            break ;
                }
                                    // Cut the two split boxes.
                        //   The second argument to CutBox is the new box count.
                        //   The third argument is the new min and max values.

                mmNew = mm ;
                mmNew.Bmin = mm.Bmin ;
                mmNew.Bmax = B ;

                CutBox (    piCount, iCount, mmNew, iRes, iLevel + 1,
                                                pboxes, piEntry) ;

                mmNew.Bmin = B + 1 ;
                mmNew.Bmax = mm.Bmax ;

          CutBox (    piCount, iBoxCount - iCount, mmNew, iRes, iLevel + 1,
                                                pboxes, piEntry) ;
     }
          // Otherwise, if red is the largest side, split it (just like blue)
     else if (mm.Rmax - mm.Rmin > mm.Gmax - mm.Gmin)
    {
                iCount = 0 ;
                for (R = mm.Rmin ; R < mm.Rmax ; R++)
                {
```

```
                                        for (B = mm.Bmin ; B <= mm.Bmax ; B++)
                                        for (G = mm.Gmin ; G <= mm.Gmax ; G++)
                                  iCount += piCount [PACK_RGB (R, G, B, iRes)] ;
                                        if (iCount >= iBoxCount / 2)
                                                break ;
                                        if (R == mm.Rmax - 1)
                                                break ;
          }
                mmNew = mm ;
                mmNew.Rmin = mm.Rmin ;
                mmNew.Rmax = R ;

                CutBox (    piCount, iCount, mmNew, iRes, iLevel + 1,
                                        pboxes, piEntry) ;

                mmNew.Rmin  = R + 1 ;
                mmNew.Rmax  = mm.Rmax ;
          CutBox (    piCount, iBoxCount - iCount, mmNew, iRes, iLevel + 1,
                                        pboxes, piEntry) ;
    }
                          // Otherwise, split along the green size
      else
      {
                iCount = 0 ;
                for (G = mm.Gmin ; G < mm.Gmax ; G++)
                {
                          for ( B = mm.Bmin ; B <= mm.Bmax ; B++)
                          for ( R = mm.Rmin ; R <= mm.Rmax ; R++)
                          iCount += piCount [PACK_RGB (R, G, B, iRes)] ;

                          if (  iCount >= iBoxCount / 2)
                                        break ;

                          if (  G == mm.Gmax - 1)
                                        break ;
                }
                mmNew = mm ;
                mmNew.Gmin = mm.Gmin ;
                mmNew.Gmax = G ;

                CutBox (    piCount, iCount, mmNew, iRes, iLevel + 1,
                                        pboxes, piEntry) ;

                mmNew.Gmin  = G + 1 ;
                mmNew.Gmax  = mm.Gmax ;

          CutBox (    piCount, iBoxCount - iCount, mmNew, iRes, iLevel + 1,
                                        pboxes, piEntry) ;
```

```
     }
}

/*-------------------------------------------------------------------------
-
     Compare routine for qsort
-------------------------------------------------------------------------
-*/

static int Compare (const BOXES * pbox1, const BOXES * pbox2)
{
     return pbox1->iBoxCount - pbox2->iBoxCount ;
}

/*-------------------------------------------------------------------------
-
     DibPalMedianCut:  Creates palette based on median cut algorithm

-------------------------------------------------------------------------*/
HPALETTE DibPalMedianCut (HDIB hdib, int iRes)
{
     BOXES            boxes [256] ;
     HPALETTE         hPalette ;
     int         i, iArraySize, iCount, R, G, B, iTotCount, iDim, iEntry = 0 ;
     int              * piCount ;
     LOGPALETTE *     plp ;
     MINMAX           mm ;

                      // Validity checks

     if (DibBitCount (hdib) < 16)
                      return NULL ;
     if (iRes < 3 || iRes > 8)
                      return NULL ;
                      // Accumulate counts of pixel colors
     iArraySize = 1 << (3 * iRes) ;
     if (NULL == (piCount = calloc (iArraySize, sizeof (int))))
                      return NULL ;
     AccumColorCounts (hdib, piCount, iRes) ;
                // Find the dimensions of the total box
     iDim = 1 << iRes ;
     mm.Rmin = mm.Gmin = mm.Bmin = iDim - 1 ;
     mm.Rmax = mm.Gmax = mm.Bmax = 0 ;

     iTotCount = 0 ;
     for (R = 0 ; R < iDim ; R++)
     for (G = 0 ; G < iDim ; G++)
     for (B = 0 ; B < iDim ; B++)
```

```
                    if ((iCount = piCount [PACK_RGB (R, G, B, iRes)]) > 0)
                    {
                                iTotCount += iCount ;
                                if (R < mm.Rmin) mm.Rmin = R ;
                                if (G < mm.Gmin) mm.Gmin = G ;
                                if (B < mm.Bmin) mm.Bmin = B ;
                                if (R > mm.Rmax) mm.Rmax = R ;
                                if (G > mm.Gmax) mm.Gmax = G ;
                                if (B > mm.Bmax) mm.Bmax = B ;
                    }

                // Cut the first box (iterative function).
              // On return, the boxes structure will have up to 256 RGB values,
                //    one for each of the boxes, and the number of pixels in
                //  each box.
                //  The iEntry value will indicate the number of non-empty boxes.

     CutBox (piCount, iTotCount, mm, iRes, 0, boxes, &iEntry) ;
     free (piCount) ;

                // Sort the RGB table by the number of pixels for each color
     qsort (boxes, iEntry, sizeof (BOXES), Compare) ;
     plp = malloc (sizeof (LOGPALETTE) + (iEntry - 1) * sizeof (PALETTEENTRY)) ;
     if (plp == NULL)
                    return NULL ;
     plp->palVersion              = 0x0300 ;
     plp->palNumEntries     = iEntry ;

     for (i = 0 ; i < iEntry ; i++)
     {
                plp->palPalEntry[i].peRed  = boxes[i].rgbBoxAv.rgbRed ;
                plp->palPalEntry[i].peGreen= boxes[i].rgbBoxAv.rgbGreen ;
                plp->palPalEntry[i].peBlue = boxes[i].rgbBoxAv.rgbBlue ;
                plp->palPalEntry[i].peFlags= 0 ;
    }

     hPalette = CreatePalette (plp) ;
     free (plp) ;
     return hPalette ;
}
```

第一个函式——DibPalDibTable——看起来应该很熟悉。它根据 DIB 的颜色表建立了调色盘。这与本章前面的 SHOWDIB3 中所用到的 PACKEDIB.C 里的 PackedDibCreatePalette 函式相似。在 SHOWDIB3 中，只有当 DIB 有颜色表时才执行此函式。在 8 位元显示模式下试图显示 16 位元、24 位元或 32 位元 DIB 时，此函式就没用了。

预设情况下，执行在 256 色显示模式下时，DIBBLE 将首先尝试呼叫

DibPalDibTable 来根据 DIB 颜色表建立调色盘。如果 DIB 没有颜色表，则 DIBBLE 将呼叫 CreateHalftonePalette 并将 fHalftonePalette 变数设定为 TRUE。此逻辑发生在 WM_USER_CREATEPAL 讯息处理期间。

DIBPAL.C 也执行函式 DibPalAllPurpose，因为此函式与 SHOWDIB4 中的 CreateAllPurposePalette 函式非常相似，所以它看起来也很熟悉。您也可以从 DIBBLE 功能表中选择此调色盘。

在 256 色模式下显示点阵图最有趣的是，您可以直接控制 Windows 用於显示图像的颜色。如果您选择并显现调色盘，则 Winsows 将使用此调色盘中的颜色，而不是其他调色盘中的颜色。

例如，您可以用 DibPalUniformGrays 函式来单独建立一种灰阶调色盘。使用两种灰阶的调色盘则只含有 00-00-00（黑色）和 FF-FF-FF（白色）。用此调色盘来输出图像将提供某些照片中常用的高对比「黑白」效果。使用 3 种灰阶将在黑色和白色中间添加中间灰色，使用 4 种灰阶将添加 2 种灰阶。

用 8 种灰阶，您就有可能看到明显的轮廓——相同灰阶的无规则斑点，虽然很明显地执行了最接近颜色演算法，但是一般仍不带有任何审美判断。通常到 16 种灰阶就可以明显改善图像画质。使用 32 种灰阶差不多就可以消除全部轮廓了。而目前普遍认为 64 种灰阶是现在大多数显示设备的极限。在这点以上，再提升也没什么边际效益了。在 6 位元颜色解析度的设备上提供超过 64 种灰阶看不出有什么改进之处。

迄今为止，对於 8 位元显示模式下显示 16 位元、24 位元和 32 位元彩色 DIB，我们最多就是能够设计通用调色盘（这对灰阶图像很有效，但通常不适於彩色图像）或者使用中间色调色盘，它用混色显示与通用颜色调色盘合用。

还应注意，当您在 8 位元颜色模式下为大张 16 位元、24 位元或 32 位元 DIB 选择通用调色盘时，为了要显示这些图像，程式将花费一些时间依据 DIB 的内容来建立 GDI 点阵图物件。如果不需要调色盘，则程式根据 DIB 来建立 DDB 的时间会更少（用 8 位元彩色模式显示大 24 位元 DIB 时，比较 SHOWDIB1 和 SHOWDIB4 的性能，您也能看出这点区别）。这是为什么呢？

它按最接近颜色搜寻。通常，用 8 位元显示模式显示 24 位元 DIB 时（或者将 DIB 转换为 DDB），GDI 必须将 DIB 中的每个图素都与静态 20 种颜色中的一种相贴近。完成此操作的唯一方法是决定哪种静态颜色与图素颜色最接近。这包括计算图素与三维 RGB 颜色中每种静态颜色的距离。这将花些时间，特别是在 DIB 图像中有上百万个图素时。

在建立 232 色调色盘时，例如 DIBBLE 和 SHOWDIB4 中的通用调色盘，您会很快将搜索最接近颜色的时间增加到超过 11 倍！GDI 现在必须彻底检查 232 种

颜色，而不是 20 种。那就是显示 DIB 的整个作业放慢的原因。

这里的教训是避免在 8 位元显示模式下显示 24 位元（或 16 位元，或 32 位元）DIB。您应该找出最接近 DIB 图像颜色范围的 256 色调色盘，来将它们转换成 8 位元 DIB。这经常称为「最佳调色盘」。当我研究这个问题的时候，Paul Heckbert 编写的〈Color Image Quantization for Frame Buffer Displays〉（刊登在 1982 年 7 月出版的《Computer Graphics》）对此问题有所帮助。

均匀分布

建立 256 色调色盘最简单的方法是选择范围统一的 RGB 颜色值，它与 DibPalAllPurpose 中的方法相似。此方法的优点是您不必检查 DIB 中的实际图素。这个函式是 DibPalCreateUniformColors，它依据范围统一的 RGB 三原色索引建立调色盘。

一个合理的分布包括 8 阶红色和绿色以及 4 阶蓝色(肉眼对蓝色较不敏感)。调色盘是 RGB 颜色值的集合，它是红色和绿色值 0x00、0x24、0x49、0x6D、0x92、0xB6、0xDB 和 0xFF 以及蓝色值 0x00、0x55、0xAA 和 0xFF 的所有可能的组合，共有 256 种颜色。另一种可能的统一分布调色盘使用 6 阶红色、绿色和蓝色。此调色盘是红色、绿色和蓝色值为 0x00、0x33、0x66、0x99、0xCC 和 0xFF 的所有可能的组合，调色盘中的颜色数是 6 的 3 次方，即 216。

这两个选项和其他几个选项都由 DIBBLE 提供。

「Popularity」演算法

「Popularity」演算法是 256 色调色盘问题相当明显的解决方法。您要做的就是走遍点阵图中的所有图素，并找出 256 种最普通的 RGB 颜色值。这些就是您在调色盘中使用的值。DIBPAL 的 DibPalPopularity 函式中实作了这种演算法。

不过，如果每种颜色都使用整个 24 位元，而且假设需要用整数来计算所有的颜色，那么阵列将占据 64MB 记忆体。另外，您可以发现点阵图中实际上没有（或很少）重复的 24 位元图素值，这样就没有所谓常见的颜色了。

要解决这个问题，您可以只使用每个红色、绿色和蓝色值中最重要的 n 位元——例如，6 位元而不是 8 位元。因为大多数的彩色扫描器和视讯显示卡都只有 6 位元的解析度，所以这样规定更有意义。这将阵列减少到大小更合理的 256KB 或 1MB。只使用 5 位元能将可用的颜色总数减少到 32,768。通常，使用 5 位元要比 6 位元的性能更好。对此，您可以用 DIBBLE 和一些图像颜色来自己检验。

「Median Cut」演算法

DIBPAL.C 中的 DibPalMedianCut 函式执行 Paul Heckbert 的 Median Cut 演算法。此演算法在概念上相当简单，但在程式码中实作要比 Popularity 演算法更困难，它适合递回函式。

画出 RGB 颜色立方体。图像中的每个图素都是此立方体中的一个点。一些点可能代表图像中的多个图素。找出包括图像中所有图素的立体方块，找出此方块的最大尺寸，并将方块分成两个，每个方块都包括相同数量的图素。对於这两个方块，执行相同的操作。现在您就有 4 个方块，将这 4 个方块分成 8 个，然後再分成 16 个、32 个、64 个、128 个和 256 个。

现在您有 256 个方块，每个方块都包括相同数量的图素。取每个方块中图素 RGB 颜色值的平均值，并将结果用於调色盘。

实际上，这些方块通常包含图素的数量并不相同。例如，通常包括单个点的方块会有更多的图素。这发生在黑色和白色上。有时，一些方块里头根本没有图素。如果这样，您就可以省下更多的方块，但是我决定不这样做。

另一种最佳化调色盘的技术称为「octree quantization」，此技术由 Jeff Prosise 提出，并於 1996 年 8 月发表在《Microsoft Systems Journal》上（包含在 MSDN 的 CD 中）。

转换格式

DIBBLE 还允许将 DIB 从一种格式转换到另一种格式。这用到了 DIBCONV 档案中的 DibConvert 函式，如程式 16-25 所示。

程式 16-25　DIBCONV 档案

```
DIBCONV.H

/*------------------------------------------------------------------------
--
      DIBCONV.H header file for DIBCONV.C
------------------------------------------------------------------------
-*/

HDIB DibConvert (HDIB hdibSrc, int iBitCountDst) ;
DIBCONV.C
/*------------------------------------------------------------------------
-
      DIBCONV.C --     Converts DIBs from one format to another
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------------
*/
```

```
#include <windows.h>
#include "dibhelp.h"
#include "dibpal.h"
#include "dibconv.h"

HDIB DibConvert (HDIB hdibSrc, int iBitCountDst)
{
      HDIB                              hdibDst ;
      HPALETTE                    hPalette ;
      int                               i, x, y, cx, cy, iBitCountSrc, cColors ;
      PALETTEENTRY pe ;
      RGBQUAD                     rgb ;
      WORD                              wNumEntries ;

      cx = DibWidth (hdibSrc) ;
      cy = DibHeight (hdibSrc) ;
      iBitCountSrc = DibBitCount (hdibSrc) ;

      if (iBitCountSrc == iBitCountDst)
                        return NULL ;
                  // DIB with color table to DIB with larger color table:
      if ((iBitCountSrc < iBitCountDst) && (iBitCountDst <= 8))
      {
                  cColors = DibNumColors (hdibSrc) ;
                  hdibDst = DibCreate (cx, cy, iBitCountDst, cColors) ;

                  for (i = 0 ; i < cColors ; i++)
                  {
                              DibGetColor (hdibSrc, i, &rgb) ;
                              DibSetColor (hdibDst, i, &rgb) ;
                  }

                  for (x = 0 ; x < cx ; x++)
                  for (y = 0 ; y < cy ; y++)
                  {
                        DibSetPixel (hdibDst, x, y, DibGetPixel (hdibSrc, x, y)) ;
                  }
      }
                  // Any DIB to DIB with no color table
      else if (iBitCountDst >= 16)
      {
                  hdibDst = DibCreate (cx, cy, iBitCountDst, 0) ;
                  for (x = 0 ; x < cx ; x++)
                  for (y = 0 ; y < cy ; y++)
                  {
                              DibGetPixelColor (hdibSrc, x, y, &rgb) ;
                              DibSetPixelColor (hdibDst, x, y, &rgb) ;
```

```
          }
     }
               // DIB with no color table to 8-bit DIB
     else if (iBitCountSrc >= 16 && iBitCountDst == 8)
     {
                    hPalette = DibPalMedianCut (hdibSrc, 6) ;
                    GetObject (hPalette, sizeof (WORD), &wNumEntries) ;

                    hdibDst = DibCreate (cx, cy, 8, wNumEntries) ;
                    for (i = 0 ; i < (int) wNumEntries ; i++)
                    {
                              GetPaletteEntries (hPalette, i, 1, &pe) ;
                              rgb.rgbRed                  = pe.peRed ;
                              rgb.rgbGreen                = pe.peGreen ;
                              rgb.rgbBlue                 = pe.peBlue ;
                              rgb.rgbReserved        = 0 ;

                              DibSetColor (hdibDst, i, &rgb) ;
                    }

                    for (x = 0 ; x < cx ; x++)
                    for (y = 0 ; y < cy ; y++)
                    {
                                   DibGetPixelColor(hdibSrc, x, y, &rgb) ;

                                   DibSetPixel (hdibDst, x, y,
                         GetNearestPaletteIndex (hPalette,
                         RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlue))) ;
                    }
                    DeleteObject (hPalette) ;
          }
                         // Any DIB to monochrome DIB

          else if (iBitCountDst == 1)
          {
                    hdibDst = DibCreate (cx, cy, 1, 0) ;
                    hPalette = DibPalUniformGrays (2) ;

                    for (i = 0 ; i < 2 ; i++)
                    {
                              GetPaletteEntries (hPalette, i, 1, &pe) ;

                                   rgb.rgbRed   = pe.peRed ;
                                   rgb.rgbGreen = pe.peGreen ;
                                   rgb.rgbBlue  = pe.peBlue ;
                                   rgb.rgbReserved = 0 ;

                                   DibSetColor (hdibDst, i, &rgb) ;
```

```
                    }

                    for (x = 0 ; x < cx ; x++)
                    for (y = 0 ; y < cy ; y++)
                    {
                                DibGetPixelColor(hdibSrc, x, y, &rgb);

                                DibSetPixel (hdibDst, x, y,
                          GetNearestPaletteIndex (hPalette,
                          RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlue))) ;
                    }
                    DeleteObject (hPalette) ;
     }
                    // All non-monochrome DIBs to 4-bit DIB
     else if (iBitCountSrc >= 8 && iBitCountDst == 4)
     {
                    hdibDst = DibCreate (cx, cy, 4, 0) ;
                    hPalette = DibPalVga () ;

                    for (i = 0 ; i < 16 ; i++)
                    {
                              GetPaletteEntries (hPalette, i, 1, &pe) ;
                              rgb.rgbRed = pe.peRed ;
                              rgb.rgbGreen = pe.peGreen ;
                              rgb.rgbBlue  = pe.peBlue ;
                              rgb.rgbReserved  = 0 ;

                                   DibSetColor (hdibDst, i, &rgb) ;
                    }

                    for (x = 0 ; x < cx ; x++)
                    for (y = 0 ; y < cy ; y++)
                    {
                                DibGetPixelColor(hdibSrc, x, y, &rgb);

                              DibSetPixel (hdibDst, x, y,
                    GetNearestPaletteIndex (hPalette,
                    RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlue))) ;
                    }
                    DeleteObject (hPalette) ;
     }
                    // Should not be necessary
     else
                    hdibDst = NULL ;
     return hdibDst ;
}
```

将 DIB 从一种格式转换成另一种格式需要几种不同的方法。

要将带有颜色表的 DIB 转换成另一种也带有颜色表但有较大的图素宽度的 DIB（亦即，将 1 位元 DIB 转换成 4 位元或 8 位元 DIB，或将 4 位元 DIB 转换成 8 位元 DIB），所需要做的就是透过呼叫 DibCreate 来建立新的 DIB，并在呼叫时带有希望的位元数以及与原始 DIB 中的颜色数相等的颜色数。然後函式复制图素位元和颜色表项目。

如果新的 DIB 没有颜色表（即位元数是 16、24 或 32），那么 DIB 只需要按新格式建立，而且通过呼叫 DibGetPixelColor 和 DibSetPixelColor 从现有的 DIB 中复制图素位元。

下面的情况可能更普遍:现有的 DIB 没有颜色表(即位元数是 16、24 或 32)，而新的 DIB 每图素占 8 位元。这种情况下，DibConvert 呼叫 DibPalMedianCut 来为图像建立最佳化的调色盘。新 DIB 的颜色表设定为调色盘中的 RGB 值。DibGetPixelColor 函式从现有的 DIB 中获得图素颜色。透过呼叫 GetNearestPaletteIndex 来转换成 8 位元 DIB 中的图素值，并透过呼叫 DibSetPixel 将图素值储存到 DIB。

当 DIB 需要转换成单色 DIB 时，用包括两个项目——黑色和白色——的颜色表建立新的 DIB。另外，GetNearestPaletteIndex 有助於将现有 DIB 中的颜色转换成图素值 0 或 1。类似地，当 8 个图素位元或更多位元的 DIB 要转换成 4 位元DIB时,可从DibPalVga函式获得DIB颜色表,同时GetNearestPaletteIndex也有助於计算图素值。

尽管 DIBBLE 示范了如何开始写一个图像处理程式基础，但是程式最後还是没有全部完成，我们总是会想到还有些功能没有加进去里头。但是很可惜的是，我们现在得停止继续研究这些东西，而往下讨论别的东西了。