

BMP文件格式分析

本来不想写这篇东西，因为介绍BMP文件结构的资料太多了，都有些滥了。但刚写完BMP的读写模块，又不想不留下点什么，所以就写了，全当是学习笔记吧。自己以后查资料时也方便一些，也许对某些初哥还会有点用^^

注：本文参考了[林福宗](#)老师的有关BMP文件格式的文章，在此声明。

简介

BMP(Bitmap-File)图形文件是Windows采用的图形文件格式，在Windows环境下运行的所有图象处理软件都支持BMP图象文件格式。Windows系统内部各图像绘制操作都是以BMP为基础的。Windows 3.0以前的BMP图文件格式与显示设备有关，因此把这种BMP图象文件格式称为设备相关位图DDB(device-dependent bitmap)文件格式。Windows 3.0以后的BMP图象文件与显示设备无关，因此把这种BMP图象文件格式称为设备无关位图DIB(device-independent bitmap)格式（注：Windows 3.0以后，在系统中仍然存在DDB位图，象BitBlt()这种函数就是基于DDB位图的，只不过如果你想将图像以BMP格式保存到磁盘文件中时，微软极力推荐你以DIB格式保存），目的是为了让Windows能够在任何类型的显示设备上显示所存储的图象。BMP位图文件默认的文件扩展名是BMP或者bmp（有时它也会以.DIB或.RLE作扩展名）。

6.1.2 文件结构

位图文件可看成由4个部分组成：位图文件头(bitmap-file header)、位图信息头(bitmap-information header)、彩色表(color table)和定义位图的字节阵列，它具有如下所示的形式。

位图文件的组成	结构名称	符号
位图文件头(bitmap-file header)	BITMAPFILEHEADER	bmfh
位图信息头(bitmap-information header)	BITMAPINFOHEADER	bmih
彩色表(color table)	RGBQUAD	aColors[]





图像数据阵列字节	BYTE	aBitmapBits []
----------	------	-------------------

位图文件结构可综合在表6-01中。

表01 位图文件结构内容摘要

	偏移量	域的名称	大小	内容
图像文件头	0000h	文件标识	2 bytes	<p>两字节的内容用来识别位图的类型：</p> <p>‘BM’ : Windows 3.1x, 95, NT, ...</p> <p>‘BA’ : OS/2 Bitmap Array</p> <p>‘CI’ : OS/2 Color Icon</p> <p>‘CP’ : OS/2 Color Pointer</p> <p>‘IC’ : OS/2 Icon</p> <p>‘PT’ : OS/2 Pointer</p> <p>注：因为OS/2系统并没有被普及开，所以在编程时，你只需判断第一个标识“BM”就行。</p>
	0002h	File Size	1 dword	用字节表示的整个文件的大小
	0006h	Reserved	1 dword	保留，必须设置为0
	000Ah	Bitmap Data Offset	1 dword	从文件开始到位图数据开始之间的数据(bitmap data)之间的偏移量
	000Eh	Bitmap Header Size	1 dword	<p>位图信息头(Bitmap Info Header)的长度，用来描述位图的颜色、压缩方法等。下面的长度表示：</p> <p>28h - Windows 3.1x, 95, NT, ...</p> <p>0Ch - OS/2 1.x</p> <p>F0h - OS/2 2.x</p> <p>注：在Windows95、98、2000等操作系统中，位图信息头的长度并不一定是28h，因为微软已经制定出了新的BMP文件格式，其中的信息头结构变化比</p>

				较大, 长度加长。所以最好不要直接使用常数28h, 而是应该从具体的文件中读取这个值。这样才能确保程序的兼容性。
	0012h	Width	1 dword	位图的宽度, 以像素为单位
	0016h	Height	1 dword	位图的高度, 以像素为单位
	001Ah	Planes	1 word	位图的位面数 (注: 该值将总是1)
图 象 信 息 头	001Ch	Bits Per Pixel	1 word	每个像素的位数 1 - 单色位图 (实际上可有两种颜色, 缺省情况下是黑色和白色。你可以自己定义这两种颜色) 4 - 16 色位图 8 - 256 色位图 16 - 16bit 高彩色位图 24 - 24bit 真彩色位图 32 - 32bit 增强型真彩色位图
	001Eh	Compression	1 dword	压缩说明: 0 - 不压缩 (使用BI_RGB表示) 1 - RLE 8-使用8位RLE压缩方式(用BI_RLE8表示) 2 - RLE 4-使用4位RLE压缩方式(用BI_RLE4表示) 3 - Bitfields-位域存放方式(用BI_BITFIELDS表示)
	0022h	Bitmap Data Size	1 dword	用字节数表示的位图数据的大小。该数必须是4的倍数
	0026h	HResolution	1 dword	用像素/米表示的水平分辨率
	002Ah	VResolution	1 dword	用像素/米表示的垂直分辨率
	002Eh	Colors	1 dword	位图使用的颜色数。如8-比特/像素表示为100h或者 256.
	0032h	Important	1 dword	指定重要的颜色数。当该域的值等于颜色数时 (或者等于0时), 表示所有

		Colors		颜色都一样重要
调色板数据	根据BMP版本的不同而不同	Palette	N * 4 byte	调色板规范。对于调色板中的每个表项, 这4个字节用下述方法来描述RGB的值:  1字节用于蓝色分量  1字节用于绿色分量  1字节用于红色分量  1字节用于填充符(设置为0)
图象数据	根据BMP版本及调色板尺寸的不同而不同	Bitmap Data	xxx bytes	该域的大小取决于压缩方法及图像的尺寸和图像的位深度, 它包含所有的位图数据字节, 这些数据可能是彩色调色板的索引号, 也可能是实际的RGB值, 这将根据图像信息头中的位深度值来决定。

构件详解

1. 位图文件头

位图文件头包含有关于文件类型、文件大小、存放位置等信息, 在Windows 3.0以上版本的位图文件中用BITMAPFILEHEADER结构来定义:

```
typedef struct tagBITMAPFILEHEADER { /* bmfh */
    UINT bfType;
    DWORD bfSize;
    UINT bfReserved1;
    UINT bfReserved2;
    DWORD bfOffBits;
} BITMAPFILEHEADER;
```

其中:

bfType

说明文件的类型。(该值必需是0x4D42, 也就是字符'BM'。我们不需要判断OS/2的位图标识, 这么做现在来看似乎已经没有什么意义了, 而且如果要支持OS/2的位图, 程序将变得很繁琐。所以, 在此只建议你检查'BM'标识)

bfSize

说明文件的大小, 用字节为单位

bfReserved1

保留, 必须设置为0

bfReserved2	保留, 必须设置为0
bfOffBits	说明从文件头开始到实际的图象数据之间的字节的偏移量。这个参数是非常有用的, 因为位图信息头和调色板的长度会根据不同情况而变化, 所以你可以用这个偏移值迅速的从文件中读取到位数据。

2. 位图信息头

位图信息用BITMAPINFO结构来定义, 它由位图信息头(bitmap-information header)和彩色表(color table)组成, 前者用BITMAPINFOHEADER结构定义, 后者用RGBQUAD结构定义。BITMAPINFO结构具有如下形式:

```
typedef struct tagBITMAPINFO { /* bmi */
    BITMAPINFOHEADER bmiHeader;
    RGBQUAD bmiColors[1];
} BITMAPINFO;
```

其中:





bmiHeader	说明BITMAPINFOHEADER结构, 其中包含了有关位图的尺寸及位格式等信息
bmiColors	说明彩色表RGBQUAD结构的阵列, 其中包含索引图像的真实RGB值。

BITMAPINFOHEADER结构包含有位图文件的大小、压缩类型和颜色格式, 其结构定义为:

```
typedef struct tagBITMAPINFOHEADER { /* bmih */
    DWORD biSize;
    LONG biWidth;
    LONG biHeight;
    WORD biPlanes;
    WORD biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    LONG biXPelsPerMeter;
    LONG biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
} BITMAPINFOHEADER;
```

其中:

说明BITMAPINFOHEADER结构所需要的字数。注: 这个值并不一定是BITMAPINFOHEADER结构的尺寸, 它也可

biSize	能是sizeof(BITMAPV4HEADER)的值,或是sizeof(BITMAPV5HEADER)的值。这要根据该位图文件的格式版本来决定,不过,就现在的情况来看,绝大多数的BMP图像都是BITMAPINFOHEADER结构的(可能是后两者太新的缘故吧:-)。
biWidth	说明图象的宽度,以像素为单位
biHeight	说明图象的高度,以像素为单位。注:这个值除了用于描述图象的高度之外,它还有另一个用处,就是指明该图像是倒向的位图,还是正向的位图。如果该值是一个正数,说明图像是倒向的,如果该值是一个负数,则说明图像是正向的。大多数的BMP文件都是倒向的位图,也就是时,高度值是一个正数。(注:当高度值是一个负数时(正向图像),图像将不能被压缩(也就是说biCompression成员将不能是BI_RLE8或BI_RLE4)。
biPlanes	为目标设备说明位面数,其值将总是被设为1
biBitCount	说明比特数/像素,其值为1、4、8、16、24、或32
biCompression	说明图象数据压缩的类型。其值可以是下述值之一: <ul style="list-style-type: none">  BI_RGB: 没有压缩;  BI_RLE8: 每个像素8比特的RLE压缩编码,压缩格式由2字节组成(重复像素计数和颜色索引);  BI_RLE4: 每个像素4比特的RLE压缩编码,压缩格式由2字节组成  BI_BITFIELDS: 每个像素的比特由指定的掩码决定。
biSizeImage	说明图象的大小,以字节为单位。当用BI_RGB格式时,可设置为0
biXPelsPerMeter	说明水平分辨率,用像素/米表示
biYPelsPerMeter	说明垂直分辨率,用像素/米表示
biClrUsed	说明位图实际使用的彩色表中的颜色索引数(设为0的话,则说明使用所有调色板项)
biClrImportant	说明对图象显示有重要影响的颜色索引的数目,如果是0,表示都重要。

现就BITMAPINFOHEADER结构作如下说明:

(1) 彩色表的定位

应用程序可使用存储在biSize成员中的信息来查找在BITMAPINFO结构中的彩色表,如下所示:

```
pColor = ((LPSTR) pBitmapInfo + (WORD) (pBitmapInfo->bmiHeader.biSize))
```

(2) biBitCount

biBitCount=1 表示位图最多有两种颜色, 缺省情况下是黑色和白色, 你也可以自己定义这两种颜色。图像信息头调色板中将有两个调色板项, 称为索引0和索引1。图象数据阵列中的每一位表示一个像素。如果一个位是0, 显示时就使用索引0的RGB值, 如果位是1, 则使用索引1的RGB值。

biBitCount=4 表示位图最多有16种颜色。每个像素用4位表示, 并用这4位作为彩色表的表项来查找该像素的颜色。例如, 如果位图中的第一个字节为0x1F, 它表示有两个像素, 第一个像素的颜色就在彩色表的第2表项中查找, 而第二个像素的颜色就在彩色表的第16表项中查找。此时, 调色板中缺省情况下会有16个RGB项。对应于索引0到索引15。

biBitCount=8 表示位图最多有256种颜色。每个像素用8位表示, 并用这8位作为彩色表的表项来查找该像素的颜色。例如, 如果位图中的第一个字节为0x1F, 这个像素的颜色就在彩色表的第32表项中查找。此时, 缺省情况下, 调色板中会有256个RGB项, 对应于索引0到索引255。

biBitCount=16 表示位图最多有 2^{16} 种颜色。每个像素用16位(2个字节)表示。这种格式叫作高彩色, 或叫增强型16位色, 或64K色。它的情况比较复杂, 当**biCompression**成员的值是BI_RGB时, 它没有调色板。16位中, 最低的5位表示蓝色分量, 中间的5位表示绿色分量, 高的5位表示红色分量, 一共占用了15位, 最高的一位保留, 设为0。这种格式也被称作555 16位位图。如果**biCompression**成员的值是BI_BITFIELDS, 那么情况就复杂了, 首先是原来调色板的位置被三个DWORD变量占据, 称为红、绿、蓝掩码。分别用于描述红、绿、蓝分量在16位中所占的位置。在Windows 95(或98)中, 系统可接受两种格式的位域: 555和565, 在555格式下, 红、绿、蓝的掩码分别是: 0x7C00、0x03E0、0x001F, 而在565格式下, 它们则分别为: 0xF800、0x07E0、0x001F。你在读取一个像素之后, 可以分别用掩码“与”上像素值, 从而提取出想要的颜色分量(当然还要再经过适当的左右移操作)。在NT系统中, 则没有格式限制, 只不过要求掩码之间不能有重叠。(注: 这种格式的图像使用起来是比较麻烦的, 不过因为它的显示效果接近于真彩, 而图像数据又比真彩图像小的多, 所以, 它更多的被用于游戏软件)。

biBitCount=24 表示位图最多有 2^{24} 种颜色。这种位图没有调色板(**bmiColors**成员尺寸为0), 在位数组中, 每3个字节代表一个像素, 分别对应于颜色R、G、B。

biBitCount=32 表示位图最多有 2^{32} 种颜色。这种位图的结构与16位位图结构非常类似, 当**biCompression**成员的值是BI_RGB时, 它也没有调色板, 32位中有24位用于存放RGB值, 顺序是: 最高位—保留, 红8位、绿8位、蓝8位。这种格式也被成为888 32位图。如果**biCompression**成员的值是BI_BITFIELDS时, 原来调色板的位置将被三个DWORD变量占据, 成为红、绿、蓝掩码, 分别用于描述红、绿、蓝分量在32位中所占的位置。在Windows 95(or 98)中, 系统只接受888格式, 也就是说三个掩码的值将只能是: 0xFF0000、0xFF00、0xFF。而在NT系统中, 你只要注意使掩码之间不产生重叠就行。(注: 这种图像格式比较规整, 因为它是DWORD对齐的, 所以在内存中进行图像处理时可进行汇编级的代码优化(简单))。

(3) ClrUsed

BITMAPINFOHEADER结构中的成员**ClrUsed**指定实际使用的颜色数目。如果**ClrUsed**设置成0, 位图使用的颜色数目就等于**biBitCount**成员中的数目。请注意, 如果**ClrUsed**的值不是可用颜色的最大值或不是0, 则在编程时应该注意调色板尺寸的计算, 比如在4位位图中, 调色板的缺省尺寸应该是 $16 * \text{sizeof}(\text{RGBQUAD})$, 但是, 如果**ClrUsed**的值不是16或者不是0, 那么调色板的尺寸就应该是**ClrUsed * sizeof(RGBQUAD)**。

(4) 图象数据压缩

① **BI_RLE8**: 每个像素为8比特的RLE压缩编码, 可使用编码方式和绝对方式中的任何一种进

行压缩, 这两种方式可在同一幅图中的任何地方使用。

编码方式: 由2个字节组成, 第一个字节指定使用相同颜色的像素数目, 第二个字节指定使用的颜色索引。此外, 这个字节对中的第一个字节可设置为0, 联合使用第二个字节的值表示:

- ◆ 第二个字节的值为0: 行的结束。
- ◆ 第二个字节的值为1: 图象结束。
- ◆ 第二个字节的值为2: 其后的两个字节表示下一个像素从当前开始的水平和垂直位置的偏移量。

绝对方式: 第一个字节设置为0, 而第二个字节设置为0x03~0xFF之间的一个值。在这种方式中, 第二个字节表示跟在这个字节后面的字节数, 每个字节包含单个像素的颜色索引。压缩数据格式需要字边界(word boundary)对齐。下面的例子是用16进制表示的8-位压缩图象数据:

03 04 05 06 00 03 45 56 67 00 02 78 00 02 05 01 02 78 00 00 09 1E 00 01
这些压缩数据可解释为 :

压缩数据	扩展数据
03 04	04 04 04
05 06	06 06 06 06 06
00 03 45 56 67 00	45 56 67
02 78	78 78
00 02 05 01	从当前位置右移5个位置后向下移一行
02 78	78 78
00 00	行结束
09 1E	1E 1E 1E 1E 1E 1E 1E 1E 1E
00 01	RLE编码图象结束

② BI_RLE4: 每个像素为4比特的RLE压缩编码, 同样也可使用编码方式和绝对方式中的任何一种进行压缩, 这两种方式也可在同一幅图中的任何地方使用。这两种方式是:

编码方式: 由2个字节组成, 第一个字节指定像素数目, 第二个字节包含两种颜色索引, 一个在高4位, 另一个在低4位。第一个像素使用高4位的颜色索引, 第二个使用低4位的颜色索引, 第3个使用高4位的颜色索引, 依此类推。

绝对方式: 这个字节对中的第一个字节设置为0, 第二个字节包含有颜色索引数, 其后续字节包含有颜色索引, 颜色索引存放在该字节的高、低4位中, 一个颜色索引对应一个像素。此外, BI_RLE4也同样联合使用第二个字节中的值表示:

- ◆ 第二个字节的值为0: 行的结束。
- ◆ 第二个字节的值为1: 图象结束。
- ◆ 第二个字节的值为2: 其后的两个字节表示下一个像素从当前开始的水平和垂直位置的偏移量。

下面的例子是用16进制数表示的4-位压缩图象数据:

03 04 05 06 00 06 45 56 67 00 04 78 00 02 05 01 04 78 00 00 09 1E 00 01

这些压缩数据可解释为：

压缩数据	扩展数据
03 04	0 4 0
05 06	0 6 0 6 0
00 06 45 56 67 00	4 5 5 6 6 7
04 78	7 8 7 8
00 02 05 01	从当前位置右移5个位置后向下移一行
04 78	7 8 7 8
00 00	行结束
09 1E	1 E 1 E 1 E 1 E 1
00 01	RLE图象结束

3. 彩色表

彩色表包含的元素与位图所具有的颜色数相同，象素的颜色用RGBQUAD结构来定义。对于24-位真彩色图象就不使用彩色表（同样也包括16位、和32位位图），因为位图中的RGB值就代表了每个象素的颜色。彩色表中的颜色按颜色的重要性排序，这可以辅助显示驱动程序为不能显示足够多颜色数的显示设备显示彩色图象。RGBQUAD结构描述由R、G、B相对强度组成的颜色，定义如下：

```
typedef struct tagRGBQUAD { /* rgbq */
    BYTE rgbBlue;
    BYTE rgbGreen;
    BYTE rgbRed;
    BYTE rgbReserved;
} RGBQUAD;
```

其中：

rgbBlue	指定蓝色强度
rgbGreen	指定绿色强度
rgbRed	指定红色强度
rgbReserved	保留，设置为0

4. 位图数据

紧跟在彩色表之后的是图象数据字节阵列。图象的每一扫描行由表示图象象素的连续的字节组成，每一行的字节数取决于图象的颜色数目和用象素表示的图象宽度。扫描行是由底向上存储的，这就是说，阵列中的第一个字节表示位图左下角的象素，而最后一个字节表示位图右上角的象素。（只针对与倒向DIB，如果是正向DIB，则扫描行是由顶向下存储的），倒向DIB的原点在图像的左下角，而正向DIB的原点在图像的左上角。同时，每一扫描行的字节数必需是4的整倍数，也就是DWORD对齐的。如果你想确保图像的扫描行DWORD对齐，可使用下面的代码：

$$(((width*biBitCount)+31)>>5)<<2$$

5. 参考书目

《图象文件格式(上、下)—Windows编程》
《图像文件格式大全》
《Programming Windows by Charles Petzold》

6. 相关站点

各种格式: <http://www.wotsit.org/>
各种格式: <http://www.csdn.net/>
位图格式: http://www.cica.indiana.edu/graphics/image_specs/bmp.format.txt

〈完〉

YZ 2000-8-13 13:11