BMP 图像旋转大作业参考阅读

BMP 图像旋转大作业参考阅读

1. BMP 图像文件格式介绍（一）

BMP 是 bitmap 的缩写形式，它一般由 4 部分组成：

- 文件头信息块
- 图像描述信息块
- 颜色表（在真彩色模式下无颜色表）
- 图像数据区

在系统中以 BMP 为扩展名保存。

现在讲解 BMP 的 4 个组成部分：

1.1. 文件头信息块

- `0000-0001`：文件标识，为字母 “BM” 的 ASCII 码。
- `0002-0005`：文件大小。
- `0006-0009`：保留，每字节以 “`00`” 填写。
- `000A-000D`：记录图像数据区的起始位置。各字节的信息含义依次为：文件头信息块大小、图像描述信息块大小、图像颜色表的大小、

保留（为 01）。

1.2. 图像描述信息块

- 000E-0011：图像描述信息块的大小，常为 28H。
- 0012-0015：图像宽度。
- 001A-001B：图像的 plane 总数（恒为 1）。
- 001C-001D：记录像素的位数，图像的颜色数由该值决定。
- 001E-0021：数据压缩方式（数值位为 0：不压缩；数值位为 1：8 位压缩；数值位为 2：4 位压缩。
- 0022-0025：图像区数据的大小。
- 0026-0029：水平方向每米有多少像素。在设备无关位图（.DIB）中，每字节以 00H 填写。
- 002A-002D：垂直方向每米有多少像素。在设备无关位图（.DIB）中，每字节以 00H 填写。
- 002E-0031：此图像所用的颜色数，如值为 0，表示所有颜色一样重要。

1.3. 颜色表

颜色表的大小根据所使用的颜色模式而定：2 色图像为 8 字节；16 色图像为 64 字节；256 色图像为 1024 字节。其中，每 4 字节表示一种颜色，并以 B（蓝色）、G（绿色）、R（红色）、alpha（像素的透明度值，一般不需要）的顺序排列。即首先 4 字节表示颜色号 0 的颜色，接下来颜色号 1 的颜色，依此类推。

1.4. 图像数据区

颜色表之后为位图文件的图像数据区，在此部分记录着每点像素对应的颜色号，其记录方式也随颜色模式而定：2 色图像每点占 1 位（8 位为 1 字节），16 色图像每点占 4 位（半字节），256 色图像每点占 8 位（1 字节），真彩色图像每点占 24 位（3 字节）。所以，整个数据区的大小也会随颜色模式而变化。图像数据信息大小有如下计算公式：

```
图像数据信息大小 =（图像宽度 * 图像高度 * 记录像素的位数）/ 8
```

然而，未压缩的图像数据区的大小，除了真彩色模式外，其余的均大于或等于数据信息的大小。这是为什么呢？原因有两个：

1. BMP 文件记录一行图像是以字节为单位的。因此，就不存在一个字节中的数据位信息表示的点在不同的两行中。例如，设显示模式为 16 色（每点像素占 4 位），在每个字节分配两个点的信息时，如果图像的宽度位为奇数，那么最后一个像素点的信息将独占一个字节，

这个字节的后 4 位将没有意义。接下来的一个字节将开始记录下一行的信息。

2. 为了显示的方便，除了真彩色外，其他的每种颜色模式的行字节数都要用数据 “00” 补齐为 4 的整数倍。如果显示模式为 16 色，当图像宽为 19 时，存储时每行则要补充 `4 -（19 / 2 + 1）% 4 = 2` 个字节（加 1 是因为最后一个像素点独占了 1 字节）。如果显示模式为 256 色，当图像宽为 19 时，每行也要补充 `4 - 19 % 4 = 1` 个字节。

还有一点我要申明，当屏幕初始化为 16 或 256 色模式时，一定要设置调色板或修正颜色值，否则无法得到正确的图像颜色。

2. BMP 图像文件格式介绍（二）

对于现存的所有图像文件格式，我们在这里主要介绍 BMP 文件格式，并且文件里的图像数据是未压缩的。因为图像的数字化处理主要是对图像中的各个像素进行相应的处理，而未压缩的 BMP 图像中的像素数值正好与实际要处理的数字图像相对应，这种格式的文件最适合我们进行数字化处理。请读者记住，压缩过的图像是无法直接进行数字化处理的，如 JPEG、GIF 等格式的文件，此时首先要对图像文件解压缩，这就要涉及一些比较复杂的压缩算法。

2.1. BMP 文件组成

BMP 文件由文件头、位图信息头、颜色信息和图像数据四部分组成。文件头主要包含文件的大小、文件类型、图像数据偏离文件头的长度等信息；位图信息头包含图像的尺寸信息、图像用几个比特数值来表示一个像素、图像是否压缩、图像所用的颜色数等信息；颜色信息包含图像所用到的颜色表，显示图像时需用到这个颜色表来生成调色板，但如果图像为真彩色，即图像的每个像素用 24 个比特来表示，文件中就没有这一块信息，也就不需要操作调色板；文件中的数据块表示图像的相应的像素值。需要注意的是：图像的像素值在文件中的存放顺序为从左到右、从下到上，也就是说，在 BMP 文件中首先存放的是图像的最后一行像素，最后才存放图像的第一行像素，但对于同一行的像素，则是按照从左到右的顺序存储的。另外一个值得注意的细节是：文件存储图像的每一行像素值时，如果存储该行像素值所占的字节数为 4 的倍数，则正常存储；否则，需要在每行的末端补 0，凑足为 4 的倍数。

2.2. BMP 文件头

BMP 文件头数据结构含有 BMP 文件的类型、文件大小和位图起始位置等信息。其结构定义如下：

```
typedef struct tagBITMAPFILEHEADER
{
    WORD bfType;            // 位图文件的类型，必须为"BM"
    DWORD bfSize;           // 位图文件的大小，以字节为单位
    WORD bfReserved1;       // 位图文件保留字，必须为0
    WORD bfReserved2;       // 位图文件保留字，必须为0
    DWORD bfOffBits;        // 位图数据的起始位置，以相对于位图文件头的
偏移量表示，以字节为单位
} BITMAPFILEHEADER;
```

该结构占据 14 个字节。

2.3. 位图信息头

BMP 位图信息头数据用于说明位图的尺寸等信息。其结构如下：

```
typedef struct tagBITMAPINFOHEADER
{
    DWORD biSize;           // 本结构所占用字节数
    LONG biWidth;           // 位图的宽度，以像素为单位
    LONG biHeight;          // 位图的高度，以像素为单位
    WORD biPlanes;          // 目标设备的平面数，必须为1
    WORD biBitCount;        // 每个像素所需的位数，必须是1（双色）、
4（16色）、8（256色）或24（真彩色）之一
    DWORD biCompression;    // 位图压缩类型，必须是0（不压缩）、
1（BI_RLE8压缩类型）或2（BI_RLE4压缩类型）之一
    DWORD biSizeImage;      // 位图的大小，以字节为单位
    LONG biXPelsPerMeter;   // 位图水平分辨率，每米像素数
    LONG biYPelsPerMeter;   // 位图垂直分辨率，每米像素数
    DWORD biClrUsed;        // 位图实际使用的颜色表中的颜色数
    DWORD biClrImportant;   // 位图显示过程中重要的颜色数
} BITMAPINFOHEADER;
```

该结构占据 40 个字节。

注意：对于 BMP 文件格式，在处理单色图像和真彩色图像时，无论图像数据多么庞大，都不对图像数据进行任何压缩处理。一般情况下，如果位图采用压缩格式，那么 16 色图像采用 RLE4 压缩算法，256 色图像采用 RLE8 压缩算法。

2.4. 颜色表

颜色表用于说明位图中的颜色，它有若干个表项，每一个表项是一个 RGBQUAD 类型的结构，定义一种颜色。RGBQUAD 结构的定义如下：

```
typedef struct tagRGBQUAD
{
    BYTE rgbBlue;          // 蓝色的亮度（取值范围为0-255）
    BYTE rgbGreen;         // 绿色的亮度（取值范围为0-255）
    BYTE rgbRed;           // 红色的亮度（取值范围为0-255）
    BYTE rgbReserved;      // 保留，必须为0
} RGBQUAD;
```

颜色表中 RGBQUAD 结构数据的个数由 BITMAPINFOHEADER 中的 biBitCount 项来决定，当 biBitCount 为 1、4、8 时，分别有 2、16、256 个颜色表项；当 biBitCount 为 24 时，图像为真彩色，图像中每个像素的颜色用三个字节表示，分别对应 R、G、B 值，图像文件中没有颜色表项。

位图信息头和颜色表组成位图信息。其结构如下：

```
typedef struct tagBITMAPINFO
{
    BITMAPINFOHEADER bmiHeader;  // 位图信息头
    RGBQUAD bmiColors[1];        // 颜色表
} BITMAPINFO;
```

注意：RGBQUAD 数据结构中，增加了一个保留字段 rgbReserved，它不代表任何颜色，必须取固定的值 “0”，同时，RGBQUAD 结构定义的颜色值中，红色、绿色和蓝色的排列顺序与一般真彩色图像文件的颜色数据排列顺序恰好相反，即：若某个位图中的一个像素点的颜色的描述为 “00,00,ff,00”，则表示该点为红色，而不是蓝色。

2.5. 位图数据

位图数据记录了位图的每一个像素值或该对应像素的颜色表的索引值，图像记录顺序在扫描行内时是从左到右、扫描行之间时是从下到上。这种格式我们又称为 Bottom_Up 位图。当然与之相对的还有 Up_Down 形式的位图，它的记录顺序是从下到上的。位图的一个像素值所占的字节数：

	2 色	16 色	256 色	真彩
颜色表字节数	8 字节	64 字节	1024 字节	无颜色表项
像素点字节数	1 位	4 位	1 字节	3 字节

如前所述，Windows 规定图像文件中一个扫描行所占的字节数必须是 4 的倍数（即以字为单位），不足的以 0 填充。图像文件中一个扫描行所占的字节数计算方法：

```
DataSizePerLine = (biWidth * biBitCount + 31) / 8  // 一个扫描行所占的字节数
```

位图数据的大小按下式计算（不压缩情况下）：

```
DataSize = DataSizePerLine * biHeight
```

上述是 BMP 文件格式的说明，搞清楚了以上的结构，就可以正确地操作图像文件、对它进行读或写操作了。

3. GIF 图像文件格式

GIF 图像文件格式的全称为 Graphics Interchange Format，从这个名字可以看出，这种图像格式主要是为了通过网络传输图像而设计的。GIF 文件不支持 24 位真彩色图像，最多只能存储 256 色的图像或灰度图像；GIF 格式文件也无法存储 CMY 和 HIS 模型的图像数据。另外，GIF 图像文件的各种数据区域一般没有固定的数据长度和存储顺序，所以为了方便程序寻找数据区，将数据区中的第一个字节作为标志符。最后需要读者注意的是 GIF 文件存储图像数据有两种排列顺序：顺序排列和交叉排列。交叉排列的方式适合网络传输，这样一来便允许用户在不完全掌握图像数据之前，获取当前图像的轮廓数据。

GIF 文件格式分为 87 和 89 两个版本，该文件主要由 5 个部分组成，它们是按顺序出现的：文件头块、逻辑屏幕描述块、可选择的调色板块、图像数据块，最后是标志文件结束的尾块，该块总是取固定的值 3BH。其中第一个和第二个块用 GIF 图像文件头结构描述如下：

```
typedef struct tagGIFHEADER
{
    DB Signature;           // 该字段占6个字节，为了用于指明图像为GIF格式，前3个字符必须为"GIF"，后3个字符用于指定是哪个版本，87或89
    DW ScreenWidth;
    DW ScreenDepth;         // 占2个字节，以像素为单位表示图像的宽、高
    DB GlobalFlagByte;      // 该字节的各个位用于调色板的描述
    DB BackgroundColor;     // 代表图像的背景颜色的索引
    DB AspectRatio;         // 图像的长宽比
} GIFHEADER;
```

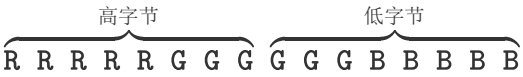
由于 GIF 格式允许一个文件存储多个图像，因此 GIF 格式中的调色板有通用调色板和局部调色板之分。其中通用调色板适用于文件中的所有图像，而局部调色板只适用于某一个图像。格式中的数据区域一般分为 4 个部分：图像数据识别区域、局部调色板区域、采用压缩算法得到的图像数据区域和结束标志区域。

在 GIF89 版本中，它包含 7 个部分，分别是文件头、通用调色板数据、图像数据区和 4 个补充数据区，它们主要是用于提示程序如何处理图像的。

4. RGB 格式

RGB1、GRB4、RGB8 都是调色板类型的 RGB 格式，在描述这些媒体类型的格式细节时，通常会在 `BITMAPINFOHEADER` 数据结构后面跟一个调色板（定义一系列颜色）。它们的图像数据并不是真正的颜色值，而是当前像素颜色值在调色板中的索引。以 RGB1 （2 色位图）为例，若它的调色板中定义的两种颜色值依次为 `0x000000`（黑色）和 `0xFFFFFF`（白色），那么图像数据 `001101010111...`（每个像素用 1 位表示）对应各像素的颜色为：黑黑白白黑白黑白黑白白白.....

RGB565 使用 16 位表示一个像素，这 16 位中的 5 位用于 R，6 位用于 G，5 位用于 B。程序中通常使用一个字（`WORD`，一个字等于两个字节）来操作一个像素。当读出一个像素后，这个字的各个位意义如下：



可以组合使用屏蔽字和移位操作来得到 RGB 各分量的值：

```
#define RGB565_MASK_RED 0xF800
#define RGB565_MASK_GREEN 0x07E0
#define RGB565_MASK_BLUE 0x001F
R = (wPixel & RGB565_MASK_RED) >> 11;    // 取值范围0-31
G = (wPixel & RGB565_MASK_GREEN) >> 5;   // 取值范围0-63
B = wPixel & RGB565_MASK_BLUE;           // 取值范围0-31
```

RGB555 是另一种 16 位的 RGB 格式，RGB 分量都用 5 位表示（剩下的 1 位不用）。使用一个字读出一个像素后，这个字的各个位意义如下：

可以组合使用屏蔽字和移位操作来得到 RGB 各分量的值：

```
#define RGB555_MASK_RED 0x7C00
#define RGB555_MASK_GREEN 0x03E0
#define RGB555_MASK_BLUE 0x001F
R = (wPixel & RGB555_MASK_RED) >> 10;    // 取值范围0-31
G = (wPixel & RGB555_MASK_GREEN) >> 5;   // 取值范围0-31
B = wPixel & RGB555_MASK_BLUE;           // 取值范围0-31
```

RGB24 使用 24 位来表示一个像素，RGB 分量都用 8 位表示，取值范围为 0-255。注意在内存中 RGB 各分量的排列顺序为：BGR BGR BGR ...

通常可以使用 RGBTRIPLE 数据结构来操作一个像素，它的定义为：

```
typedef struct tagRGBTRIPLE
{
    BYTE rgbtBlue;      // 蓝色分量
    BYTE rgbtGreen;     // 绿色分量
    BYTE rgbtRed;       // 红色分量
} RGBTRIPLE;
```

RGB32 使用 32 位来表示一个像素，RGB 分量各用去 8 位，剩下的 8 位用作 Alpha 通道或者不用。注意在内存中 RGB 各分量的排列顺序为：BGRA BGRA BGRA ...

通常可以使用 RGBQUAD 数据结构来操作一个像素，它的定义为：

```
typedef struct tagRGBQUAD
{
    BYTE rgbBlue;       // 蓝色分量
    BYTE rgbGreen;      // 绿色分量
    BYTE rgbRed;        // 红色分量
    BYTE rgbReserved;   // 保留字节（用作Alpha通道或忽略）
} RGBQUAD;
```