CS162 – Programming Techniques

BMP File

Cảm ơn thầy Trần Duy Quang đã cung cấp template cho môn học

Department of Software Engineering-FIT-VNU-HCMUS

1

BMP file format

The BMP file format, also known as bitmap image file or device independent bitmap (DIB) file format or simply a bitmap, is a raster graphics image file format used to store bitmap digital images, independently of the display device[1].

1.1 File structure

Component	Size (byte)	Description
Bitmap file header	14	Store general information about the bitmap image file
DIB header	40	Store detailed information about the bitmap image and define the pixel format
Color table	Variable-size	Define colors used by the bitmap image data (Pixel array)
Pixel array	Variable-size	Define the actual values of the pixels

1.2 Bitmap file header

Offset (HEX)	Size (byte)	Description
00	2	"BM" string
02	4	Size of bitmap file in bytes
06	2	Reserved
08	2	Reserved
0A	4	The offset of byte where the bitmap array can be found

1.3 DIB header

Offset (HEX)	Size (byte)	Description
0E	4	The size of DIB header (40 bytes)
12	4	The bitmap width in pixels
16	4	The bitmap height in pixels
1A	2	Number of color planes (1)
1C	2	Color depth (1, 4, 8, 16, 32)
1E	4	Compression method (0)
22	4	Pixel array size

[1] http://en.wikipedia.org/wiki/BMP_file_format

26	4	The horizontal resolution of the image. (pixel per meter, signed integer)
2A	4	The vertical resolution of the image. (pixel per meter, signed integer)
2E	4	The number of colors in the color palette, or 0 to default to 2^n
32	4	The number of important colors used, or 0 when every color is important; generally ignored

1.4 Color table

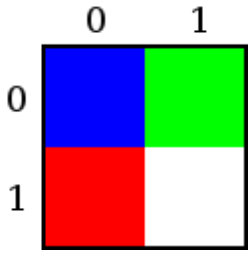
- 24-bit images or higher: Ignore
- < 24-bit images:

Offset (HEX)	Size (byte)	Description
36	4	Color format Blue, Green, Red, Alpha
…		

1.5 Pixel array

Offset (HEX)	Size (byte)	Description
[54 + size of color table]	Depend on color depth	Color of first pixel - 24-bit images or higher: Blue, Green, Red - < 24-bit images: Index of color in color table
…		

1.6 Example



Offset	Size	Hex Value	Value	Description
BMP Header				
0h	2	42 4D	"BM"	ID field (42h, 4Dh)
2h	4	46 00 00 00	70 bytes (54+16)	Size of the BMP file

6h	2	00 00	Unused	Application specific
8h	2	00 00	Unused	Application specific
Ah	4	36 00 00 00	54 bytes (14+40)	Offset where the pixel array (bitmap data) can be found
DIB Header				
Eh	4	28 00 00 00	40 bytes	Number of bytes in the DIB header (from this point)
12h	4	02 00 00 00	2 pixels (left to right order)	Width of the bitmap in pixels
16h	4	02 00 00 00	2 pixels (bottom to top order)	Height of the bitmap in pixels. Positive for bottom to top pixel order.
1Ah	2	01 00	1 plane	Number of color planes being used
1Ch	2	18 00	24 bits	Number of bits per pixel
1Eh	4	00 00 00 00	0	BI_RGB, no pixel array compression used
22h	4	10 00 00 00	16 bytes	Size of the raw bitmap data (including padding)
26h	4	13 0B 00 00	2835 pixels/meter horizontal	Print resolution of the image, 72 DPI × 39.3701 inches per meter yields 2834.6472
2Ah	4	13 0B 00 00	2835 pixels/meter vertical	
2Eh	4	00 00 00 00	0 colors	Number of colors in the palette
32h	4	00 00 00 00	0 important colors	0 means all colors are important
Start of pixel array (bitmap data)				
36h	3	00 00 FF	0 0 255	Red, Pixel (0,1)
39h	3	FF FF FF	255 255 255	White, Pixel (1,1)
3Ch	2	00 00	0 0	Padding for 4 byte alignment (could be a value other than zero)
3Eh	3	FF 00 00	255 0 0	Blue, Pixel (0,0)
41h	3	00 FF 00	0 255 0	Green, Pixel (1,0)
44h	2	00 00	0 0	Padding for 4 byte alignment (could be a value other than zero)

2

Source code

```
struct BmpSignature
{
    unsigned char data[2];
};

#pragma pack(1)
struct BmpHeader
{
    BmpSignature signature;
    uint32_t fileSize;
    uint16_t reserved1;
    uint16_t reserved2;
    uint32_t dataOffset;
};
```

```
struct BmpDib
{
    uint32_t dibSize;
    int32_t  imageWidth;
    int32_t  imageHeight;
    uint16_t colorPlaneCount;
    uint16_t pixelSize;
    uint32_t compressMethod;
    uint32_t bitmapByteCount;
    int32_t  horizontalResolution;
    int32_t  verticalResolution;
    uint32_t colorCount;
    uint32_t importantColorCount;
};
```

```
struct Color
{
    unsigned char blue;
    unsigned char green;
    unsigned char red;
};
```

```
struct ColorTable
{
    Color    *colors;
    uint32_t length;
};
```

```
struct PixelArray
{
    Color    **pixels;
    uint32_t rowCount;
    uint32_t columnCount;
};
```

```
#include <Windows.h>
bool isBmpFile(FILE *f)
{
    if (f == NULL)
        return false;

    BmpSignature signature;
    fseek(f, 0, 0L);
    fread(&signature, sizeof(BmpSignature), 1, f);

    return signature.data[0] == 'B' && signature.data[1] == 'M';
}
```

```
void readBmpHeader(FILE *f, BmpHeader &header)
{
    if (f == NULL)
        return;

    fseek(f, 0, 0L);
    fread(&header, sizeof(BmpHeader), 1, f);
}
```

```
void printBmpHeader(BmpHeader header)
{
    printf("*** BMP Header ***\n");
    printf("- File Size   : %d byte(s)\n", header.fileSize);
    printf("- Reserved1   : %d\n", header.reserved1);
    printf("- Reserved2   : %d\n", header.reserved2);
    printf("- Data Offset: %d byte(s)\n", header.dataOffset);
}

void readBmpDib(FILE *f, BmpDib &dib)
{
    if (f == NULL)
        return;

    fseek(f, sizeof(BmpHeader), 0L);
    fread(&dib, sizeof(BmpDib), 1, f);
}

void printBmpDib(BmpDib dib)
{
    printf("*** BMP Dib ***\n");
    printf("- DIB Size               : %d byte(s)\n", dib.dibSize);
    printf("- Image Width            : %d\n", dib.imageWidth);
    printf("- Image Height           : %d\n", dib.imageHeight);
    printf("- Number of Color Planes : %d\n", dib.colorPlaneCount);
    printf("- Pixel Size             : %d bit(s)\n", dib.pixelSize);
    printf("- Compress Method        : %d\n", dib.compressMethod);
    printf("- Bitmap Size            : %d byte(s)\n", dib.bitmapByteCount);
    printf("- Horizontal Resolution  : %d\n", dib.horizontalResolution);
    printf("- Vertical Resolution    : %d\n", dib.verticalResolution);
    printf("- Number of Colors       : %d\n", dib.colorCount);
    printf("- Number of Impt Colors  : %d\n", dib.importantColorCount);
}
```

```
void readBmpPixelArray(FILE *f, BmpHeader header, BmpDib dib, PixelArray &data)
{
    if (f == NULL)
        return;

    data.rowCount = dib.imageHeight;
    data.columnCount = dib.imageWidth;
    data.pixels = new Color*[data.rowCount];

    char paddingCount = (4 - (dib.imageWidth * (dib.pixelSize / 8) % 4)) % 4;

    fseek(f, header.dataOffset, 0L);

    for (int i = 0; i < data.rowCount; i++)
    {
        scanBmpPixelLine(f, data.pixels[data.rowCount - 1 - i], dib.imageWidth);
        skipBmpPadding(f, paddingCount);
    }
}
```

```
void scanBmpPixelLine(FILE *f, Color *&line, uint32_t length)
{
    if (f == NULL)
        return;

    line = new Color[length];
    fread(line, sizeof(Color), length, f);
}
```

```
void skipBmpPadding(FILE *f, char count)
{
    if (f == NULL)
        return;

    if (count == 0)
        return;

    char padding[3];
    fread(padding, count, 1, f);
}
```

```
void drawBmp(BmpDib dib, PixelArray data)
{
    HWND console = GetConsoleWindow();
    HDC hdc = GetDC(console);

    for (int i = 0; i < dib.imageHeight; i++)
        for (int j = 0; j < dib.imageWidth; j++)
        {
            Color pixel = data.pixels[i][j];
            SetPixel(hdc, j, i, RGB(pixel.red, pixel.green, pixel.blue));
        }

    ReleaseDC(console, hdc);
}
```

```
void releaseBmpPixelArray(PixelArray data)
{
    for (int i = 0; i < data.rowCount; i++)
        delete []data.pixels[i];

    delete []data.pixels;
}
```