Formato Bitmap

Canoi Gomes

23 de Junho de 2024

Conteúdo

1 Introdução

No texto anterior eu dei uma rápida explicação sobre como funciona um arquivo binário, como construir o seu próprio e fazer um programa para escrever e ler.

O formato de arquivo bitmap (possui a extensão **.bmp**) é um formato normalmente utilizado para guardar os dados de pixel de uma image. É um dos formatos que são conhecidos por geralmente não ter compressão, então diferente de um .png, onde existe um algoritmo de compressão para codificar e decodificar, resultando em um arquivo com tamanho menor. Já no bitmap não, nós conseguimos calcular exatamente qual será o tamanho do arquivo, se temos uma image 24x24 com pixels RGB (ou seja, meu pixel é representado por 3 bytes, 1 para cada canal), nós podemos fazer o seguinte para determinar o tamanho do arquivo:

$24 * 24 = 576$

$576 * 3 = 1728$

Ou seja, nosso arquivo terá por volta de 1.7 kb, isso desconsiderando o cabeçalho.

2 Cabeçalho

O formato bitmap possui dois cabeçalhos, um **cabeçalho de arquivo** e um **cabeçalho de informação**. O de arquivo tem 14 bytes, e o de informação de 40 bytes.

Basic BMP File Format			
Name		**Size**	**Description**
Header		14 bytes	Windows Structure: BITMAPFILEHEADER
	Signature	2 bytes	'BM'
	FileSize	4 bytes	File size in bytes
	reserved	4 bytes	unused (=0)
	DataOffset	4 bytes	File offset to Raster Data
InfoHeader		40 bytes	Windows Structure: BITMAPINFOHEADER
	Size	4 bytes	Size of InfoHeader =40
	Width	4 bytes	Bitmap Width
	Height	4 bytes	Bitmap Height
	Planes	2 bytes	Number of Planes (=1)
	BitCount	2 bytes	Bits per Pixel 1 = monochrome palette. NumColors = 1 4 = 4bit palletized. NumColors = 16 8 = 8bit palletized. NumColors = 256 16 = 16bit RGB. NumColors = 65536 (?) 24 = 24bit RGB. NumColors = 16M
	Compression	4 bytes	Type of Compression 0 = BI_RGB no compression 1 = BI_RLE8 8bit RLE encoding 2 = BI_RLE4 4bit RLE encoding
	ImageSize	4 bytes	(compressed) Size of Image It is valid to set this =0 if Compression = 0
	XpixelsPerM	4 bytes	horizontal resolution: Pixels/meter
	YpixelsPerM	4 bytes	vertical resolution: Pixels/meter
	ColorsUsed	4 bytes	Number of actually used colors
	ColorsImportant	4 bytes	Number of important colors 0 = all
ColorTable		4 * NumColors bytes	present only if Info.BitsPerPixel <= 8 colors should be ordered by importance
	Red	1 byte	Red intensity
	Green	1 byte	Green intensity
	Blue	1 byte	Blue intensity
	reserved	1 byte	unused (=0)
	repeated NumColors times		
Raster Data		Info.ImageSize bytes	The pixel data

2.1 Cabeçalho de Arquivo

O cabeçalho de arquivo possui 14 bytes de tamanho, como descrito na imagem acima:

Validação	**2 bytes**	É usado para a validação do formato, seria seu magic number, geralmente é preenchido com ”BM”.
Tamanho	**4 bytes**	Tamanho do arquivo em bytes.
Reservado	**4 bytes**	Geralmente não é utilizado.
Offset	**4 bytes**	A partir de onde podemos começar a ler os dados dos pixels do arquivo. Em uma imagem padrão, que não tenha informações extra antes dos pixels, esse valor é 54 geralmente, que seria a soma dos valores de ambos os cabeçalhos (14 + 40).

Poderia representá-lo assim em C.

```c
struct FileHeader {
    char magic[2];
    char size[4];
    char reserved[4];
    char offset[4];
};
```

Listing 1: Cabeçalho de Arquivo em C

2.2 Cabeçalho de Informação

O cabeçalho de informação tem 40 bytes de tamanho, e é nele que geralmente vão as informações relativas a imagem como sua paleta de cores, a quantidade de bits que estou usando para representar um pixel, as dimensões, entre outras coisas.

Campo	Bytes	Descrição
Tamanho do Cabeçalho	**4 bytes**	Aqui irá conter o tamanho do cabeçalho em bytes, geralmente esse valor será 40 mesmo.
Largura	**4 bytes**	Largura da imagem
Altura	**4 bytes**	Altura da imagem
Planos	**2 bytes**	Número de planos (também não sei para que serve, geralmente é 1)
Bits Por Pixel	**2 bytes**	Esse campo é interessante, nele eu especifico a quantidade de bits que quero usar para representar um pixel. É comum esse valor ser 24 (RGB com 3 bytes, ou seja 1 byte para cada canal) ou 32 (RGBA com 4 bytes). Porém também pode ser 16 (também é RGB, mas é um RGB565, 5 bits para R, 6 para G e 5 para B). Se setarmos valores abaixo de 16, como 8 ou 4, precisamos especificar uma paleta de cores logo depois do cabeçalho. 8 bits por exemplo, preciso de uma paleta com 256 cores, 4 bits uma paleta com 16 cores e assim vai.
Compressão	**4 bytes**	Aqui basicamente você pode setar um modo de compressão, mas como não estamos utilizando nenhum geralmente esse valor será zero mesmo.
Tamanho da Imagem	**4 bytes**	Aqui vai representar o tamanho da imagem após a compressão. Se não tiver nenhum compressão, como é o nosso caso, esse valor pode ser zero também.

XpixelsPerM	**4 bytes**	
YpixelsPerM	**4 bytes**	
Cores Usadas	**4 bytes**	
Cores Importantes	**4 bytes**	

Posso representar esse cabeçalho assim em C:

```c
struct InfoHeader {
    char header_size[4];
    char width[4];
    char height[4];
    char planes[2];
    char bits_per_pixel[2];
    char compression[4];
    char image_size[4];
    char x_pixels[4];
    char y_pixels[4];
    char colors_used[4];
    char colors_important[4];
};
```

Listing 2: Cabeçalho de Informação em C