Formato de Arquivos Gráficos

Darren Meyer, WPI CS Department
Tradução: Bruno Freitas Scalon

INTRODUÇÃO

Uma das áreas-chave na construção de qualquer programa gráfico é a abilidade de importar e exportar gráficos usando formatos de arquivos padrão. Aqui discutiremos três populares formatos de arquivos gráficos e dois processos de compressão de bitmaps. Os três tipos de arquivo são o Microsoft Windows Bitmap (BMP), o Adobe Encapsulated PostScript (EPS), e o CompuServe Graphics Interchange Format (GIF). Os processos de compressão são o Run-Length Encoding (RLE) e o Limpel-Ziv-Welsh Compression (LZW).

Note que este documento contém uma discussão sobre os processos de compressão e a estrutura do formato dos arquivos, e não necessariamente toda informação necessária para uma implementação (apesar de que para alguns essa informação esteja presente). As referências no fim fornecem informações de onde obter as especificações completas de cada um. Porém, esse documento fornece explicações que não estão presentes nas especificações completas.

COMPRESSÃO RUN-LENGTH

A codificação Run-length é muito simples, intuitiva, e é um método de compressão muito eficiente para bitmaps. Seu principal conceito é detectar pixels repetidos em cada coluna de uma imagem e convertê-los em uma contagem de pixels e em um par de valores, ao invés de convertê-los repetidamente e individualmente. A compressão RLE não funciona bem para **stipple patterns** ou imagens scaneadas, que não possuem pixels repetidos em colunas - o tamanho desses tipos de imagem normalmente ficam maiores após a compressão RLE. Apesar dessa limitação, o RLE é muito bom para outros tipos de imagem, e é suportado pelos formatos BMP, TIFF, e Apple MacPaint, entre outros.

Quando fazemos compressão run-length, toda a informação original é scaneada em colunas e convertida em bytes de contagem (pares de grupos de informação). Há dois tipos de agrupamentos que podem ser feitos. O primeiro é uma série, que é um grupo de valores de pixels repetidos processado como uma contagem de pixels seguidos pelo seu

valor. O segundo tipo de agrupamento é a seqüência, que é um grupo de bytes não repetidos, que é processado como uma contagem de pixels seguido de seus valores. Todas as séries e seqüências não devem se extender além da linha scaneada para deixar a lógica mais simples.

Para distinguir entre séries e seqüências, uma série de n bytes recebe a contagem 1 - n, enquanto que uma seqüência de j bytes recebe a contagem j - 1. O exemplo abaixo mostra a compressão RLE para uma série de bytes que tem cinco ocorrências do byte A e uma seqüência BCDE. A série é processada com a contagem -4 seguida pelo valor A, e a seqüência é processada com a contagem 3 seguida dos valores BCDE.

```
        Original:  AAAAABCDE
  Processado:  -4A3BCDE
```

O código fonte para fazer a descompactação RLE a partir de um arquivo aparece abaixo:

```
for (y = 0; y < HEIGHT; y++)
  {
    int x = 0;
    while ( x < WIDTH )
     {
       int count = fgetc (infile);
       if (count < 0 )          /* Process a run */
        {
          int i = 1 - count;
          char c = fgetc (infile);
          while ( i-- )
            image[x++][y] = c;
        }
       else                     /* Process a sequence */
        {
          int i = count + 1;
          while ( i-- )
            image[x++][y] = fgetc (infile);
        }
     }
        }
```

A compressão RLE é mais complexa que a descompressão, o que é comum para muitos processos de compactação. Isso ocorre porque, geralmente, expressões de entrada de dados podem ser codificadas de várias maneiras, e geralmente é difícil determinar qual é a melhor. Marv

Luse, o autor da minha principal fonte para este documento, diz que os ganhos que se consegue na compressão geralmente não justificam o desenvolvimento completo de um codificador otimizado, principalmente por isso representar mais trabalho e maior lentidão. Ao contrário disso, ele propõe o uso de um codificador simples porém inteligente.

COMPRESSÃO LZW

A compressão LZW é muito popular devido seu uso nas imagens GIF e à alta compressão que ela consegue virtualmente em todas as imagens. Foi lançada em 1977 por J. Ziv e A. Limpel, e foi mais tarde refinada por T. Welsh. Depois de ser usada gratuitamente por muitos anos, a UniSys Corporation notificou a CompuServe entre outras que eles detém a patente do processo. A UniSys decidiu então que seria permitido o uso gratuito do processo apenas para fins não comerciais, mas usuários comerciais deveriam obter uma licença para o uso. A compressão LZW é usada pelo formato GIF e é uma das compressões suportadas pelo formato TIFF.

O objetivo da compressão LZW é substituir valores de entrada repetidos por códigos n-bit. Isso é feito gerando-se uma tabela de valores, que é um mapeamento entre os valores de pixel e os códigos de compressão. Essa tabela é construida pelo compressor durante o processamento dos dados, e devido ao método de codificação o decodificador pode reconstruir essa tabela enquanto processa as informações comprimidas. Isso difere dos outros algorítimos de compressão, como a codificação de Huffman, onde a tabela de consulta deve ser incluida com a informação comprimida.

A LZW trabalha baseada no fato de que muitos agrupamentos de pixels são comuns em imagens: ela lê as informações da imagem e tenta codificar o maior número de agrupamentos de pixels que puder através da tabela de valores, posicionando grupos desconhecidos na tabela de valores para que possam ser comprimidos em ocorrências futuras. Para uma imagem com valores de pixel n-bit, ela usa códigos de compressão que possuem n + 1 bits ou mais. Enquanto um código de compressão menor ajuda a conseguir maiores quantidades de compressão, o tamanho do código de compressão limita o tamanho da tabela. Por exemplo, um arranjo comum com código de compressão de 12 bit para imagens com 8 bit por pixel, que permite 4096 entradas na tabela de valores: se o codificador fica sem espaço na tabela, um compressor LZW tradicional aborta o processo e tenta codificar novamente porém com um código de compressão maior. (A versão GIF do compressor LZW é preparada para lidar com essas situações).

No início da compressão ou descompressão, a tabela de valores é preparada para conter todos os valores possíveis de pixels da imagem (chamados raízes). Depois que isso é feito, o codificador rastreia valores de pixels na imagem até encontrar algum que não está na tabela. Ele então codifica os valores encontrados, mas não o valor novo, e os transporta para a tabela para poderem ser reconhecidos posteriormente. Ele continua o processo até que toda imagem esteja processada. Abaixo está o código fonte para um codificador LZW:

```
/* Put the roots in the string table */
for (int i = 0; i < MAXPIXEL; i++)
  put i in the string table as entry i

/* Encode the data */
char *prefix = NULL;
char *string;
while ( have more input )
  {
    char c = fgetc (input);
    string = append c to prefix;
    if ( string is in string table )
      {
        prefix = string;
      }
    else
      {
        add string to string table;
        output code for the current prefix;
        prefix = string with only c;
      }
  }
output code for the current prefix;
```

O formato GIF extende o código básico do LZW para acrescentar duas raízes adicionais na tabela: um código para exluir valores e um fim para o código de entrada. O código para excluir valores é usado quando a tabela está cheia, e faz com que ela seja reiniciada para conter somente as raízes. O fim do código de entrada é usado para simplificar a lógica do processo de descompressão.

O FORMATO BMP

O formato BMP é o formato Microsoft Windows Bitmap para Device Independent Bitmaps (DIBs). Ele contém imagens com 1, 4, 8, ou 24

bits por pixel, e é armazenado na forma de linhas scaneadas, de baixo para cima (o que causa problemas já que quase todos os formatos gráficos armazenam informações de cima para baixo). Há duas versões incompatíveis desse formato, uma introduzida com a versão 1.x do OS/2 e a outra com o Windows 3.0. Como o formato do Windows 3.0 é mais recente e mais comum, é o que será descrito aqui. O formato BMP suporta imagens RLE de 4 e 8 bit por pixel, mas essa facilidade é raramente usada, mesmo pela Microsoft. O arquivo inteiro é ampliado para ser uniformemente divisível por 4 bytes para um acesso otimizado. A ordem dos bytes é de tal forma que o bit menos significante vem sempre primeiro (padrão Intel).

O formato BMP é muito popular nos computadores pessoais baseados no chip da Intel já que é suportado pelos aplicativos da Microsoft e pode ter seu processamento otimizado pelos processadores da Intel. Por ser simples e compatível com outros sistemas, é uma boa alternativa para novos aplicativos.

O início do arquivo BMP é usado para indicar o formato e outras informações básicas. Seus membros aparecem abaixo:

unsigned short type = 'BM' (ASCII), indicates it is a BMP file
unsigned long, file size in bytes
unsigned short, reserved
unsigned short, reserved
unsigned long, offset in bytes to the image data

O setor de informações do arquivo BMP é usado para guardar todas as informações sobre as dimensões da imagem. Seus membros aparecem abaixo:

unsigned long, size of the header in bytes
long width, width of the image in pixels
long height, height of the image in pixels
unsigned short, the number of image planes, currently always 1
unsigned short, the number of bits for each image pixel
unsigned long, a code to indicate the compression type
unsigned long, the size of the image in bytes
long, the number of x pixels per meter
long, the number of y pixels per meter
unsigned long, the number of colors used
unsigned long, the number of colors considered important

Os dados da paleta BMP são identificados por quatro caracteres que indicam as componentes azuis, verdes e vermelhas da imagem, nessa ordem. O quarto byte é reservado. A ordem estranha da especificação RGB e o uso de um byte reservado permitem carregar as informações numa integral com ordem apropriada nos processadores Intel. Há uma entrada de paleta para cada cor da imagem com as cores mais importantes, se houver, em primeiro lugar.

Um arquivo BMP completo contém no início do arquivo as informação métricas, como descrito acima. Depois vem as entradas da paleta, com as informações da imagem por último. Lembre-se que o arquivo é ampliado para ser divisível em grupos de 4 bytes.

FORMATO EPS

O formato EPS é o Encapsulated PostScript format da Adobe. Para entender esse formato de arquivo, é necessário uma explicação básica sobre PostScript. PostScript é um formato de arquivo usado basicamente por todas as impressoras de alta resolução, e por algumas de média resolução também. O PostScript foi feito para armazenar documentos inteiros, incluindo os textos, gráficos (curvas, linhas, etc...), e imagens bitmap. Todo o texto e os objetos gráficos dos arquivos PostScript são redimensionados para a resolução máxima suportada pela impressora, ou qualquer outro equipamento utilizado, o que ajuda na popularização do PostScript. PostScript é extremamente complexo e sofisticado, pois é baseado em uma linguagem de programação **stack-based** completa. Por exemplo, ele suporta cálculos, sub-rotinas, variáveis, comentários e muitos outros conceitos de linguagem de programação. É até mesmo possível escrever um programa completo de ray-tracing em PostScript. O programa e o resto das informações podem então serem descarregados em um impressora, serem executados e finalmente impressos!

Os arquivos PostScript contém também uma parte legível com a descrição do documento. Por exemplo, se estiver escrito "2 3 add" isso quer dizer que o processador deve inserir um " 2" na memória, inserir um "3" na memória, e depois executar uma adição. Esse comando recupera esses dois itens da memória, faz a adição, e insere o resultado de volta na memória. Assim a adição de três números pode ser feita escrevendo-se "1 2 3 add add" ou "1 2 add 3 add" chegando-se ao mesmo resultado no final. Há vários livros disponíveis sobre a linguagem PostScript se você quiser maiores informações. Eles estão listados no final deste texto.

Devido à complexidade do PostScript, o PostScript Document Structuring Conventions (DSC) foi introduzido para ajudar a definir a ordem das tarefas sem a necesidade de um arquivo PostScript. Os arquivos DSC também contém comentários para indicar sua estruturação entre outras informações adicionais, como o título do documento, o autor, etc. Apesar disso, a interpretação do PostScript ainda é muito difícil. Muitos aplicativos oferecem exportação de PostScript, o que é relativamente fácil implementar, para permitir impressões de alta resolução. Mas muito poucos interpretam o PostScript.

Agora, de volta aos arquivos EPS. Eles são simplesmente arquivos PostScript que seguem as instruçõs DSC, e deve conter somente a descrição de uma única página. O PostScript não apresenta muitas vantagens na compressão de bitmaps. A complexidade do PostScript faz do formato EPS uma má opção para armazenamento de arquivos bitmap. Porém, como é relativamente fácil colocar um arquivo EPS dentro de um documento PostScript, o formato EPS é geralmente a melhor opção para exportar gráficos para aplicativos que trabalham com documentos baseados em layouts. A exportação de arquivos EPS geralmente é fácil de ser implementada em aplicativos que já suportam o formato PostScript para impressões.

Os arquivos EPS devem conter um preview em baixa resolução quando se trata de uma imagem padrão, como um TIFF numa máquina com sistema UNIX, um Windows Metafile BMP em um PC Intel Intel 80x86, ou um PICT em um Machintosh. Muitos programas que dizem importar arquivos EPS na verdade apenas interpretam esse preview, que muitas vezes é muito menos do que satisfatório.

Como os arquivos EPS e PostScript têm um formato meio livre e com muitas operações complexas, eles não serão mais discutidos aqui. Para maiores informações, olhe as referências no fim desse documento.

FORMATO GIF

O formato GIF foi desenvolvido pela CompuServe para ser usado com um modem. Isso explica sua alta compressão, sua estrutura orientada, e o interlaçamento das linhas da imagem (o que permite ver a base da imagem antes de receber todo o arquivo). Como ele usa a compressão LZW, patenteada pela UniSys, é necessário uma licença para uso comercial. O formato GIF pertence à CompuServe, mas para usá-lo basta ter seu copyright na documentação impressa. O formato GIF tem duas versões: o formato 87a foi introduzido em 1987, e define a

estrutura básica. A versão 89a extendeu o formato para permitir comentários específicos. Quando for gravar arquivos GIF que não precisam das extensões 89a, o uso do formato 87a é recomendado. Os arquivos GIF suportam 8 bits por pixel ou menos.

Apesar da limitação no número de bits por pixel e seu design original para transmissões via modem, o cabeçalho com informações completas e sua alta compressão tem feito o formato GIF extremamente popular para imagens rasterizadas. É portanto uma ótima escolha para aplicativos que trabalham com imagens rasterizadas e desejam possuir funções como importação e/ou exportação de imagens. A edição de Maio de 1995 da revista MacWorld lançou uma reportagem informando que a CompuServe está trabalhando em um "novo padrão" para permitir imagens de 24 bit e esquemas de compressão públicos, o que deve significar ainda mais extensões ou até mesmo um formato totalmente novo.

A estrutura do arquivo GIF é complexa, pois ele deve poder conter qualquer número de imagens e de extensões. A composição básica é a seguinte:

File Header, either "GIF87a" or "GIF89a" in ASCII
Logical Screen Descriptor
Global Color Table (optional)
Block Definitions
File Trailer, ";" in ASCII

A seção Logical Screen Descriptor começa com a largura e altura do dispositivo físico que criou a imagem. Depois vem um byte que diz se há uma tabela de cores global, quantos bits por pixel há na imagem, se a tabela de cores está classificada em ordem de importância e o número de itens da tabela de cores. Depois vem o índice da cor de fundo e uma especificação do **aspect ratio** (razão entre a altura e a largura).

A Tabela Global de Cores só está presente se as especificações do Logical Screen Descriptor contém um sinal que indica sua presença. Normalmente ela está presente, e contém uma série de trincas RGB que são codificadas como um byte para cada um dos três componentes. Se estiver classificada por ordem de importância, os itens mais importantes vêm antes. Cada imagen de um arquivo GIF pode conter sua própria tabela de cores local. E se ela tiver a tabela global é ignorada.

As especificações da propriedade Block Definitions contém o número de imagens e extensões 89a, em qualquer ordem. O tipo de bloco (imagem

ou extensão) pode ser determinado pelo primeiro byte do bloco. Os blocos de imagem têm a seguinte definição:

Logical Image Descriptor
Local Color Table (optional)
Minimum LZW Code Size
Image data
Block Terminator (a byte containing 0x00 hexadecimal)

As especificações da propriedade **Logical Image Descriptors** começam com o caracter ',' no formato ASCII, seguido das coordendas do canto superior esquerdo da imagem, em pixels. Depois vêm a largura e altura da imagem, também em pixels. A última parte de cada **descriptor** é um byte que indica se a imagem tem uma Tabela de Cores Local, se ela está classificada em ordem de importância, e o número de itens dessa tabela.

A Tabela de Cores Local está presente somente se um sinal apropriado está presente nas especificações do **Logical Image Descriptor**. Sua estrutura é idêntica à Tabela de Cores Global.

O tamanho mínimo do código LZW é o tamanho do início do código da imagem. O tamanho máximo para arquivos GIF é 12 bits, e o tamanho do código inicial é 3 para imagens de 1 bit por pixel e normalmente o número de bits por pixel mais 1 para outras imagens.

As informações da imagem consistem de um byte para indicar o tamanho dos dados da imagem no bloco (1 - 255 bytes) seguido pelos dados da imagem comprimidos no formato LZW. Pode haver qualquer quantidade de bytes indicadores de tamanho seguidos por bytes da imagem propriamente dita na seção de dados da imagem.

Para Blocos de Extensão, as informações seguintes estão presente::

Extension Introducer (a byte containing 0x21 hexadecimal)
Extension Type (a byte with the extension number)
Extension data
Block Terminator (a byte containing 0x00 hexadecimal)

Os dados de extensão estão no mesmo formato dos dados da imagem, com um byte indicando o tamanho, seguido pelos dados correspondentes. Novamente, pode haver qualquer quantidade de bytes indicadores de tamanho seguidos por bytes de dados. A interpretação dos dados depende do tipo da extensão. Há atualmente quatro

extensões: uma para especificar comentários, uma para armazenar informações específicas do aplicativo, uma para anotações em forma de texto, e a última especifica como alternar entre imagens de um mesmo arquivo GIF.

CONCLUSÃO

Para escrever um aplicativo que usa 8 bit de cores ou menos, o formato GIF é uma escolha excelente, pois suporta vários aplicativos em várias plataformas. O BMP também é uma boa escolha, apesar de seu suporte ser mais limitado em plataformas não compatíveis com a Intel. Finalmente, programas sofisticados devem suportar a abilidade da impressão PostScript e exportação no formato EPS.

Para aplicativos que desejam usar seu próprio formato de arquivos, as compressões RLE e LZW são bons algorítimos para reduzir o tamanho dos dados. A RLE é simples de ser implementada e é gratuita, enquanto que a LZW normalmente possui uma melhor performance mas requer uma licença para uso comercial.

REFERÊNCIAS

Muitas das informações abaixo foram obtidas do livro "Bitmapped Graphics Programming in C++" de Marv Luse. Além de conter uma excelente descrição sobre formatos de arquivo e formas de compressão, também possui uma excelente discussão sobre **dithering** (mistura de cores) e controle de cores. Finalmente, e talvez o mais importante para muitas pessoas, inclui o código C++ completo para implementar a leitura e escrita de cada formato de arquivo e esquema de compressão. Seu código C++ é bem simples, o que possibilita sua conversão para uso em programas em C de forma relativamente simples.

As especificações oficiais do formato GIF podem ser obtidas em:
ftp://ftp.hawaii.edu/mirrors/info-mac/info/dev/gif-format-gif89a.txt

Se você deseja incluir suporte à PostScript e/ou EPS em seu programa, você precisará do manual oficial do PostScript, o "PostScript Language Reference Manual" da Adobe Systems Incorporated. Algumas estações UNIX vêm com manuais de PostScript ou versões online do mesmo.