Reading the Windows BMP Graphic File Format

Intro

Any discussion that involves graphics of any kind eventually covers some graphic file format. Now here's the hard part. Which file format is going to best suit the graphics needed. There are literally hundreds of them. One of the most popular is the windows .bmp format. I like it because its very easy to read and extremely easy to use. It has the capacity of using a compression scheme , but this is very rare indeed. Every bitmap starts with a header. There are actually 2 different headers, but I've put them into one for easy of use and simplicity. Here it is!

The Code

```
struct Fileheader
 { unsigned short Type;          // signature - 'BM'
   unsigned  long Size;          // file size in bytes
   unsigned short Reserved1;     // 0
   unsigned short Reserved2;     // 0
   unsigned long  OffBits;       // offset to bitmap
   unsigned long  StructSize;    // size of this struct (40)
   unsigned long  Width;         // bmap width in pixels
   unsigned long  Height;        // bmap height in pixels
   unsigned short Planes;        // num planes - always 1
   unsigned short BitCount;      // bits per pixel
   unsigned long  Compression;   // compression flag
   unsigned long  SizeImage;     // image size in bytes
   long           XPelsPerMeter; // horz resolution
   long           YPelsPerMeter; // vert resolution
   unsigned long  ClrUsed;       // 0 -> color table size
   unsigned long  ClrImportant;  // important color count
   Fileheader()
   {Size=Width=Height=Planes=BitCount=Compression=SizeImage=XPelsPerMeter=
    YPelsPerMeter=ClrUsed=ClrImportant=Type=StructSize=Reserved1=Reserved2=OffBits=0;
   }
};
```

Most of the elements are pretty self explanatory. I only use a couple of elements from this structure, the others may or may not be supported depending on the graphics program you are using. Another point to mention is that the Size element can't be trusted. I've found a few programs that fill it in with garbage. If the color depth is 8 bits or lower there will be a color palette following this structure. Its number of elements depends on how many bits per pixel there is. Here's one Palette entry:

```
struct RGBQUAD
{ unsigned char rgbBlue;
  unsigned char rgbGreen;
  unsigned char rgbRed;
  unsigned char rgbReserved;
  RGBQUAD( )
  {rgbBlue = rgbGreen = rgbRed = 0;
   rgbReserved = 0;
  }
};
```

Now theoretically our image is going to appear immediately after the color palette has been read, if there is one. For safety you might want to rewind the stream and advance the file pointer to OffBits according to the structure. From there we can read in SizeImage bytes and we'll have the data! Here's a function that will pull it off!

In header file:

```
 Fileheader f;
 unsigned char* Bmp;
 RGBQUAD Palette[256];
```

In Cpp file:

```
void Bitmap::ReadBitmap(char* FileName)
{ FILE *stream=fopen(FileName,"rb");

 if(stream ==NULL)
  {cout<<"ERROR: Can't open "<<FileName<<endl;
  }
 else
  { if((fread(&f,sizeof(Fileheader),1,stream)) == -1)
      {cout<<"ERROR: Can't Read Fileheader Stucture!";
      }

    for(int l=0;l<256;l++)
      {if((fread(&Palette[l],sizeof(RGBQUAD),1,stream))==-1)
          {cout<<"ERROR: Can't read bitmap's color palette!";
          }
      }
     long size=(((f.Width * (f.BitCount / 8)) + 3) & (~3))*f.Height;

     Bmp = new unsigned char[size];

    if(Bmp==NULL)
      {cout<<"ERROR: Can't allocate memory for image!";
      }
     else
```

```
    { if((fread((unsigned char*)Bmp,(long)size,1,stream))== -1)
       {cout<<"ERROR: Can't read in image!";
       }
    }
 cout<<FileName<<" "<<f.Width<<"x"<<f.Height<<endl;
  }
}
```

Whatever you do, don't try reading a bitmap in text mode, it just won't work well :). That accounts for the "rb" settings when we open it up. If our stream is logically equal to NULL, then we know that it wasn't successful so print an error message. Inside the else, which assumes our open was ok, we've got the main section that actually reads in the data. First off lets try to read the bitmap header (Fileheader). If that equals -1, then we know that it didn't work so print out the appropriate error message. The next section attempts to read in the color palette. Here we are assuming that it will be 256 entries which I hardcode since I only use 256 color bitmaps! After that we allocate our unsigned char array, double checking that we actually got the requested amount. The final section does one fread and checks to see if it was successful! Before exiting, we write our filename and the dimensions of the bitmap to the screen. That's it! The size variable calculations are explained below.

From here you can start getting into displaying and doing some special effects with the image data. The only odd thing worth mentioning is that bitmaps are stored in reverse scan line order. Come again? Yep, think of the image flipped up side down. You can make it easier on yourself and flip it right side up before moving on, or modify your functions to traverse the bitmap from bottom to top, your choice :) Also, I've included a couple of else's only in vital areas of this function. To be complete you should decide what to do if such errors occur, and place them next to the error messages. Then decide wether you can proceed or not. For instance it won't be that big of a deal if we can't read in the palette of the image if we already have a system palette that covers all the bitmaps we are using...get it?

Yet another weird aspect of bitmaps is that their width is DWORD aligned meaning that it is padded to be a multiple of 4. The code as it stands reads in the entire bitmap with padding. If you choose, you could read in the bitmap line by line and clip off the extra padding. Here's an easy formula to figure out the real bitmap width:

```
PhysicalWidth = ((Width * (BitsPerPixel / 8)) + 3) & (~3)
ScanPadding = PhysicalWidth - (Width * ( BitsPerPixel / 8 ));
```

That's all the mystery I can reveal form now! If you have any question ,comments, complaints whatever! send me some Feedback.

Contact Information

I just wanted to mention that everything here is copyrighted, feel free to distribute this document to anyone you want, just don't modify it! You can get a hold of me through my website or direct email. Please feel free to email me about anything.

Email : deltener@mindtremors.com
Webpage : http://www.invsereality.org