Raster Data Tutorial

Author: Bill Green (2002)
HOME EMAIL

INTRODUCTION

A BMP computer image is the easiest to understand because it does not use compression, making pixel data retrieval much easier. The table below shows how the pixel data is stored from the first byte to the last.

TABLE 1: BMP File Structure

Byte # to fseek file pointer	Information
0	Signature
2	File size
18	Width (number of columns)
22	Height (number of rows)
28	Bits/pixel
46	Number of colors used
54	Start of color table
54 + 4*(number of colors)	Start of raster data

The first 14 bytes are dedicated to the header information of the BMP. The next 40 bytes are dedicated towards the info header, where one can retrieve such characteristics as width, height, file size, and number of colors used. Next, is the color table, which is 4 x (number of colors used) bytes long. So for an 8-bit grayscale image (number of colors is 256), the color table would be 4 x 256 bytes long or 1024 bytes. And the last bit of data in a BMP file is the pixel data, or raster data. The raster data starts at byte 54 (header + info header) + 4 x number of colors (color table). For an 8-bit grayscale image, the raster data would start at byte 54 + 1024 = 1078. The size of the raster data is (width x height) – 1 bytes. Therefore, a 100 row by 100 column 8-bit grayscale image would have (100 x 100) – 1 = 9,999 bytes of raster data starting at byte 1078 and continuing to the end of the BMP.

In terms of image processing, the most important information is the following:
(1) Number of columns – byte #18
(2) Number of rows - byte #22
(3) Raster data – byte #(4 x number of colors) to byte #1078 + (number of columns x number of rows) – 1

In C, the most efficient way of declaring this important information is that of struct.

```
typedef struct { int rows;                /* number of rows */
                 int cols;                /* number of columns */
                 unsigned char* data;   /* raster data */
               }  sImage;
```

READING BMP RASTER DATA

TEST.bmp is a 20 row by 20 column BMP image which we will use to read raster data from. In an 8 bit

BMP image, black is 0 and white is 255. The top left corner of TEST.bmp starts at a pixel value of 0 (black) and progressively works its way down the diagonal to pixel value of 255 (white). Thinking of rows and columns in a BMP is not the same as thinking of rows and columns in a matrix. In a matrix row 0 and column 0 would start you at the top left corner of the matrix. However, in a BMP, the rows increase from bottom to top. Therefore, row 0 and column 0 in a BMP would correspond to the bottom left corner.

(TEST.bmp is scaled up here to a 100 by 100 BMP,
so be sure and download the zip file to test out your raster data program.)

TEST.bmp contains 20 rows and 20 columns, so we know we will have 400 bytes of raster data. We also know the raster data will start at byte #(54 + 4 x number of colors). The number of colors of TEST.bmp is 256 because it is a grayscale image with colors ranging from 0 to 255. Therefore, the raster data will start at byte #1078 and the file size will be 1078 + 400 = 1478 bytes. Knowing this, let's try our first program to read raster data and print it to a text file.

To be compiled with Turbo C

Note: download raster.zip rather than cutting and pasting from below.

```
#include (stdio.h)
#include (stdlib.h)
#include (math.h)

/*-------STRUCTURES---------*/
typedef struct {int rows; int cols; unsigned char* data;} sImage;

/*-------PROTOTYPES---------*/
long getImageInfo(FILE*, long, int);

int main(int argc, char* argv[])
{
  FILE                  *bmpInput, *rasterOutput;
  sImage                originalImage;
  unsigned char         someChar;
  unsigned char*        pChar;
  int                   nColors;  /* BMP number of colors */
  long                  fileSize; /* BMP file size */
  int                   vectorSize; /* BMP vector size */
  int                   r, c;       /* r = rows, c = cols */


  /* initialize pointer */
  someChar = '0';
  pChar = &someChar;

  if(argc < 2)
  {
    printf("Usage: %s bmpInput.bmp\n", argv[0]);
    exit(0);
  }
  printf("Reading filename %s\n", argv[1]);

  /*--------READ INPUT FILE------------*/
  bmpInput = fopen(argv[1], "rb");
  fseek(bmpInput, 0L, SEEK_END);

  /*--------DECLARE OUTPUT TEXT FILE--------*/
```

```
  rasterOutput = fopen("data.txt", "w");

  /*--------GET BMP DATA---------------*/
  originalImage.cols = (int)getImageInfo(bmpInput, 18, 4);
  originalImage.rows = (int)getImageInfo(bmpInput, 22, 4);
  fileSize = getImageInfo(bmpInput, 2, 4);
  nColors = getImageInfo(bmpInput, 46, 4);
  vectorSize = fileSize - (14 + 40 + 4*nColors);

  /*-------PRINT DATA TO SCREEN-------------*/
  printf("Width: %d\n", originalImage.cols);
  printf("Height: %d\n", originalImage.rows);
  printf("File size: %ld\n", fileSize);
  printf("# Colors: %d\n", nColors);
  printf("Vector size: %d\n", vectorSize);

  /*----START AT BEGINNING OF RASTER DATA-----*/
  fseek(bmpInput, (54 + 4*nColors), SEEK_SET);

  /*----------READ RASTER DATA----------*/
  for(r=0; r<=originalImage.rows - 1; r++)
  {
    for(c=0; c<=originalImage.cols - 1; c++)
    {
      /*-----read data and print in (row,column) form----*/
      fread(pChar, sizeof(char), 1, bmpInput);
      fprintf(rasterOutput, "(%d, %d) = %d\n", r, c, *pChar);
    }
  }

  fclose(bmpInput);
  fclose(rasterOutput);


}


/*----------GET IMAGE INFO SUBPROGRAM--------------*/
long getImageInfo(FILE* inputFile, long offset, int numberOfChars)
{
  unsigned char         *ptrC;
  long                  value = 0L;
  unsigned char         dummy;
  int                   i;

  dummy = '0';
  ptrC = &dummy;

  fseek(inputFile, offset, SEEK_SET);

  for(i=1; i<=numberOfChars; i++)
  {
    fread(ptrC, sizeof(char), 1, inputFile);
    /* calculate value based on adding bytes */
    value = (long)(value + (*ptrC)*(pow(256, (i-1))));
  }
  return(value);

} /* end of getImageInfo */
```

Running your raster data program, you will get an ASCII file called data.txt with some entries looking like the following:

```
(0, 0) = 255
(0, 1) = 255
```

```
(0, 2) = 255
(0, 3) = 255
     :
(1, 0) = 255
(1, 1) = 255
(1, 2) = 255
(1, 3) = 255
     :
(2, 0) = 255
(2, 1) = 255
(2, 2) = 255
(2, 3) = 255
     :
(3, 0) = 255
(3, 1) = 255
(3, 2) = 255
(3, 3) = 255
     :
(4, 0) = 255
(4, 1) = 255
(4, 2) = 255
     :
(4, 14) = 207
(4, 15) = 207
(4, 16) = 255
(4, 17) = 255
(4, 18) = 255
(4, 19) = 255
     :
(5, 0) = 255
(5, 1) = 255
(5, 2) = 255
(5, 3) = 255
     :
(5, 14) = 207
(5, 15) = 207
     :
(6, 0) = 255
(6, 1) = 255
(6, 2) = 255
(6, 3) = 255
     :
(6, 12) = 159
(6, 13) = 159
     :
(7, 0) = 255
(7, 1) = 255
(7, 2) = 255
(7, 3) = 255
     :
(7, 12) = 159
(7, 13) = 159
(7, 14) = 255
     :
(8, 0) = 255
(8, 1) = 255
(8, 2) = 255
     :
(8, 10) = 111
(8, 11) = 111
(8, 12) = 255
     :
(9, 0) = 255
(9, 1) = 255
(9, 2) = 255
(9, 3) = 255
     :
(9, 10) = 111
(9, 11) = 111
```

```
(9, 12) = 255
        :
(10, 0) = 255
(10, 1) = 255
(10, 2) = 255
(10, 3) = 255
        :
 (10, 8) = 79
 (10, 9) = 79
(10, 10) = 255
        :
```

EXPLANATION

Notice how entry (4, 14) is 207 meaning that the pixel in the row 4, column 14 is very close to white. This is not the case if you are thinking in terms of matrices. Just remember that in BMPs, the raster data is stored from left to right and bottom to top.

The program begins by opening the BMP file entered at the command prompt. For example if your program is called raster.c, at the command prompt you would enter:

raster test.bmp

We want to read a binary BMP file which required us to add "rb" in our fopen statement for read binary. We also have to set the file pointer to the beginning of the BMP file – byte #0. We accomplish these 2 tasks with the following lines of code:

```
bmpInput = fopen(argv[1], "rb");
fseek(bmpInput, 0L, SEEK_END);
```

From the pixel data table we see that byte #18 contains the width of the BMP file. So to get the corresponding number of columns we have to set the file pointer to byte number 18. We also have to set the pointer to read the height, number of colors, file size, vector size, etc. Instead of tediously performing this task repeatedly, it would be much easier to write a subprogram that does it for you each time it is called in main. The subprogram is a function which means it returns a value to main. For example, to get the width of a BMP, all you would have to type is:

```
originalImage.rows = (int) getImageInfo(bmpInput, 18, 4);
```

The 18 corresponds to where you want to start reading data from and the 4 corresponds to how many bytes you want to read.

MANIPULATING A BMP FILE

As our first image processing task, we want to take the TEST.bmp and reflect it. In other words, instead of having decrease from black to white down the diagonal, we want it to decrease from white to black. If the pixel value was black, we want it to be white.

reflect(k, k) = 255 – TEST(k, k)

Aside from this simple algorithm, reflect.bmp and TEST.bmp are identical, so we would like to copy the header information and the color table from TEST.bmp to reflect.bmp. This can be done with the following 2 procedures. A procedure is a subprogram that does not return a value.

```
/*-------------COPIES HEADER AND INFO HEADER----------------*/
void copyImageInfo(FILE* inputFile, FILE* outputFile)
{
  unsigned char         *ptrC;
  unsigned char         dummy;
```

```
  int                   i;

  dummy = '0';
  ptrC = &dummy;

  fseek(inputFile, 0L, SEEK_SET);
  fseek(outputFile, 0L, SEEK_SET);

  for(i=0; i<=50; i++)
  {
    fread(ptrC, sizeof(char), 1, inputFile);
    fwrite(ptrC, sizeof(char), 1, outputFile);
  }

}
```

```
/*----------------COPIES COLOR TABLE-----------------------------*/
void copyColorTable(FILE* inputFile, FILE* outputFile, int nColors)
{
  unsigned char         *ptrC;
  unsigned char         dummy;
  int                   i;

  dummy = '0';
  ptrC = &dummy;

  fseek(inputFile, 54L, SEEK_SET);
  fseek(outputFile, 54L, SEEK_SET);

  for(i=0; i<=(4*nColors); i++)  /* there are (4*nColors) bytesin color table */
  {
    fread(ptrC, sizeof(char), 1, inputFile);
    fwrite(ptrC, sizeof(char), 1, outputFile);
  }

}
```

The following code reflects a BMP file. The differences from raster.c are in red:

To be compiled with Turbo C

Note: download reflect.zip rather than cutting and pasting from below.

```
#include (stdio.h)
#include (stdlib.h)
#include (math.h)

/*-------STRUCTURES---------*/
typedef struct {int rows; int cols; unsigned char* data;} sImage;

/*-------PROTOTYPES---------*/
long getImageInfo(FILE*, long, int);
void copyImageInfo(FILE* inputFile, FILE* outputFile);
void copyColorTable(FILE* inputFile, FILE* outputFile, int nColors);


int main(int argc, char* argv[])
{
```

```
  FILE                  *bmpInput, *bmpOutput;
  sImage                originalImage;
  unsigned char         someChar;
  unsigned char*        pChar;
  int                   nColors;  /* BMP number of colors */
  long                  fileSize; /* BMP file size */
  int                   vectorSize; /* BMP vector size */
  int                   r, c;       /* r = rows, c = cols */


  /* initialize pointer */
  someChar = '0';
  pChar = &someChar;

  if(argc < 2)
  {
    printf("Usage: %s bmpInput.bmp\n", argv[0]);
    exit(0);
  }
  printf("Reading filename %s\n", argv[1]);

  /*--------READ INPUT FILE------------*/
  bmpInput = fopen(argv[1], "rb");
  fseek(bmpInput, 0L, SEEK_END);

  /*--------DECLARE OUTPUT FILE--------*/
  bmpOutput = fopen("reflect.bmp", "wb");

  /*--------GET BMP DATA---------------*/
  originalImage.cols = (int)getImageInfo(bmpInput, 18, 4);
  originalImage.rows = (int)getImageInfo(bmpInput, 22, 4);
  fileSize = getImageInfo(bmpInput, 2, 4);
  nColors = getImageInfo(bmpInput, 46, 4);
  vectorSize = fileSize - (14 + 40 + 4*nColors);

  /*-------PRINT DATA TO SCREEN-------------*/
  printf("Width: %d\n", originalImage.cols);
  printf("Height: %d\n", originalImage.rows);
  printf("File size: %ld\n", fileSize);
  printf("# Colors: %d\n", nColors);
  printf("Vector size: %d\n", vectorSize);

  copyImageInfo(bmpInput, bmpOutput);
  copyColorTable(bmpInput, bmpOutput, nColors);

  /*----START AT BEGINNING OF RASTER DATA-----*/
  fseek(bmpInput, (54 + 4*nColors), SEEK_SET);

  /*----------READ RASTER DATA----------*/
  for(r=0; r<=originalImage.rows - 1; r++)
  {
    for(c=0; c<=originalImage.cols - 1; c++)
    {
      /*-----read data, reflect and write to output file----*/
      fread(pChar, sizeof(char), 1, bmpInput);
      if(*pChar == 255) *pChar = 255;
      else *pChar = 255 - *pChar;
      fwrite(pChar, sizeof(char), 1, bmpOutput);
    }
  }

  fclose(bmpInput);
  fclose(bmpOutput);

}

/*----------GET IMAGE INFO SUBPROGRAM--------------*/
long getImageInfo(FILE* inputFile, long offset, int numberOfChars)
{
```

```
  unsigned char                 *ptrC;
  long                          value = 0L;
  unsigned char                 dummy;
  int                           i;

  dummy = '0';
  ptrC = &dummy;

  fseek(inputFile, offset, SEEK_SET);

  for(i=1; i<=numberOfChars; i++)
  {
    fread(ptrC, sizeof(char), 1, inputFile);
    /* calculate value based on adding bytes */
    value = (long)(value + (*ptrC)*(pow(256, (i-1))));
  }
  return(value);

} /* end of getImageInfo */

/*-------------COPIES HEADER AND INFO HEADER----------------*/
void copyImageInfo(FILE* inputFile, FILE* outputFile)
{
  unsigned char         *ptrC;
  unsigned char         dummy;
  int                   i;

  dummy = '0';
  ptrC = &dummy;

  fseek(inputFile, 0L, SEEK_SET);
  fseek(outputFile, 0L, SEEK_SET);

  for(i=0; i<=50; i++)
  {
    fread(ptrC, sizeof(char), 1, inputFile);
    fwrite(ptrC, sizeof(char), 1, outputFile);
  }

}

/*----------------COPIES COLOR TABLE-----------------------------*/
void copyColorTable(FILE* inputFile, FILE* outputFile, int nColors)
{
  unsigned char         *ptrC;
  unsigned char         dummy;
  int                   i;

  dummy = '0';
  ptrC = &dummy;

  fseek(inputFile, 54L, SEEK_SET);
  fseek(outputFile, 54L, SEEK_SET);

  for(i=0; i<=(4*nColors); i++)  /* there are (4*nColors) bytesin color table */
  {
    fread(ptrC, sizeof(char), 1, inputFile);
    fwrite(ptrC, sizeof(char), 1, outputFile);
  }

}
```

You are visitor number: