34 Fire

“Love never fails.”

Beginners of mode 13h programming will always try to do *fire program*. It is of course an easy program. In order to set palette registers, we must know what are all the colors used by ‘Fire’. After setting palette registers and loading the screen values, we can generate a “firing” screen with certain logic.

34.1 Extracting Palette

We can manually find out the colors used by “Fire” (image). But it is quite tedious. Instead, we can extract palette information from a BMP file that has the 'fire' image.

34.1.1 PAL Utility

The following code fragment extracts palette information from a known BMP file (**Fire.bmp**) and saves in another file (**Fire.pal**). This palette (**Fire.pal**) file can then be included in our main-fire program.

Let’s call the following program as PAL utility!

```
/*-----------------------------------------------------------
      PAL - utility to extract palette from a BMP file
 *---
 */

#include <stdio.h>

#define BM_TYPE 19778u

typedef unsigned int WORD;
typedef unsigned long DWORD;
typedef unsigned char BYTE;

typedef struct tagBITMAPFILEHEADER
{
    WORD    bfType;
    DWORD   bfSize;
    WORD    bfReserved1;
    WORD    bfReserved2;
    DWORD   bfOffBits;
} BITMAPFILEHEADER;
```

```
typedef struct tagBITMAPINFOHEADER
{
    DWORD  biSize;
    DWORD  biWidth;
    DWORD  biHeight;
    WORD   biPlanes;
    WORD   biBitCount;
    DWORD  biCompression;
    DWORD  biSizeImage;
    DWORD  biXPelsPerMeter;
    DWORD  biYPelsPerMeter;
    DWORD  biClrUsed;
    DWORD  biClrImportant;
} BITMAPINFOHEADER;

typedef struct tagRGBQUAD {
    BYTE    rgbBlue;
    BYTE    rgbGreen;
    BYTE    rgbRed;
    BYTE    rgbReserved;
} RGBQUAD;

int main( int argc, char *argv[] )
{
   BITMAPFILEHEADER fheader,  *header = &fheader;
   BITMAPINFOHEADER finfo,  *info = &finfo;
   RGBQUAD trgb, *rgb = &trgb;
   FILE *bfp, *pfp;
   short num_col;
   int i;

   if ( argc < 3 )
      {
       printf( "Usage: PAL file.bmp palfile\n\a" );
       exit( 1 );
      }
   bfp = fopen( argv[1], "rb" );
   pfp = fopen( argv[2], "w" );
   if ( bfp==NULL || pfp==NULL )
      {
        printf( "File Error!\n\a" );
        exit( 1 );
      }
   fprintf( pfp, "/*  Palette file created with PAL  */\n"
                 "/*  File name: %s  */\n"
                 "BYTE pal[768] = { ", argv[2]
        );
   fread( header, sizeof( BITMAPFILEHEADER ), 1, bfp );
   fread( info, sizeof( BITMAPINFOHEADER ), 1, bfp );
   if ( header->bfType != BM_TYPE )
        printf( "%s is not a bmp!\n\a", argv[1]);
     else
       {
        /*  We only handle 256 color types with this code!  */
        if ( info->biPlanes == 1 && info->biBitCount == 8 )
           {
            num_col = info->biClrUsed ? info->biClrUsed : 256;
            for ( i=0; i <  num_col-1; ++i )
             {
                fread( rgb, sizeof( RGBQUAD ), 1, bfp );
                if ( i%4 == 0 )
                  fprintf( pfp, "\n\t\t %d, %d, %d,", rgb->rgbRed>>2,
                        rgb->rgbGreen>>2, rgb->rgbBlue>>2 );
                  else
                    fprintf( pfp, "\t%d, %d, %d,",  rgb->rgbRed>>2,
                        rgb->rgbGreen>>2, rgb->rgbBlue>>2 );
             }
            fread( rgb, sizeof( RGBQUAD ), 1, bfp );
            fprintf( pfp, "\t%d, %d, %d\n\t};\n", rgb->rgbRed>>2,
                        rgb->rgbGreen>>2, rgb->rgbBlue>>2 );
            fprintf( pfp, "/*_______EOF %s _________________________*/",
                        argv[2] );
            }
          else
             printf("This code only does 256 color BMP's\n");
       }
   fcloseall( );
   return(0);
} /*--main( )-------*/
```

34.1.2 Using PAL

In order to extract palette information (i.e., colors used by 'Fire'), run the above program as:

```
C:\WAR>PAL Fire.bmp Fire.pal
```

I've got the following palette file from the known **Fire.bmp** file:

```
/*  Palette file created with PAL  */
/*  File name: fire.pal  */
BYTE pal[768] = {
             0, 0, 0,   0, 0, 6,    0, 0, 6,    0, 0, 7,
             0, 0, 8,   0, 0, 8,    0, 0, 9,    0, 0, 10,
2, 0, 10,  4, 0, 9,    6, 0, 9,    8, 0, 8,
10, 0, 7,  12, 0, 7,   14, 0, 6,   16, 0, 5,
18, 0, 5,  20, 0, 4,   22, 0, 4,   24, 0, 3,
26, 0, 2,  28, 0, 2,   30, 0, 1,   32, 0, 0,
32, 0, 0,  33, 0, 0,   34, 0, 0,   35, 0, 0,
36, 0, 0,  36, 0, 0,   37, 0, 0,   38, 0, 0,
39, 0, 0,  40, 0, 0,   40, 0, 0,   41, 0, 0,
42, 0, 0,  43, 0, 0,   44, 0, 0,   45, 0, 0,
46, 1, 0,  47, 1, 0,   48, 2, 0,   49, 2, 0,
50, 3, 0,  51, 3, 0,   52, 4, 0,   53, 4, 0,
54, 5, 0,  55, 5, 0,   56, 6, 0,   57, 6, 0,
58, 7, 0,  59, 7, 0,   60, 8, 0,   61, 8, 0,
63, 9, 0,  63, 9, 0,   63, 10, 0,  63, 10, 0,
63, 11, 0, 63, 11, 0,  63, 12, 0,  63, 12, 0,
63, 13, 0, 63, 13, 0,  63, 14, 0,  63, 14, 0,
63, 15, 0, 63, 15, 0,  63, 16, 0,  63, 16, 0,
63, 17, 0, 63, 17, 0,  63, 18, 0,  63, 18, 0,
63, 19, 0, 63, 19, 0,  63, 20, 0,  63, 20, 0,
63, 21, 0, 63, 21, 0,  63, 22, 0,  63, 22, 0,
63, 23, 0, 63, 24, 0,  63, 24, 0,  63, 25, 0,
63, 25, 0, 63, 26, 0,  63, 26, 0,  63, 27, 0,
63, 27, 0, 63, 28, 0,  63, 28, 0,  63, 29, 0,
63, 29, 0, 63, 30, 0,  63, 30, 0,  63, 31, 0,
63, 31, 0, 63, 32, 0,  63, 32, 0,  63, 33, 0,
63, 33, 0, 63, 34, 0,  63, 34, 0,  63, 35, 0,
63, 35, 0, 63, 36, 0,  63, 36, 0,  63, 37, 0,
63, 38, 0, 63, 38, 0,  63, 39, 0,  63, 39, 0,
63, 40, 0, 63, 40, 0,  63, 41, 0,  63, 41, 0,
63, 42, 0, 63, 42, 0,  63, 43, 0,  63, 43, 0,
63, 44, 0, 63, 44, 0,  63, 45, 0,  63, 45, 0,
63, 46, 0, 63, 46, 0,  63, 47, 0,  63, 47, 0,
63, 48, 0, 63, 48, 0,  63, 49, 0,  63, 49, 0,
63, 50, 0, 63, 50, 0,  63, 51, 0,  63, 52, 0,
63, 52, 0, 63, 52, 0,  63, 52, 0,  63, 52, 0,
63, 53, 0, 63, 53, 0,  63, 53, 0,  63, 53, 0,
63, 54, 0, 63, 54, 0,  63, 54, 0,  63, 54, 0,
63, 54, 0, 63, 55, 0,  63, 55, 0,  63, 55, 0,
63, 55, 0, 63, 56, 0,  63, 56, 0,  63, 56, 0,
63, 56, 0, 63, 57, 0,  63, 57, 0,  63, 57, 0,
63, 57, 0, 63, 57, 0,  63, 58, 0,  63, 58, 0,
63, 58, 0, 63, 58, 0,  63, 59, 0,  63, 59, 0,
63, 59, 0, 63, 59, 0,  63, 60, 0,  63, 60, 0,
63, 61, 0, 63, 61, 0,  63, 61, 0,  63, 62, 0,
63, 62, 0, 63, 62, 0,  63, 62, 0,  63, 63, 0,
63, 63, 1, 63, 63, 2,  63, 63, 3,  63, 63, 4,
63, 63, 5, 63, 63, 6,  63, 63, 7,  63, 63, 8,
63, 63, 9, 63, 63, 10, 63, 63, 10, 63, 63, 11,
                63, 63, 12,63, 63, 13, 63, 63, 14, 63, 63, 15,
                63, 63, 16,63, 63, 17, 63, 63, 18, 63, 63, 19,
                63, 63, 20,63, 63, 21, 63, 63, 21, 63, 63, 22,
                63, 63, 23,63, 63, 24, 63, 63, 25, 63, 63, 26,
                63, 63, 27,63, 63, 28, 63, 63, 29, 63, 63, 30,
                63, 63, 31,63, 63, 31, 63, 63, 32, 63, 63, 33,
                63, 63, 34,63, 63, 35, 63, 63, 36, 63, 63, 37,
                63, 63, 38,63, 63, 39, 63, 63, 40, 63, 63, 41,
                63, 63, 42,63, 63, 42, 63, 63, 43, 63, 63, 44,
                63, 63, 45,63, 63, 46, 63, 63, 47, 63, 63, 48,
                63, 63, 49,63, 63, 50, 63, 63, 51, 63, 63, 52,
                63, 63, 52,63, 63, 53, 63, 63, 54, 63, 63, 55,
                63, 63, 56,63, 63, 57, 63, 63, 58, 63, 63, 59,
                63, 63, 60,63, 63, 61, 63, 63, 62, 63, 63, 63,
                63, 63, 60,63, 63, 61, 63, 63, 62, 63, 63, 63
         };
/*_______EOF fire.pal _________________________*/
```

34.2 Fire Program

This program is actually a clone of **Fire!.asm**, a Turbo Assembler program written by **Adam Hyde**. Now, let's look into the logic of our fire program!

We have already created the palette file with our PAL utility. Thus we have avoided programming complexity. We need that palette file (**Fire.pal**) only at compile time. After creating EXE file, we no more require that palette file!

Like any other mode 13h programs, first of all, we have to set up the palette registers with corresponding color values. For that, we have used functions `InitializeMCGA( )` and `SetUpPalette( )`. We use off-screen buffer called `Buffer`. This `Buffer` holds all pixel values. The size of the `Buffer` is 320x104. For 'fire' effect, we have to alter the pixel values present on the `Buffer`. And we must copy our `Buffer` to the Video RAM repeatedly. We copy a single row of the `Buffer` to two rows of Video RAM. You may find that our `Buffer` is 320x104 and not 320x100. The reason is that we don't need to alter the last 4 rows for 'fire' effect.

We have two important functions namely `Random( )` and `AveragePixels( )`. First we create two bottom lines with random pixel values. Since we have only 256 colors, the random values should be between 0 and 255. Using `AveragePixels( )` function, we alter the pixel values of `Buffer`. Then we copy our `Buffer` to Video RAM. We have to repeat this process until a key is pressed. If a key is pressed, we switch back to Text mode using `TextMode( )` function.

```
#include <dos.h>

#define BufferX         (320L)   /* Width of screen buffer */
#define BufferY         (104L)   /* Height of screen buffer */
```

```
#define BufferLen       (33280u)  /*      320*104           */
#pragma inline

typedef unsigned int WORD;
typedef unsigned char BYTE;

BYTE  Buffer[BufferLen];      /*  The screen buffer */
WORD Seed  =  0x3749;         /*  The seed value  */

#include "fire.pal"                   /* palette, generated with PAL */

BYTE far *Video = MK_FP( 0xa000, 0u );

void InitializeMCGA( void )
{
   asm {
         MOV   AH, 00H        /*  Set video mode */
         MOV   AL, 13H        /*  Mode 13h      */
         INT   10H            /*  We are now in 320x200x256 */
       }
} /*--InitializeMCGA( )------*/

void SetUpPalette( void )
{
   asm {
        .386
        MOV   SI, OFFSET pal  /*  SI now points to the palette  */
        MOV   CX, 768         /*  Prepare for 768 OUTs */
        MOV   DX, 03C8H       /*  Palette WRITE register */
        XOR   AL, AL          /*  Start at color 0 */
        CLI                   /*  Disable interrupts */
        OUT   DX, AL          /*  Send value */
        CLD                   /*  Forward direction */
        INC   DX              /*  Now use palette DATA register */
        REP   OUTSB           /*  768 multiple OUTs */
        STI                   /*  Enable interrupts */
       }
} /*--SetUpPalette( )--------*/

BYTE Random( void )
{
   asm {
         MOV   AX, Seed       /*  Move the seed value into AX */
         MOV   DX, 8405H      /*  Move 8405H into DX */
         MUL   DX             /*  Put 8405H x Seed into DX:AX */
         INC   AX             /*  Increment AX */
         MOV   Seed, AX       /*  We have a new seed */
```

```
      }
   return( _DL );
} /*--Random( )---------*/

void AveragePixels( void )
{
   long i;
   for ( i = 320; i < BufferX*BufferY-BufferX ; ++i )
     {
      Buffer[i-BufferX] = ( Buffer[i] + Buffer[i+1] + Buffer[i-1] +
                            Buffer[i+BufferX] ) / 4;
      if ( Buffer[i-BufferX]!=0 )
                 Buffer[i-BufferX] -= 1;
     }
} /*--AveragePixels(  )-------*/

void TextMode( void )
{
   asm {
         MOV   AH, 00H        /* Set video mode */
         MOV   AL, 03H        /* Mode 03h */
         INT   10H            /* Enter 80x25x16 mode */
      }
} /*--TextMode( )----------*/

int main( void )
{
   unsigned long i, j, k;

   InitializeMCGA( );
   SetUpPalette( );
   while( !kbhit( ) )
    {
      AveragePixels( );
      for ( i = BufferX*BufferY - 2*BufferX; i < BufferX*BufferY; ++i )
            Buffer[i] = Random( );
      for( i=k=0; k<BufferY-4; ++k, i+=320 )
         for( j=0 ; j<320; ++i, ++j  )
           {
            Video[i] = Buffer[320*k+j];
            Video[i+320] = Buffer[320*k+j];
           }
    }
   TextMode( );

   return(0);
} /*--main( )---------*/
```

Exercises

1. Replace the values of palette buffer `pal[768]` found at the palette file (`Fire.pal`) with some random values. Now, execute the program. Observe the effect.
2. Write a program that generates ‘whirlpool’ or ‘lake’ effect.
3. Write a program that simulates ‘waving Indian Tricolor flag’.

Suggested Projects

1. Write a DOS based screen saver. (Hint: Use TSR concepts!)