Windows BMP 图象的颜色匹配及显示

梁学军　　(北京师范大学)

BMP 图象文件最早应用于 Microsoft Windows 窗口系统,如今已得到了广泛的应用。但是,目前在 DOS 环境下开发的 BMP 图象的显示程序往往都要根据 BMP 图象文件中提供的颜色表来重新设置系统的调色板。这样,当它应用于某特定的 DOS 图形界面之后,该图象界面的其他元素的色彩也跟着被改变了。解决这个问题,可以采用系统当前调色板的颜色来匹配 BMP 图象中的色彩。然而,颜色匹配的一般算法比较复杂。从实用的角度出发,笔者找出了图形方式下系统缺省调色板的颜色与 Windows BMP 图象中色彩的匹配关系。从而,实现了在不改变系统缺省调色板设置的情况下,显示 Windows BMP 图象。本文给出的源程序在 Borlard C++3。1 中调试通过,并已应用在我单位的教学软件开发工具之中。

为方便起见,我们只讨论 16 色的未压缩的 BMP 图象文件(有关 BMP 图象文件格式的详细内容请参考其他书刊)。并且,由于叙述上的原因,假设你的微机使用的是 VGA 显示卡。其实,本文的源程序也适用于 EGA 显示卡,只需将源程序中的 480 改为 350。

首先,通过对 Windows 的画笔(Paintbrush)绘图工具软件的分析,笔者发现:凡是经过 Windows 的画笔软件存盘之后得到的 16 色 BMP 图象文件都有相同的颜色表。而且,该图象文件中位图数据部分的象素值与该颜色表所提供的 16 种颜色的 RGB 值之间的对应关系如表 1 所示。举例来说,若某点象素值为 4,则该象素点在图象中应该是兰色。

另一方面,当初始 VGA 显示卡为 640×480 分辨率的 16 色图形方式时,无论使用 BIOS 中断方法,还是使用 Borland C++中的 Initgraph 函数调用,系统缺省调色板的索引号与 VGA 显示卡的 DAC 寄存器中的 RGB 值的对应关系总是如表 2 所示。

我们知道,BMP 图象文件的位图数据中每个象素值所代表的颜色的最终实现决定于以该象素值为调色板索引号所对应的 VGA 显示卡的 DAC 寄存器中的 RGB 值。从而,在不改变表 2 所示的对应关系的情况下,要想使每个象素保持或接近原有的颜色,就要对每个象素的值进行适当的匹配变换。例如:Windows BMP 图象文件中的某个象素值是 4,代表兰色,就应该把 4 变换为 1。由此,我们得到了一个象素值的匹配变换表。在源程序中,我们将这个变换表表示为 16 个整数组成的数组 Win2 BIOS。此外,源程序中的变量 P 指向显示缓冲区的首地址,函数 farptr 用于进行指针的运算。程序中采用的是 VGA 或 EGA 的写方式。逐点对象素进行显示。显示出的图象位于屏幕的左上角。

表 1:象素值与颜色表的 RGB 值的对应关系

象素值	R	G	B
0	0	0	0
1	32	0	0
2	0	32	0
3	32	32	0
4	0	0	32
5	32	0	32
6	0	32	32
7	32	32	32
8	48	48	48
9	63	0	0
10	0	63	0
11	63	63	0
12	0	0	63
13	63	0	63
14	0	63	63
15	63	63	63

表 2:调色板索引号与 DAL 寄存器的 RGB 值的对应关系

索引号	R	G	B
0	0	0	0
1	0	0	42
2	0	42	0
3	0	42	42
4	42	0	0
5	42	0	42
6	42	21	0
7	42	42	42
8	21	21	21
9	21	21	63
10	21	63	21
11	21	63	63
12	63	21	21
13	63	21	63
14	63	63	21
15	63	63	63

这个变换表从表 1、表 2 可以看出:

表 1 中的颜色　　0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

对应表 2 中的近似颜色　0 4 2 6 1 5 3 8 7 12 10 14 9 13 11 15

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <graphics.h>

#ifndef MK_FP
#define MK_FP(seg,ofs)  ((void far *)\
                         (((unsigned long (seg) << 16) | (unsigned)(ofs)))
#endif

typedef struct {
        int bfType;
        long bfSize;
        int bfReserved1;
        int bfReserved2;
        long bfoffBits;
} BITMAPFILEHEADER;
typedef struct {
        long biSize;
        long biWidth;
        long biHeight;
        int biPlanes;
        int biBitCount;
        long biCompress;
        long biSizeImage;
        long biXPelsPerMeter;
        long biYPelsPerMeter;
        long biClrUsed;
        long biClrImprotant;
}BITMAPINFOHEADER;

char far * farptr(char far *, long);
void init();

main(int argc, char * * argv)
{
        FILE *fp;
        BITMAPFILEHEADER FileHeader;
        BITMAPINFOHEADER InfoHeader;
        unsigned int width, depth, bytes, linebytes;
        int Win2Bios[16]={0,4,2,6,1,5,3,8,7,12,10,14,9,13,11,15};
        char far * p = (char far *)0xa0000000;
        char far * ptr;
        int  color;
        register int d, w;

        if(argc==1)
        {
                printf("Usage: %s BmpFileName\n",(argv[0]));
                return(1);
        }

        if((fp=fopen(argv[1],"rb"))==NULL)
        {
                printf("[%s] open error.\n",argv[1]);
                return(1);
        }

        if(fread((char *)&FileHeader,1,sizeof(BITMAPFILEHEADER),fp)
                        !=sizeof(BITMAPFILEHEADER))
        {

                printf("[%s] read error. \n",argv[1]);
                return(1);
        }

                if(fread((char *)&InfoHeader,1,sizeof(BITMAPINFOHEADER),fp)
                        !=sizeof(BITMAPINFOHEADER))
        {
                printf("[%s] read error. \n",argv[1]);
                return(1);
        }

        if(InfoHeader.biCompress != 0)
        {
                printf("Only support non_compressed BMP file. \n");
                return(1);
        }

        if(InfoHeader.biBitCount != 4)
        {
                printf("[%s] is not a 16 color's BMP file.\n",argv[1]);
                return(1);
        }

        width = (unsigned)InfoHeader.biWidth;
        depth = (unsigned)InfoHeader.biHeight;
        depth = (depth>480)?480:depth;
```

```
        bytes = (width+7)/8,
        linebytes = bytes*4;

        init();
        fseek(fp,118,SEEK_SET);

        outp(0x3ce,5);
        outp(0x3cf,0);
        outp(0x3ce,1);
        outp(0x3cf,0xff);

        for(d=0;d<depth;d++)
        {
                ptr = farptr(p,(long)(depth-d-1)*80);
                for(w=0;w<linebytes;w++)
                {
                        if(w<(width>>1))
                        {
                                color = (int)(fgetc(fp));
                                outp(0x3ce,0);
                                outp(0x3cf,Win2Bios[(color & 0xf0)>>4]);
                                outp(0x3ce,8);
                                outp(0x3cf,0X80 >> ((w<<1)&7));
                                *ptr = *ptr | 0x00;

                                outp(0x3ce,0);
                                outp(0x3cf,Win2Bios[(color & 0x0f)]);
                                outp(0x3ce,8);
                                outp(0x3cf,0X80 >> (((w<<1)+1)&7));

                                *ptr = *ptr | 0x00;

                                if((w&3)==3)
                                        ptr = farptr(ptr,1);
                        }
                        else if ((w==(width>>1)) && (width & 1 ))

                        {
                                color = (int)(fgetc(fp));
                                outp(0x3ce,0);
                                outp(0x3cf,Win2Bios[(color & 0xf0)>>4] );
                                outp(0x3ce,8);
                                outp(0x3cf,0X80 >> ((w<<1)&7));
                                *ptr = *ptr | 0x00;
                        }
                        else
                                color = (int)fgetc(fp);
                }
        }

        outp(0x3ce,1);
        outp(0x3cf,0);
        outp(0x3ce,0);
        outp(0x3cf,0);
        outp(0x3ce,8);
        outp(0x3cf,0xff);

        getch();
        fclose(fp);
        closegraph();
        return (0);
}

char far *farptr(char far *p, long l)
{
        unsigned int seg, off;

        seg = FP_SEG(p);
        off = FP_OFF(p);
        seg +=(off/16);
        off &= 0x000f;
        off += (unsigned int)(l & 0x000fL);
        seg += (l / 16L);
        p = (char far *)MK_FP(seg,off);
        return(p);
}

void init()
{
        int gdriver=DETECT, gmode, error_code;
        /* initialize graphics mode */
        initgraph(&gdriver,&gmode,"c:\\borlandc\\bgi");
        error_code=graphresult();
        if( error_code != 0 )
        {
                printf("Graphics error : %s\n", grapherrormsg(error_code));
                printf("Press any key to halt:");
                getch();
                exit(1);
        }
}
```