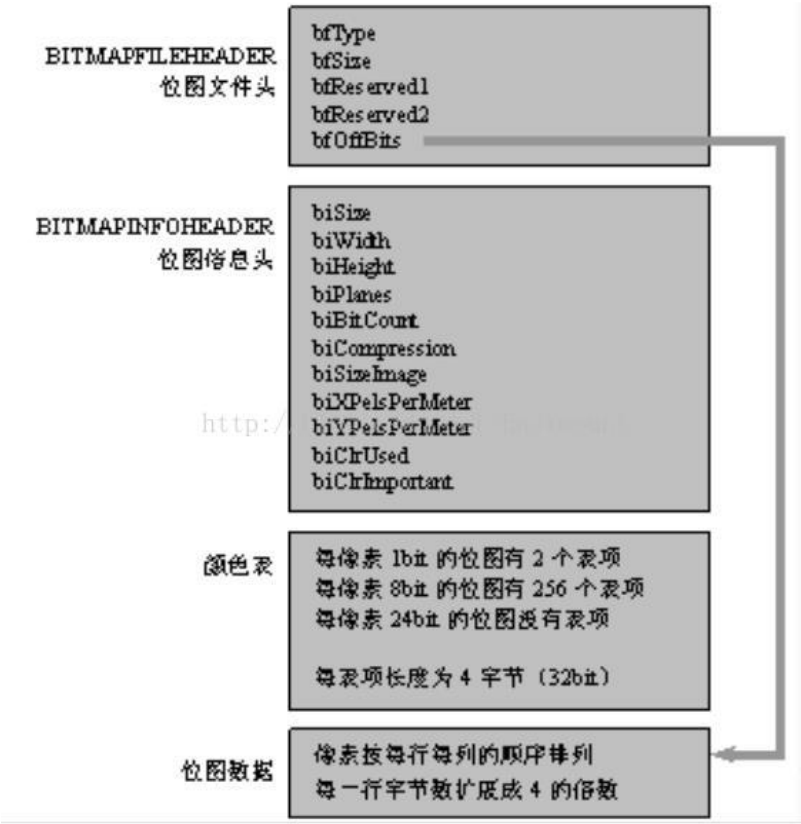
【数字图像处理】一.MFC详解显示BMP格式图片

本文主要是讲述《数字图像处理》系列栏目中的第一篇文章.主要详细介绍了BMP图片格式,同时使用C++和MFC显示BMP格式,主要结合自己的《数字图像处理》课程和以前的项目叙述讲解.

一.BMP图片格式定义

BMP文件格式是Windows操作系统推荐和支持的标准图像文件格式,是一种将内存或显示器的图像数据不经过压缩而直接按位存盘的文件格式,故称位图(bitmap),其扩展名为BMP.BMP图像通常有4个部分组成:位图文件头、位图信息头、颜色表、位图数据.如下图所示:



第一部分为位图文件头BITMAPFILEHEADER.位图文件头结构长度固定为14个字节,包含文件的类型、大小、位图文件保留字、位图数据距文件头的偏移量.其中WORD为无符号16位整数(2byte),DWORD为无符号32位整数(4byte).具体结构体定义如下:

```
//位图文件头
typedef struct tagBITMAPFILEHEADER {
        WORD        bfType;           //位图文件的类型，必须为BM 0x424d 表示.bmp
        DWORD       bfSize;           //位图文件的大小，以字节为单位 包括该14字节
        WORD        bfReserved1;      //位图文件保留字，必须为0
        WORD        bfReserved2;      //位图文件保留字，必须为0
    DWORD      bfOffBits;         //位图数据距文件头的偏移量，以字节为单位 即前三部分和
} BITMAPFILEHEADER;
```

第二部分为位图信息头BITMAPINFOHEADER,该结构也固定为40个字节,用于说明位图的尺寸、宽高、像素、分辨率、颜色表等信息.具体结构定义如下:

```
//位图信息头
typedef struct tagBITMAPINFOHEADER {
        DWORD      biSize;             //本结构所占用字节数 40字节
        LONG       biWidth;            //位图的宽度，以像素为单位
        LONG       biHeight;           //位图的高度，以像素为单位
        WORD       biPlanes;           //目标设备的级别，必须为1
        WORD       biBitCount;         //每个像素所需的位数，必须是1（双色）、
                                       //4（16色）、8（256色）或24（真彩色）之一
        DWORD      biCompression;      //位图压缩类型，必须是 0（BI_RGB不压缩）、
                                       //1（BI_RLE8压缩类型）或2（BI_RLE压缩类型）之一
        DWORD      biSizeImage;        //位图的大小，以字节为单位
        LONG       biXPelsPerMeter;    //位图水平分辨率，每米像素数
        LONG       biYPelsPerMeter;    //位图垂直分辨率，每米像素数
        DWORD      biClrUsed;          //位图实际使用的颜色表中的颜色数
        DWORD      biClrImportant;     //位图显示过程中重要的颜色数
} BITMAPINFOHEADER;
```

第三部分为颜色表或调色板(Palette).有些位图需要调色板,有些位图如真彩色图(biBitCount=24)不需要调色板,它们的BITMAPINFOHEADER后面直接是位图数据.调色板实际是一个数组,共有biClrUsed个元素(如果该值为零,则有2的biBitCount次幂个元素).数组中每个元素的类型是一个RGBQUAD结构,占4字节.定义如下:

```
//位图颜色表
typedef struct tagRGBQUAD
{
    BYTE       rgbBlue;               //蓝色的亮度（值范围为0～255）
    BYTE       rgbGreen;                  //绿色的亮度（值范围为0～255）
    BYTE       rgbRed;                    //红色的亮度（值范围为0～255）
    BYTE       rgbReserved;           //保留，必须为0
} RGBQUAD;
```

第四部分就是实际的图像数据.对于真彩色图(24位位图 biBitCount=24),图像数据就是实际的RGB值;对于用到调色板的位图,图像数据就是该像素颜色在调色板中的索引值.下面对2色、16色、256色和真彩色位图分别介绍:
(1).2色位图:当biBitCount=1时,用1位就可以表示该像素的颜色(0表示黑,1表示白),所以8个像素占1个字节;
(2).16色位图:当biBitCount=4时,用4为可以表示一个像素的颜色,所以2个像素占1个字节;
(3).256色位图:当biBitCount=8时,用1个字节表示1个像素,1个像素占1个字节;
(4).真彩色图:当biBitCount=24时,此时用3个字节表示1个像素,其中RGB各占1字节,由于没有颜色表,位图信息头后面是位图数据.

同时,注意以下几点:
1.由于Windows规定一个扫描所占的字节数必须是4的倍数(即以long为单位),不足的以0填充.同时注意下面公式,计算只含位图数据的大小:biSizeImage=(((bi.biWidth*bi.biBitCount)+31)/(32*4))*bi.Height
在后面讲述获取文件的信息时会通过UE软件结合16进制数据进行详细讲解上面各个数据的具体含义.
2.BMP图片格式的数据是从下到上、从左到右读.即文件中最先读到的图像是最下面一行的左边第一个元素,即从左下角开始存储(0,0)点,从左下角到右上角存储数据.尤其是在图像几何变换平移、旋转时,我就犯过这样的错误,本想让图像从左下角向右上移动,结果刚好相反,后面也会通过实例加深大家的印象.
3.如果想使用C语言\C++显示图片,建议自定义个ImageStruct.h的头文件.包含BMP位图的位图文件头结构、位图信息头结构、位图颜色表3个结构,在实例变量操作.而使用MFC,因为在wingdi.h文件中系统已经定义了BMP图像的结构BITMAPFILEHEADER、BITMAPINFOHEADER,直接在View.h中用他俩实例定义即可.

二.显示BMP图片的基本步骤

在MFC工程XXXView.h类中添加成员函数void ShowBitmap(CDC* pDC,CString BmpName);通过自定义函数实现显示BMP格式图像,其中*pDC是CDC句柄,BmpName是图像文件名.具体步骤如下:

1.创建位图并调用函数LoadImage装载图标、光标或位图.

HBITMAP m_hBitmap;

m_hBitmap=(HBITMAP)LoadImage(HINSTANCE hinst,LPCTSTR lpszName,UINT uType,int cxDesired,int cyDesired,UINT fuLoad)

2.定义并创建一个内存设备环境DC,调用函数CreateCompatibleDC创建兼容的DC.

CDC dcBmp; dcBmp.CreateCompatibleDC(pDC) ;

3.定义BITMAP变量,调用函数GetBitmap将图片载入位图中,该定义是为后去图像的长宽等信息.

BITMAP m_bmp; m_bitmap.GetBitmap(&m_bmp);

4.调用函数SelectObject将位图选入兼容内存设备环境DC中.

dcBmp.SelectObject(&m_bitmap);

5.将兼容的DC中的位图填到当前DC中,调用函数BitBlt或strechBlt显示图像.

(1).BitBlt()该函数对指定的源设备环境区域中的像素进行位块(bit_block)转换,以传送到目标设备环境.

pDC->BitBlt(0,0,m_bmp.bmWidth,m_bmp.bmHeight,&dcBmp,0,0,SRCCOPY);

(2).stretchBlt()该函数从源矩形中复制位图到目标矩形,必要是按目标设备设置的模式进行图像拉伸或压缩.

pDC->StretchBlt(0,0,m_nDrawWidth,m_nDrawHeight,&dcBmp,0,0,m_bmp.bmWidth,m_bmp.bmHeight,SRCCOPY);

6.恢复临时DC的位图,删除CreateCompatibleDC得到的图片DC,删除内存中的位图及释放系统资源.

dcBmp.SelectObject(pbmpOld); DeleteObject(&m_bitmap); dcBmp.DeleteDC();

具体函数代码如下:

```
//****************显示BMP格式图片****************//
void CShowBMPView::ShowBitmap(CDC *pDC, CString BmpName)
{
        //定义bitmap指针 调用函数LoadImage装载位图
        HBITMAP m_hBitmap;
        m_hBitmap = (HBITMAP)
LoadImage(NULL,BmpName,IMAGE_BITMAP,0,0,LR_LOADFROMFILE|LR_DEFAULTSIZE|LR_CREATEDIBSECTION);
                        /*********************************************************************/
        /* 1.要装载OEM图像，则设此参数值为0  OBM_ OEM位图 OIC_OEM图标 OCR_OEM光标
        /* 2.BmpName要装载图片的文件名
        /* 3.装载图像类型:
        /*   IMAGE_BITMAP-装载位图 IMAGE_CURSOR-装载光标 IMAGE_ICON-装载图标
        /* 4.指定图标或光标的像素宽度和长度 以像素为单位
        /* 5.加载选项:
        /*   IR_LOADFROMFILE-指明由lpszName指定文件中加载图像
        /*   IR_DEFAULTSIZE-指明使用图像默认大小
        /*   LR_CREATEDIBSECTION-当uType参数为IMAGE_BITMAP时,创建一个DIB项
        /*********************************************************************/

        if( m_bitmap.m_hObject )
        {
                m_bitmap.Detach();           //切断CWnd和窗口联系
        }
        m_bitmap.Attach(m_hBitmap);      //将句柄HBITMAP m_hBitmap与CBitmap m_bitmap关联

        //边界
        CRect rect;
        GetClientRect(&rect);

        //图片显示(x,y)起始坐标
        int m_showX=0;
        int m_showY=0;
        int m_nWindowWidth = rect.right - rect.left;   //计算客户区宽度
        int m_nWindowHeight = rect.bottom - rect.top;  //计算客户区高度

        //定义并创建一个内存设备环境DC
        CDC dcBmp;
        if( !dcBmp.CreateCompatibleDC(pDC) )   //创建兼容性的DC
```

```
        return;
    BITMAP m_bmp;                           //临时bmp图片变量
    m_bitmap.GetBitmap(&m_bmp);             //将图片载入位图中

    CBitmap *pbmpOld = NULL;
    dcBmp.SelectObject(&m_bitmap);          //将位图选入临时内存设备环境

    //图片显示调用函数stretchBlt
    pDC->StretchBlt(0,0,m_bmp.bmWidth,m_bmp.bmHeight,&dcBmp,0,0,m_bmp.bmWidth,m_bmp.bmHeight,SRCCOPY);

    /*************************************************************************/
    /* BOOL StretchBlt(int x,int y,int nWidth,int nHeight,CDC* pSrcDC,
    /*                 int xSrc,int ySrc,int nSrcWidth,int nSrcHeight,DWORD dwRop );
    /* 1.参数x、y位图目标矩形左上角x、y的坐标值
    /* 2.nWidth、nHeigth位图目标矩形的逻辑宽度和高度
    /* 3.pSrcDC表示源设备CDC指针
    /* 4.xSrc、ySrc表示位图源矩形的左上角的x、y逻辑坐标值
    /* 5.dwRop表示显示位图的光栅操作方式 SRCCOPY用于直接将位图复制到目标环境中
    /*************************************************************************/

    dcBmp.SelectObject(pbmpOld);            //恢复临时DC的位图
    DeleteObject(&m_bitmap);                //删除内存中的位图
    dcBmp.DeleteDC();                       //删除CreateCompatibleDC得到的图片DC
}
```

补充:MFC中DC指device context,设备环境或设备描述表,它其实是GDI内部保存数据的一种数据结构.此结构中属性内容与特定输出设备相关,属性定义了GDI函数的工作细节.总之,使用GDI绘图函数,就需要一个DC句柄,MFC中把和DC相关的封装成类.其中CDC是一个抽象基类,可以访问整个显示器和打印机.其中下面链接中的文章详细描述了CBitmap、HBitmap、Bitmap三者的区别与联系.http://blog.csdn.net/ivan_ljf/article/details/8569130

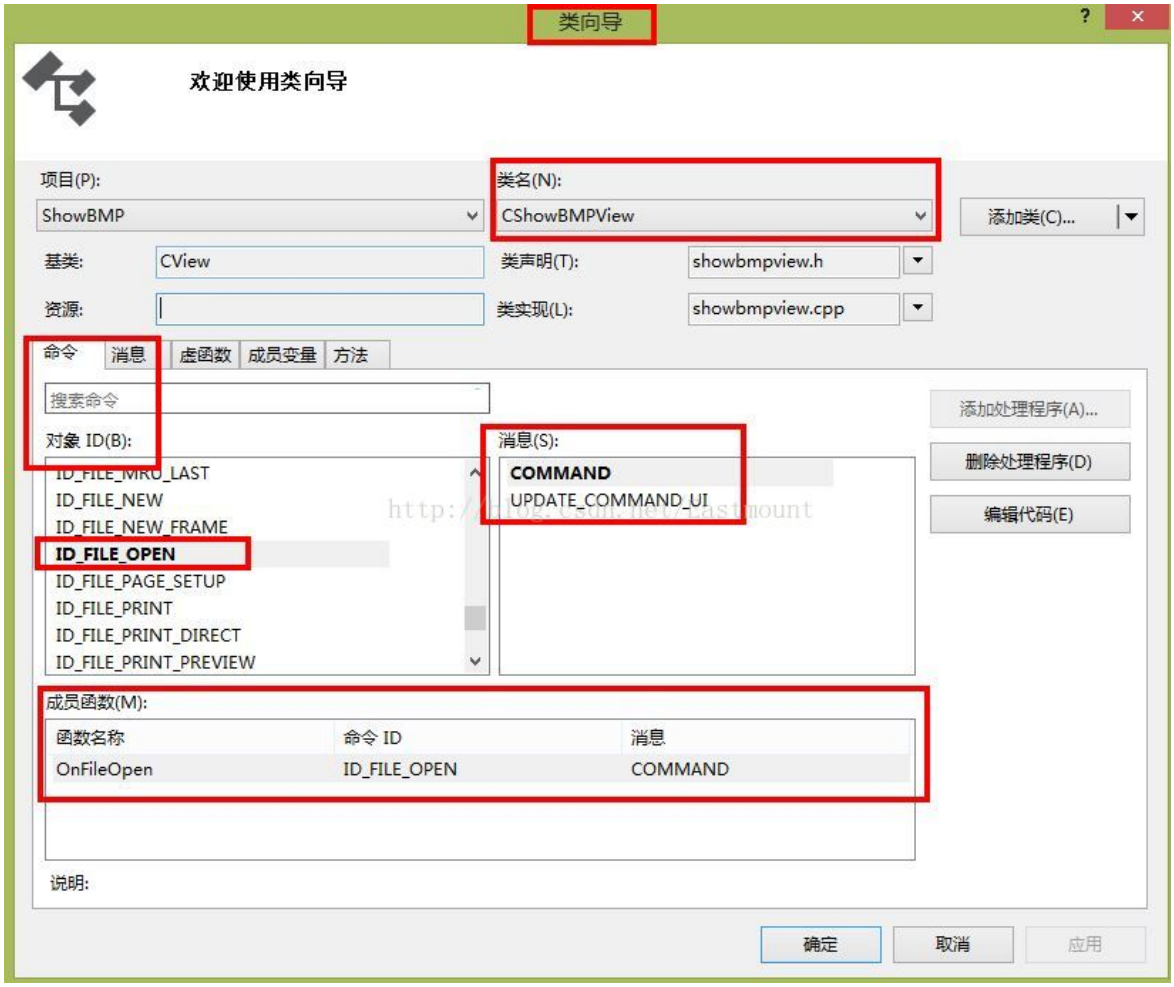
三.MFC显示BMP图片

下面将详细讲解使用VS2012 MFC创建工程的具体步骤:
第一步:新建项目"MFC应用程序",项目名为ShowBMP,在应用程序类型中选择"单个文档",点击"确定".在右栏的"资源视图"中,点击"Menu->IDR_MAINFRAM"可以查看并修改菜单视图.
第二步:向CShowBMPView类添加成员变量和成员函数.在右栏的"类视图"右键CShowBMPView添加函数或直接在ShowBMPView.h中直接添加public成员变量和成员函数.添加代码如下:

```
public:
    //成员变量
    CString BmpName;                            //保存图像文件文件名
    CString EntName;                            //保存图像文件扩展名
    CBitmap m_bitmap;                           //创建位图对象

    //成员函数
    void ShowBitmap(CDC* pDC,CString BmpName);    //用来显示指定位图bmp的函数
```

第三步:设置打开BMP图片函数."项目"->"类向导"->选择"类名"CShowBMPView->在命令对象ID中双击"ID_FILE_OPEN"->自动生成默认成员函数OnFileOpen,消息为COMMAND.双击成员函数(Member Functions)进入函数编辑.(VC++ 6.0中Ctrl+W可以实现建立类向导)



向添加成员函数CShowBMPView::OnFileOpen()添加如下代码,主要是生成打开图片的对话框,并获取图片路径及后缀.自定义四种格式为bmp gif jpg tiff,但目前只能打开bmp格式图片.

```
//**************文件打开****************//
void CShowBMPView::OnFileOpen()
{
	//四种格式的文件：bmp gif jpg tiff
	CString filter;
	filter="所有文件(*.bmp,*.jpg,*.gif,*tiff)|*.bmp;*.jpg;*.gif;*.tiff| BMP(*.bmp)|*.bmp|
JPG(*.jpg)|*.jpg| GIF(*.gif)|*.gif| TIFF(*.tiff)|*.tiff||";
	CFileDialog dlg(TRUE,NULL,NULL,OFN_HIDEREADONLY,filter,NULL);
	//按下确定按钮 dlg.DoModal() 函数显示对话框
	if( dlg.DoModal() == IDOK )
	{
		BmpName = dlg.GetPathName();	//获取文件路径名	如D:\pic\abc.bmp
		EntName = dlg.GetFileExt();	//获取文件扩展名
		EntName.MakeLower();		//将文件扩展名转换为一个小写字符
		Invalidate();			//调用该函数就会调用OnDraw重绘画图
	}
}
```

第四步:在ShowBMPView.cpp中编写void CShowBMPView::ShowBitmap(CDC *pDC, CString BmpName)函数,即“二.显示BMP图片基本步骤”.同时通过OnDraw()函数调用ShowBitmap()函数显示图片.代码如下:

```
//在OnDraw函数中调用ShowBitmap()实现图片的显示功能
void CShowBMPView::OnDraw(CDC* pDC)
```

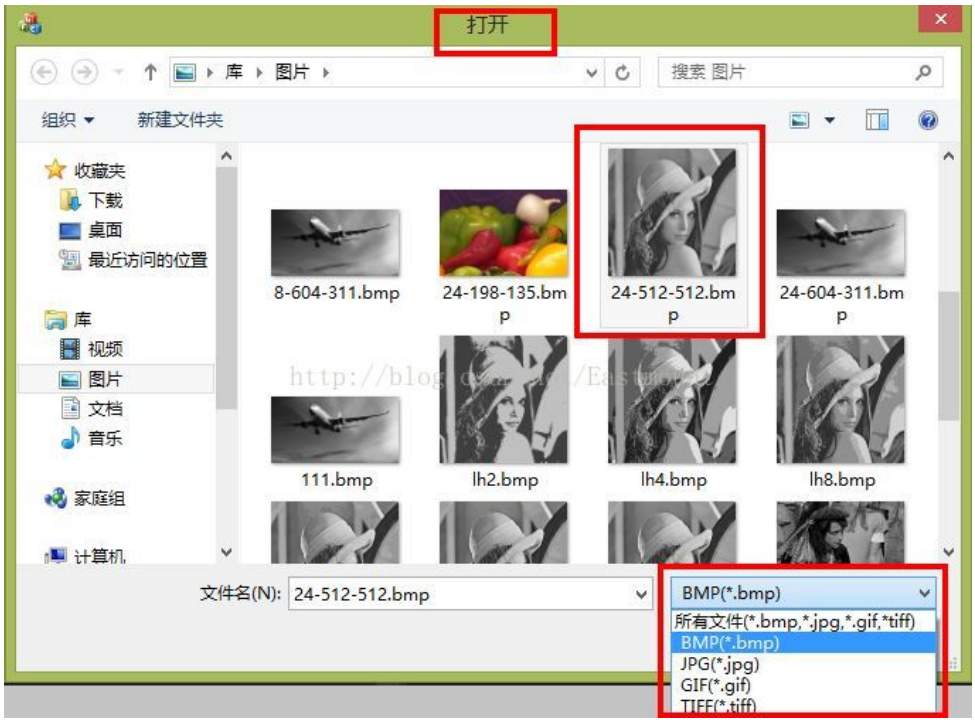
```
{
        CShowBMPDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        if (!pDoc)
                return;

        // TODO: 在此处为本机数据添加绘制代码

        if( EntName.Compare(_T("bmp")) == 0 )       //bmp格式
        {
                ShowBitmap(pDC,BmpName);                //显示图片
        }
}
```

四.运行结果及总结

运行程序后,显示如下所示:其中可以看到自定义的打开对话框和显示图片.



最后,该文章主要是数字图像处理的基础知识,详细介绍了BMP图片格式和使用MFC如何读取BMP图片的相关知识.仅以此篇纪念自己的考研结束,新的开始.同时推荐大家阅读一位叫烟雨江南的作者的文章,个人感觉帮组很大.http://blog.csdn.net/xiajun07061225/article/details/6633938 同时该项目免费下载网址:

http://download.csdn.net/detail/eastmount/6848841
希望该文章能够对大家有所帮助,同时如果文章中有错误或不足之处,还请大家海涵.
(By:Eastmount 2014-1-14 夜8点http://blog.csdn.net/eastmount)