# ImageCenterDib类的使用及开发

ImageCenterDib类来自于谢凤英《Visual C++ 数字图像处理》，为使以后方便快捷对使用该类，对其使用方法及开发总结如下。

## 一、类结构

**公有成员：**

图像数据指针：unsigned char \* m\_pImgData，指向位图起始位置

图像颜色表指针：LPRGBQUAD m\_lpColorTabe

每像素占的位数：int m\_nBitCount，灰度图像为8位，24位彩色图像为24位

**保护成员：**

int m\_imgWidth：图像的宽，像素为单位

int m\_imgHeight：图像的高，像素为单位

LPBITMAPINFOHEADER m\_lpBmpInfoHead：图像信息头指针

由于私有成员在其派生类中不可访问，因而不用过多在意，了解详细信息，可参考原代码。

**公有方法：**

ImgCenterDib()：不带参数的构造函数

ImgCenterDib(CSize size, int nBitCount, LPRGBQUAD lpColorTable,

unsigned char \*pImgData)：带参数的构造函数，在派生类的构造函数中调用

~ImgCenterDib()：析构函数

CSize GetDimensions()：获取DIB的尺寸（宽高）

**BOOL Read(LPCTSTR lpszPathName)：DIB读函数，常用，用于读取位图**

**BOOL Write(LPCTSTR lpszPathName)：DIB写函数，常用，用于保存位图**

**BOOL Draw(CDC\* pDC, CPoint origin, CSize size)：显示DIB，常用，用于显示位图**

void ReplaceDib(CSize size, int nBitCount, LPRGBQUAD lpColorTable,

unsigned char \*pImgData)：用新的数据替换DIB，用于更新DIB图像数据

int ComputeColorTabalLength(int nBitCount)：计算颜色表的长度

由于私用方法在其派生类中不可访问，因而在开发时也不用过多在意，要了解详细信息，可参考原代码。

## 二、开发

### 2.1 基本成员

对ImageCenterDib类进行开发，实际上就是在ImageCenterDib类的基础上增加一些处理函数，对DIB数据进行处理，并输出处理的结果，因此可以对派生类定义以下基本公有成员，用于输出处理结果：

int m\_nBitCountOut; //输出图像每像素位数

unsigned char \* m\_pImgDataOut; //输出图像位图数据指针

LPRGBQUAD m\_lpColorTableOut; //输出图像颜色表

这三个成员，也是ImageCenterDib构造函数所需要的三个成员，因此在修改位图数据后，可由这三个成员生成一个新的ImageCenterDib对象。

为便于在派生类中使用，定义如下私有成员：

int m\_imgWidthOut; //输出图像的宽，像素为单位

int m\_imgHeightOut; //输出图像的高，像素为单位

int m\_nColorTableLengthOut; //输出图像颜色表长度

公有成员：

一般为构造函数，析构函数，图像处理函数，为使外界能够得到处理后位图的尺寸，可以定义GetDimensions函数。

### 2.2 基本框架

以为例，给出基本派生类框架。

#### 2.2.1头文件：

#ifndef \_INSIDE\_VISUAL\_CPP\_EXAMPLEPROCESS

#define \_INSIDE\_VISUAL\_ CPP\_EXAMPLEPROCESS

#include "ImageCenterDib.h"

class ExampleProcess:public ImgCenterDib

{

**//基本成员及方法**

public:

//输出图像每像素位数

int m\_nBitCountOut;

//输出图像位图数据指针

unsigned char \* m\_pImgDataOut;

//输出图像颜色表

LPRGBQUAD m\_lpColorTableOut;

private:

//输出图像的宽，像素为单位

int m\_imgWidthOut;

//输出图像的高，像素为单位

int m\_imgHeightOut;

//输出图像颜色表长度

int m\_nColorTableLengthOut;

public:

//不带参数的构造函数

ExampleProcess();

//带参数的构造函数

ExampleProcess(CSize size, int nBitCount, LPRGBQUAD lpColorTable,

unsigned char \*pImgData);

//析构函数

~ExampleProcess();

//以像素为单位返回输出图像的宽和高

CSize GetDimensions();

**//图像处理函数**

Private:

**//其它成员函数**

}

#endif // \_INSIDE\_VISUAL\_ CPP\_EXAMPLEPROCESS

#### 2.2.2源文件：

#include "stdafx.h"

#include "ExampleProcess.h"

#include "math.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* ExampleProcess()

\*

\*说明：无参数的构造函数，对成员变量进行初始化

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ExampleProcess::ExampleProcess()

{

m\_pImgDataOut=NULL;//输出图像位图数据指针为空

m\_lpColorTableOut=NULL;//输出图像颜色表指针为空

m\_nColorTableLengthOut=0;//输出图像颜色表长度为0

m\_nBitCountOut=0;//输出图像每像素位数为0

m\_imgWidthOut=0;//输出图像的宽为0

m\_imgHeightOut=0;//输出图像的高为0

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* ExampleProcess()

\*

\*函数参数：

\* CSize size -图像大小（宽、高）

\* int nBitCount -每像素比特数

\* LPRGBQUAD lpColorTable -颜色表指针

\* unsigned char \*pImgData -位图数据指针

\*

\*返回值：

\* 无

\*

\*说明：本函数为带参数的构造函数，给定位图的大小、每像素位数、颜色表

\* 及位图数据，调用ImgCenterDib()对基类成员初始化，作为输入图像数据

\* 相关的数据成员,并初始化输出图像相关的数据成员

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ExampleProcess::ExampleProcess(CSize size, int nBitCount, LPRGBQUAD lpColorTable,

unsigned char \*pImgData):

ImgCenterDib(size, nBitCount, lpColorTable, pImgData)

{

m\_pImgDataOut=NULL;//输出图像位图数据指针为空

m\_lpColorTableOut=NULL;//输出图像颜色表指针为空

m\_nColorTableLengthOut=0;//输出图像颜色表长度为0

m\_nBitCountOut=0;//输出图像每像素位数为0

m\_imgWidthOut=0;//输出图像的宽为0

m\_imgHeightOut=0;//输出图像的高为0

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* ~ExampleProcess()

\*

\*说明：析构函数，释放资源

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ExampleProcess::~ExampleProcess()

{

//释放输出图像位图数据缓冲区

if(m\_pImgDataOut!=NULL){

delete []m\_pImgDataOut;

m\_pImgDataOut=NULL;

}

//释放输出图像颜色表

if(m\_lpColorTableOut!=NULL){

delete []m\_lpColorTableOut;

m\_lpColorTableOut=NULL;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* GetDimensions()

\*

\*函数参数：

\* 无

\*

\*返回值：

\* 图像的尺寸，用CSize类型表达

\*

\*说明：返回输出图像的宽和高

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CSize ExampleProcess::GetDimensions()

{

if(m\_pImgDataOut == NULL) return CSize(0, 0);

return CSize(m\_imgWidthOut, m\_imgHeightOut);

}

**//图像处理函数及其它函数**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* Process ()

\*说明：

\* 图像处理函数框架。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ExampleProcess::Process()

{

unsigned char\* pDIB;

int bytecount;

//释放旧的输出图像缓冲区

if(m\_pImgDataOut!=NULL){

delete []m\_pImgDataOut;

m\_pImgDataOut=NULL;

}

if(m\_lpColorTableOut!=NULL){

delete []m\_lpColorTableOut;

m\_lpColorTableOut=NULL;

}

if(m\_pImgDataOut!=NULL)

{

delete []m\_pImgDataOut;

m\_pImgDataOut=NULL;

}

pDIB=m\_pImgData;

if(m\_nBitCount==8||m\_nBitCount==24)

{

bytecount = m\_imgWidth \* m\_imgHeight \*m\_nBitCount/8; //获取位图数据区的字节数

}

else

{

AfxMessageBox("只能处理真彩色和8位灰度图像!");

return ;

}

//输出图像每像素位数与输入图像相同

m\_nBitCountOut=m\_nBitCount;

//计算颜色表长度

m\_nColorTableLengthOut=ComputeColorTabalLength(m\_nBitCountOut);

//若有颜色表，输入图像的颜色表与输出图像颜色表相同

if(m\_nColorTableLengthOut!=0){

m\_lpColorTableOut=new RGBQUAD[m\_nColorTableLengthOut];

memcpy(m\_lpColorTableOut,m\_lpColorTable,

sizeof(RGBQUAD)\*m\_nColorTableLengthOut);

}

//输出图像的宽高,与输入图像相等

m\_imgWidthOut=m\_imgWidth;

m\_imgHeightOut=m\_imgHeight;

//输出图像每行像素所占的字节数

int lineByteOut=(m\_imgWidth\*m\_nBitCountOut/8+3)/4\*4;

//为输出的位图数据分配内存

m\_pImgDataOut=new unsigned char[lineByteOut\*m\_imgHeight];

//图像处理，结果存放在上面分配的内存空间中

int lineByteOut=(m\_imgWidth\*m\_nBitCountOut/8+3)/4\*4;

……

}

## 三、ImageCenterDib源码

### 3.1 ImageCenterDib类H文件

#ifndef \_INSIDE\_VISUAL\_CPP\_IMGCENTERDIB

#define \_INSIDE\_VISUAL\_CPP\_IMGCENTERDIB

//ImgCenterDib类,image processing center,北航图像中心编写的DIB类

//ImgCenterDib类

class ImgCenterDib

{

public:

//图像数据指针

unsigned char \* m\_pImgData;

//图像颜色表指针

LPRGBQUAD m\_lpColorTable;

//每像素占的位数

int m\_nBitCount;

private:

//指向DIB的指针（包含BITMAPFILEHEADER，BITMAPINFOHEADER和颜色表）

LPBYTE m\_lpDib;

//逻辑调色板句柄

HPALETTE m\_hPalette;

//颜色表长度（多少个表项）

int m\_nColorTableLength;

protected:

//图像的宽，像素为单位

int m\_imgWidth;

//图像的高，像素为单位

int m\_imgHeight;

//图像信息头指针

LPBITMAPINFOHEADER m\_lpBmpInfoHead;

public:

//不带参数的构造函数

ImgCenterDib();

//带参数的构造函数

ImgCenterDib(CSize size, int nBitCount, LPRGBQUAD lpColorTable,

unsigned char \*pImgData);

//析构函数

~ImgCenterDib();

//获取DIB的尺寸（宽高）

CSize GetDimensions();

//DIB读函数

BOOL Read(LPCTSTR lpszPathName);

//DIB写函数

BOOL Write(LPCTSTR lpszPathName);

//显示DIB

BOOL Draw(CDC\* pDC, CPoint origin, CSize size);

//用新的数据替换DIB

void ReplaceDib(CSize size, int nBitCount, LPRGBQUAD lpColorTable,

unsigned char \*pImgData);

//计算颜色表的长度

int ComputeColorTabalLength(int nBitCount);

private:

//创建逻辑调色板

void MakePalette();

//清理空间

void Empty();

};

#endif // \_INSIDE\_VISUAL\_CPP\_IMGCENTERDIB

### 3.2 ImageCenterDib类CPP文件

#include "stdafx.h"

#include "ImageCenterDib.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* ImgCenterDib()

\*

\*说明：无参数的构造函数，对成员变量进行初始化

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ImgCenterDib::ImgCenterDib()

{

m\_lpDib=NULL;//初始化m\_lpDib为空。

m\_lpColorTable=NULL;//颜色表指针为空

m\_pImgData=NULL; // 图像数据指针为空

m\_lpBmpInfoHead=NULL; // 图像信息头指针为空

m\_hPalette = NULL;//调色板为空

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* ImgCenterDib()

\*

\*函数参数：

\* CSize size -图像大小（宽、高）

\* int nBitCount -每像素比特数

\* LPRGBQUAD lpColorTable -颜色表指针

\* unsigned char \*pImgData -位图数据指针

\*

\*返回值：

\* 无

\*

\*说明：本函数为带参数的构造函数，给定位图的大小、每像素位数、颜色表

\* 及位图数据生成一个ImgCenterDib类对象

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ImgCenterDib::ImgCenterDib(CSize size, int nBitCount, LPRGBQUAD lpColorTable,

unsigned char \*pImgData)

{

//如果没有位图数据传入，我们认为是空的DIB，此时不分配DIB内存

if(pImgData==NULL){

m\_lpDib=NULL;

m\_lpColorTable=NULL;

m\_pImgData=NULL; // 图像数据

m\_lpBmpInfoHead=NULL; // 图像信息头

m\_hPalette = NULL;

}

else{//如果有位图数据传入

//图像的宽、高、每像素位数等成员变量赋值

m\_imgWidth=size.cx;

m\_imgHeight=size.cy;

m\_nBitCount=nBitCount;

//根据每像素位数，计算颜色表长度

m\_nColorTableLength=ComputeColorTabalLength(nBitCount);

//每行像素所占字节数，必须扩展成4的倍数

int lineByte=(m\_imgWidth\*nBitCount/8+3)/4\*4;

//位图数据缓冲区的大小（图像大小）

int imgBufSize=m\_imgHeight\*lineByte;

//为m\_lpDib一次性分配内存，生成DIB结构

m\_lpDib=new BYTE [sizeof(BITMAPINFOHEADER) +

sizeof(RGBQUAD) \* m\_nColorTableLength+imgBufSize];

//填写BITMAPINFOHEADER结构

m\_lpBmpInfoHead = (LPBITMAPINFOHEADER) m\_lpDib;

m\_lpBmpInfoHead->biSize = sizeof(BITMAPINFOHEADER);

m\_lpBmpInfoHead->biWidth = m\_imgWidth;

m\_lpBmpInfoHead->biHeight = m\_imgHeight;

m\_lpBmpInfoHead->biPlanes = 1;

m\_lpBmpInfoHead->biBitCount = m\_nBitCount;

m\_lpBmpInfoHead->biCompression = BI\_RGB;

m\_lpBmpInfoHead->biSizeImage = 0;

m\_lpBmpInfoHead->biXPelsPerMeter = 0;

m\_lpBmpInfoHead->biYPelsPerMeter = 0;

m\_lpBmpInfoHead->biClrUsed = m\_nColorTableLength;

m\_lpBmpInfoHead->biClrImportant = m\_nColorTableLength;

//调色板句柄初始化为空，有颜色表时，MakePalette()函数要生成新的调色板

m\_hPalette = NULL;

//如果有颜色表，则将颜色表拷贝进DIB的颜色表位置

if(m\_nColorTableLength!=0){

//m\_lpColorTable指向DIB颜色表的起始位置

m\_lpColorTable=(LPRGBQUAD)(m\_lpDib+sizeof(BITMAPINFOHEADER));

//颜色表拷贝

memcpy(m\_lpColorTable,lpColorTable,sizeof(RGBQUAD) \* m\_nColorTableLength);

//创建逻辑调色板

MakePalette();

}

//m\_pImgData指向DIB位图数据起始位置

m\_pImgData = (LPBYTE)m\_lpDib+sizeof(BITMAPINFOHEADER)+

sizeof(RGBQUAD) \* m\_nColorTableLength;

//拷贝图像数据进DIB位图数据区

memcpy(m\_pImgData,pImgData,imgBufSize);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* ~ImgCenterDib()

\*

\*说明：析构函数，释放资源

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ImgCenterDib::~ImgCenterDib()

{

//释放m\_lpDib所指向的内存缓冲区

if(m\_lpDib != NULL)

delete [] m\_lpDib;

//如果有调色板，释放调色板缓冲区

if(m\_hPalette != NULL)

::DeleteObject(m\_hPalette);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* Draw()

\*

\*函数参数：

\* CDC\* pDC -设备环境指针

\* CPoint origin -显示矩形区域的左上角

\* CSize size -显示矩形区域的尺寸

\*

\*返回值：

\* 0为失败,1为成功

\*

\*说明：给定设备环境指针，以及需要显示的矩形区域在设备环境中的位置

\* 将m\_lpDib所指向的DIB显示出来

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BOOL ImgCenterDib::Draw(CDC\* pDC, CPoint origin, CSize size)

{

//旧的调色板句柄

HPALETTE hOldPal=NULL;

//如果DIB为空，则返回0

if(m\_lpDib == NULL) return FALSE;

//如果DIB有调色板

if(m\_hPalette != NULL) {

//将调色板选进设备环境中

hOldPal=::SelectPalette(pDC->GetSafeHdc(), m\_hPalette, TRUE);

//实现调色板

pDC->RealizePalette();

}

//设置位图伸缩模式

pDC->SetStretchBltMode(COLORONCOLOR);

//将DIB在pDC所指向的设备上进行显示

::StretchDIBits(pDC->GetSafeHdc(), origin.x, origin.y, size.cx, size.cy,

0, 0, m\_lpBmpInfoHead->biWidth, m\_lpBmpInfoHead->biHeight,m\_pImgData,

(LPBITMAPINFO) m\_lpBmpInfoHead, DIB\_RGB\_COLORS, SRCCOPY);

//恢复旧的调色板

if(hOldPal!=NULL)

::SelectPalette(pDC->GetSafeHdc(), hOldPal, TRUE);

//函数返回

return TRUE;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* Read()

\*

\*函数参数：

\* LPCTSTR lpszPathName -图像名字及路径

\*

\*返回值：

\* 0为失败,1为成功

\*

\*说明：给定一个图像文件名及其路径，读取图像数据进内存生成DIB，

\* 并存放在m\_lpDib所指向的缓冲区

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BOOL ImgCenterDib::Read(LPCTSTR lpszPathName)

{

//读模式打开图像文件

CFile file;

if (!file.Open(lpszPathName, CFile::modeRead | CFile::shareDenyWrite))

return FALSE;

BITMAPFILEHEADER bmfh;

try {

//清理空间

Empty();

//读取BITMAPFILEHEADER结构到变量bmfh中

int nCount=file.Read((LPVOID) &bmfh, sizeof(BITMAPFILEHEADER));

//异常判断

if(nCount != sizeof(BITMAPFILEHEADER)) {

throw new CException;

}

if(bmfh.bfType != 0x4d42) {

throw new CException;

}

//为m\_lpDib分配空间，读取DIB进内存

if(m\_lpDib!=NULL)

delete []m\_lpDib;

m\_lpDib=new BYTE[file.GetLength() -sizeof(BITMAPFILEHEADER)];

file.Read(m\_lpDib, file.GetLength() -sizeof(BITMAPFILEHEADER));

//m\_lpBmpInfoHead位置为m\_lpDib起始位置

m\_lpBmpInfoHead = (LPBITMAPINFOHEADER)m\_lpDib;

//为成员变量赋值

m\_imgWidth=m\_lpBmpInfoHead->biWidth;

m\_imgHeight=m\_lpBmpInfoHead->biHeight;

m\_nBitCount=m\_lpBmpInfoHead->biBitCount;

//计算颜色表长度

m\_nColorTableLength=

ComputeColorTabalLength(m\_lpBmpInfoHead->biBitCount);

//如果有颜色表,则创建逻辑调色板

m\_hPalette = NULL;

if(m\_nColorTableLength!=0){

m\_lpColorTable=(LPRGBQUAD)(m\_lpDib+sizeof(BITMAPINFOHEADER));

MakePalette();

}

//m\_pImgData指向DIB的位图数据起始位置

m\_pImgData = (LPBYTE)m\_lpDib+sizeof(BITMAPINFOHEADER) +

sizeof(RGBQUAD) \* m\_nColorTableLength;

}

catch(CException\* pe) {

AfxMessageBox("Read error");

pe->Delete();

return FALSE;

}

//函数返回

return TRUE;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* Write()

\*

\*函数参数：

\* LPCTSTR lpszPathName -指定图像名字及路径

\*

\*返回值：

\* 0为失败,1为成功

\*

\*说明：给定一个图像文件名及其路径，将m\_lpDib指向的DIB存盘

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BOOL ImgCenterDib::Write(LPCTSTR lpszPathName)

{

//写模式打开文件

CFile file;

if (!file.Open(lpszPathName, CFile::modeCreate | CFile::modeReadWrite

| CFile::shareExclusive))

return FALSE;

//填写文件头结构

BITMAPFILEHEADER bmfh;

bmfh.bfType = 0x4d42; // 'BM'

bmfh.bfSize = 0;

bmfh.bfReserved1 = bmfh.bfReserved2 = 0;

bmfh.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) +

sizeof(RGBQUAD) \* m\_nColorTableLength;

try {

//文件头结构写进文件

file.Write((LPVOID) &bmfh, sizeof(BITMAPFILEHEADER));

//文件信息头结构写进文件

file.Write(m\_lpBmpInfoHead, sizeof(BITMAPINFOHEADER));

//如果有颜色表的话，颜色表写进文件

if(m\_nColorTableLength!=0)

file.Write(m\_lpColorTable, sizeof(RGBQUAD) \* m\_nColorTableLength);

//位图数据写进文件

int imgBufSize=(m\_imgWidth\*m\_nBitCount/8+3)/4\*4\*m\_imgHeight;

file.Write(m\_pImgData, imgBufSize);

}

catch(CException\* pe) {

pe->Delete();

AfxMessageBox("write error");

return FALSE;

}

//函数返回

return TRUE;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* ReplaceDib()

\*

\*函数参数：

\* CSize size -要替换的新图像尺寸

\* int nBitCount -每像素的位数

\* LPRGBQUAD lpColorTable -颜色表指针

\* unsigned char \*pImgData -位图数据的指针

\*返回值：

\* 无

\*

\*说明：图像的尺寸，每像素位数、颜色表、及位图数据，

\* 替换m\_lpDib所指向的现有的DIB

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ImgCenterDib::ReplaceDib(CSize size, int nBitCount,

LPRGBQUAD lpColorTable,unsigned char \*pImgData)

{

//释放原DIB所占空间

Empty();

//成员变量赋值

m\_imgWidth=size.cx;

m\_imgHeight=size.cy;

m\_nBitCount=nBitCount;

//计算颜色表的长度

m\_nColorTableLength=ComputeColorTabalLength(nBitCount);

//每行像素所占字节数，扩展成4的倍数

int lineByte=(m\_imgWidth\*nBitCount/8+3)/4\*4;

//位图数据的大小

int imgBufSize=m\_imgHeight\*lineByte;

//为m\_lpDib重新分配空间，以存放新的DIB

m\_lpDib=new BYTE [sizeof(BITMAPINFOHEADER) +

sizeof(RGBQUAD) \* m\_nColorTableLength+imgBufSize];

//填写位图信息头BITMAPINFOHEADER结构

m\_lpBmpInfoHead = (LPBITMAPINFOHEADER) m\_lpDib;

m\_lpBmpInfoHead->biSize = sizeof(BITMAPINFOHEADER);

m\_lpBmpInfoHead->biWidth = m\_imgWidth;

m\_lpBmpInfoHead->biHeight = m\_imgHeight;

m\_lpBmpInfoHead->biPlanes = 1;

m\_lpBmpInfoHead->biBitCount = m\_nBitCount;

m\_lpBmpInfoHead->biCompression = BI\_RGB;

m\_lpBmpInfoHead->biSizeImage = 0;

m\_lpBmpInfoHead->biXPelsPerMeter = 0;

m\_lpBmpInfoHead->biYPelsPerMeter = 0;

m\_lpBmpInfoHead->biClrUsed = m\_nColorTableLength;

m\_lpBmpInfoHead->biClrImportant = m\_nColorTableLength;

//调色板置空

m\_hPalette = NULL;

//如果有颜色表，则将颜色表拷贝至新生成的DIB，并创建逻辑调色板

if(m\_nColorTableLength!=0){

m\_lpColorTable=(LPRGBQUAD)(m\_lpDib+sizeof(BITMAPINFOHEADER));

memcpy(m\_lpColorTable,lpColorTable,sizeof(RGBQUAD) \* m\_nColorTableLength);

MakePalette();

}

//m\_pImgData指向DIB的位图数据起始位置

m\_pImgData = (LPBYTE)m\_lpDib+sizeof(BITMAPINFOHEADER)+

sizeof(RGBQUAD) \* m\_nColorTableLength;

//将新位图数据拷贝至新的DIB中

memcpy(m\_pImgData,pImgData,imgBufSize);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* ComputeColorTabalLength()

\*

\*函数参数：

\* int nBitCount -DIB每像素位数

\*

\*返回值：

\* 颜色表的长度

\*

\*说明：给定每像素的比特数，计算出DIB颜色表的长度，颜色表长度

\* 为2的nBitCount次幂，但是24位位图颜色表长度为0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int ImgCenterDib::ComputeColorTabalLength(int nBitCount)

{

int colorTableLength;

switch(nBitCount) {

case 1:

colorTableLength = 2;

break;

case 4:

colorTableLength = 16;

break;

case 8:

colorTableLength = 256;

break;

case 16:

case 24:

case 32:

colorTableLength = 0;

break;

default:

ASSERT(FALSE);

}

ASSERT((colorTableLength >= 0) && (colorTableLength <= 256));

return colorTableLength;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* MakePalette()

\*

\*函数参数：

\* 无

\*

\*返回值：

\* 无

\*

\*说明：根据DIB的颜色表，生成一个逻辑调色板（m\_hPalette），

\* 该调色板在显示位图时将被选进设备环境中,24位彩色无调色板

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ImgCenterDib::MakePalette()

{

//如果颜色表长度为0，则不创建逻辑调色板

if(m\_nColorTableLength == 0)

return;

//删除旧的逻辑调色板句柄

if(m\_hPalette != NULL) ::DeleteObject(m\_hPalette);

//申请空间，根据颜色表生成LOGPALETTE结构

LPLOGPALETTE pLogPal = (LPLOGPALETTE) new char[2 \* sizeof(WORD) +

m\_nColorTableLength \* sizeof(PALETTEENTRY)];

pLogPal->palVersion = 0x300;

pLogPal->palNumEntries = m\_nColorTableLength;

LPRGBQUAD m\_lpDibQuad = (LPRGBQUAD) m\_lpColorTable;

for(int i = 0; i < m\_nColorTableLength; i++) {

pLogPal->palPalEntry[i].peRed = m\_lpDibQuad->rgbRed;

pLogPal->palPalEntry[i].peGreen = m\_lpDibQuad->rgbGreen;

pLogPal->palPalEntry[i].peBlue = m\_lpDibQuad->rgbBlue;

pLogPal->palPalEntry[i].peFlags = 0;

m\_lpDibQuad++;

}

//创建逻辑调色板

m\_hPalette = ::CreatePalette(pLogPal);

//释放空间

delete pLogPal;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* Empty()

\*

\*函数参数：

\* 无

\*

\*返回值：

\* 无

\*

\*说明：清理空间

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ImgCenterDib::Empty()

{

//释放DIB内存缓冲区

if(m\_lpDib != NULL) {

delete [] m\_lpDib;

m\_lpDib=NULL;

m\_lpColorTable=NULL;

m\_pImgData=NULL;

m\_lpBmpInfoHead=NULL;

}

//释放逻辑调色板缓冲区

if(m\_hPalette != NULL){

::DeleteObject(m\_hPalette);

m\_hPalette = NULL;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：

\* GetDimensions()

\*

\*函数参数：

\* 无

\*

\*返回值：

\* 图像的尺寸，用CSize类型表达

\*

\*说明：返回图像的宽和高

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CSize ImgCenterDib::GetDimensions()

{

if(m\_lpDib == NULL) return CSize(0, 0);

return CSize(m\_imgWidth, m\_imgHeight);

}