オブジェクト指向と ゲームプログラミング

▶ コンポーネント編 － 第3回　ＤＩＢ ◀

DIB

DIBとは、デバイス独立ビットマップ(Device Independent Bitmap)のことです。前回のDDBとは逆に、デバイスの仕組みや性能に依存しないビットマップ、すなわち、ピクセルの色数やフォーマットなどがビットマップを扱うデバイスによって左右されないビットマップのことを言います。 DIBは、イメージはソフトウェアメモリに配置され、転送はDDBより低速になりますが、プログラマがフォーマットを決めることができるので、イメージを直接書き換えるような場合に使用します。

DIBの生成

DIBオブジェクトは、CreateDIBSection関数で生成することができます。

CreateDIBSection関数

－説明－

CreateDIBSection関数は､アプリケーションから直接書き込み可能なデバイス独立ビットマップを生成します。この関数は、ビットマップのハンドルとイメージが置かれているメモリへのポインタを返します。

－書式－

```
HBITMAP CreateDIBSection(HDC hdc, const BITMAPINFO* pbmi, UINT iUsage, VOID** ppvBits,
                         HANDLE hSection, DWORD dwOffset)
```

－パラメータ－

１つ目の引数(hdc)は、デバイスコンテキストのハンドルです。iUsageパラメータがDIB_PAL_COLORSの場合に使用されます。

２つ目の引数(pbmi)は、DIBのサイズや色情報などさまざまな属性を格納している、BITMAPINFO構造体へのアドレスを指定します。ここで指定した属性のビットマップが生成されます。BITMAPV4HEADERまたはBITMAPV5HEADER構造体を使用すると、より詳細なフォーマットを指定することができます。

３つ目の引数(iUsage)は、色データの種類の指定です。以下のいずれかの値を指定します。

DIB_RGB_COLORS	色データは、RGB値です。
DIB_PAL_COLORS	色データは、hdcパラメータで識別されるデバイスコンテキストの論理パレットに関係する16ビットのインデックス番号です。

４つ目の引数(ppvBits)は、生成されたDIBが保持するイメージへのポインタを受け取る変数(VOID*またはLPVOID型)のアドレスを指定します。

５つ目の引数(hSection)は、DIBを作成するために関数が使うファイルマッピングオブジェクトのハンドルを指定します。使用しない場合はNULLを指定します。

６つ目の引数(dwOffset)は、hSectionパラメータが参照するファイルマッピングオブジェクト内で、ビットマップのイメージが格納されている領域の開始オフセット値を指定します。hSectionパラメータがNULLの場合、この値は無視されます。

－戻り値－

関数が正常に終了した場合は、ビットマップのハンドルを返します。それ以外の場合は、NULLを返します。ビットマップが不用になった場合は、DeleteObject関数にこのハンドルを渡すことで削除することができます。

```
// フォーマットの設定
BITMAPINFO   BmpInfo;
ZeroMemory(&BmpInfo, sizeof(BmpInfo));
BmpInfo.bmiHeader.biSize        = sizeof(BmpInfo.bmiHeader);
BmpInfo.bmiHeader.biWidth       = 640;      // 幅
BmpInfo.bmiHeader.biHeight      = 480;      // 高さ
BmpInfo.bmiHeader.biPlanes      = 1;
BmpInfo.bmiHeader.biBitCount    = 16;       // １ピクセルあたりのビット数
```

```
BmpInfo.bmiHeader.biCompression = BI_RGB;

// DIBオブジェクトの生成
LPVOID    pImage;    // イメージへのポインタを受け取る変数
HBITMAP   hDIB = CreateDIBSection(NULL, &BmpInfo, DIB_RGB_COLORS, &pImage, NULL, 0);
```

CreateDIBSection関数により、指定されたフォーマットを持つビットマップが生成され、そのハンドルとイメージへのポインタが返されます。生成されたイメージは、ビットマップファイルと同じように上下が逆転されて保持されるので、イメージを直接書き換える場合には注意が必要です。なお、ビットマップの高さに負の値を指定すると、上下が逆転しなくなります。

DIBへの描画は、BitBlt関数を始めとするGDI関数で行いますが、ビットマップへ直接描画する関数はありません。GDI関数はデバイスコンテキストへの描画になります。ビットマップからデバイスコンテキストを生成する関数はないので、CreateCompatibleDC関数でメモリデバイスコンテキストを生成し、DIBをそのデバイスコンテキストに割り当てることで代用します。この状態で、GDI関数でそのデバイスコンテキストに描画を行えば、割り当てられたDIBに描画されます。

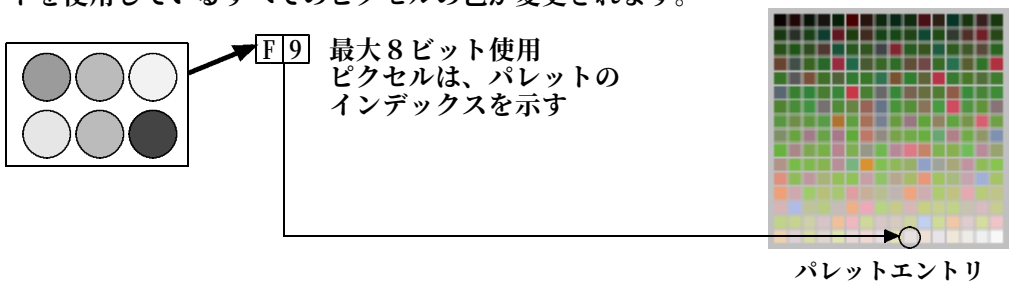
```
// メモリデバイスコンテキストの生成
HDC       hMemDC     = CreateCompatibleDC(NULL);

// メモリデバイスコンテキストにDIBを設定する
HBITMAP   hOldBitmap = (HBITMAP)SelectObject(hMemDC, hDIB);
```

DIBでは、単純な転送を行う場合、BitBlt関数よりCopyMemory関数などのメモリ操作関数を使った方が高速です。

イメージへのアクセス

イメージへのポインタから、イメージのビット列に直接アクセスすることができます。このポインタは、(上下が逆転しているので)イメージの左下を指しています。ポインタに、1ピクセルあたりのバイト数を足すと次のピクセル、1ラインあたりのバイト数を足すと次のラインに移動できます。
ピクセルの色値は、1ピクセルあたりのビット数によりその意味が決定されます。DIBでは、1ピクセルあたりのビット数に1，4，8，16，24，32を指定できます。同じ色でも、ビット数により表現が変わります。

・パレットモード
1ピクセルあたりのビット数が8ビット以下の場合には、色値そのものではなく、パレットエントリのインデックス値となります。このモードには1，4，8ビットがあり、それぞれ2，16，256色の色を扱うことができます。パレットはRGBそれぞれ8ビットが割り当てられ、16,777,216色表現できます。16,777,216色の中から扱う色をパレットエントリに設定します。パレットの内容を変更すると、そのパレットを使用しているすべてのピクセルの色が変更されます。



・16ビットモード
1ピクセルを16ビットで表現します。High Colorモードとも呼ばれ、65,536色扱うことができます。16は3で割り切れませんが、人間の目がほかの色よりも多く識別できる緑に6ビット割り当て、赤5ビット、緑6ビット、青5ビットで構成します。

・24ビットモード
　１ピクセルを24ビットで表現します。Full Colorモードとも呼ばれ、16,777,216色扱うことができます。１ピクセルは、赤８ビット、緑８ビット、青８ビットの計24ビットで表現されます。

・32ビットモード
　１ピクセルを32ビットで表現します。True Colorモードとも呼ばれ、16,777,216色扱うことができます。１ピクセルは、ダミー８ビット、赤８ビット、緑８ビット、青８ビットの計32ビットで表現されます。32ビットですが、色の構成は24ビットモードと同じであり、色数が増えるわけではありません。
　32ビットモードは、24ビットモードよりメモリを多く消費しますが、32ビットという値はシステムと相性がよいため、演算処理や転送効率に高いパフォーマンスが望めます。

　イメージへの直接アクセスは、BitBlt関数などのGDI関数の呼び出し直後に行うと不具合が起こる場合があります。GDI関数の中には、描画が終了する前に制御を戻す関数があります。描画が終了する前にイメージを直接書き換えてしまうと、正しい画像が得られません。このような場合は、GdiFlush関数を呼び出し、未完了のGDI関数をすべて終了させてから直接アクセスを行うようにします。

課　　題

　DIBを管理するクラスCDIBを作成しましょう。

(１)CDIBクラスのヘッダファイル(DIB.hpp)を以下のように作成しましょう。

―DIB.hpp―

```
/*
===========================================================================================================
                                    オブジェクト指向ゲームプログラミング
     Programmed by Hibikino software. Copyright (c) 2004 Hibikino software. All rights reserved.
===========================================================================================================

  【対象ＯＳ】
     Microsoft Windows98/2000以降

  【コンパイラ】
     Microsoft VisualC++ 6.0J ServicePack6

  【プログラム】
     DIB.hpp
                DIB(デバイス独立ビットマップ)クラスヘッダ

  【履歴】
     * Version    1.00       2004/04/dd  hh:mm:ss

===========================================================================================================
*/

#pragma once

/*********************************************************************************************************/
/*                                          インクルードファイル                                         */
/*********************************************************************************************************/
#include <windows.h>

/*********************************************************************************************************/
/*                                          DIBクラス定義                                                */
```

```
/*********************************************************************************************/
class CDIB {
public:
    CDIB();
    CDIB(const LONG inWidth, const LONG inHieght, const WORD inBPP);
    CDIB(LPCTSTR inBMPFileName) { CreateFromBitmap(inBMPFileName); }
    ~CDIB() { Release(); }

    bool Create(const LONG inWidth, const LONG inHieght, const WORD inBPP);
    bool CreateFromBitmap(LPCTSTR inBMPFileName);
    void Release();

    bool LoadFromFile(LPCTSTR inFileName);

    UINT SetPalette(RGBQUAD inColors[], const UINT inEntries);
    UINT GetPalette(RGBQUAD outColors[]);

    HDC    GetDC()     const { return ※ここは各自考えましょう; }
    BITMAP GetBitmap() const { return m_Bitmap; }
    LONG   GetWidth()  const { return ※ここは各自考えましょう; }
    LONG   GetHeight() const { return ※ここは各自考えましょう; }
    LONG   GetPitch()  const { return m_Bitmap.bmWidthBytes;    }
    WORD   GetBPP()    const { return ※ここは各自考えましょう; }
    LPVOID GetImage()  const { return m_Bitmap.bmBits;          }

private:
    HDC         m_hDC;
    HBITMAP     m_hBitmap;
    HBITMAP     m_hDefBitmap;
    BITMAP      m_Bitmap;

    CDIB(const CDIB&);
    CDIB& operator=(const CDIB&);
};
```

(２)CDIBクラスのメンバは、以下のとおりです。

CDIB コンストラクタ

CDIBオブジェクトを構築します。

書式 `CDIB();`

CDIB コンストラクタ

CDIBオブジェクトを構築し、指定されたフォーマットを持つビットマップを生成します。

書式 `CDIB(const LONG inWidth, const LONG inHieght, const WORD inBPP);`

inWidht	ビットマップの幅(ピクセル数)
inHeight	ビットマップの高さ(ピクセル数)
inBPP	ビットマップの１ピクセルあたりのビット数

CDIB コンストラクタ

CDIBオブジェクトを構築し、指定されたビットマップファイルと同じフォーマットのビットマップを生成し、イメージを読み込みます。

書式 `CDIB(LPCTSTR inBMPFileName);`

inBMPFileName	ビットマップファイルの名前

~CDIB デストラクタ

CDIBオブジェクトを解放します。

Create 生成

指定されたフォーマットのビットマップを生成します。

書式 `bool Create(const LONG inWidth, const LONG inHieght, const WORD inBPP);`

Return	成功：true　それ以外：false
inWidht	ビットマップの幅(ピクセル数)
inHeight	ビットマップの高さ(ピクセル数)
inBPP	ビットマップの１ピクセルあたりのビット数

CreateFromBitmap 生成

指定されたビットマップファイルと同じフォーマットのビットマップを生成し、イメージを読み込みます。

書bool CreateFromBitmap(LPCTSTR inBMPFileName);

Return 成功：true　それ以外：false
inBMPFileName ビットマップファイルの名前

Release 解放

CDIBオブジェクトが保持しているビットマップを解放します。

LoadFromFile 読込

指定された画像ファイルを読み込み、ビットマップへ転送します。読み込める形式は、ビットマップ、JPEG、GIFです。

書bool LoadFromFile(LPCTSTR inFileName);

Return 成功：true　それ以外：false
inFileName 画像ファイルの名前

SetPalette 設定

ビットマップがパレットを保持する場合、そのパレットの設定を行います。

書UINT SetPalette(RGBQUAD inColors[], const UINT Entries);

Return 設定できたパレットの数
inColors パレットが格納されたRGBQUAD配列の先頭アドレス
inEntries 設定するパレットの数

GetPalette 取得

ビットマップがパレットを保持する場合、そのパレットを取得します。

書UINT GetPalette(RGBQUAD outColors[]);

Return 取得できたパレットの数
outColors パレットを格納するRGBQUAD配列の先頭アドレス

GetDC アクセス関数

ビットマップと関連づけられているデバイスコンテキストを取得します。

書HDC GetDC() const;

Return デバイスコンテキストのハンドル

GetBitmap アクセス関数

ビットマップの情報を取得します。

書BITMAP GetBitmap() const;

Return ビットマップ情報が格納されたBITMAP構造体

GetWidth アクセス関数

ビットマップの幅を取得します。

書LONG GetWidth() const;

Return ビットマップの幅(ピクセル数)

GetHeight アクセス関数

ビットマップの高さを取得します。

書LONG GetHeight() const;

Return ビットマップの高さ(ピクセル数)

GetPitch アクセス関数

１ラインあたりのバイト数を取得します。Windowsのビットマップでは、１ラインがDWORD境界の配列として構成されているものと仮定しているため、この値は必ず４の倍数になります。

書LONG GetHeight() const;

Return １ラインあたりのバイト数

GetBPP アクセス関数

ビットマップの１ピクセルあたりのビット数を取得します。

書WORD GetBPP() const;

Return １ピクセルあたりのビット数

GetImage アクセス関数

ビットマップのイメージへのポインタを取得します。

書LPVOID GetImage() const;

Return イメージへのポインタ

m_hDC　メンバ変数

CDDBオブジェクトが保持するビットマップと関連づけられているデバイスコンテキストのハンドルです。

書式HDC m_hDC;

m_hBitmap　メンバ変数

CDIBオブジェクトが保持するビットマップのハンドルです。

書式HBITMAP m_hBitmap;

m_hDefBitmap　メンバ変数

CDIBオブジェクトが保持するビットマップと関連づけられているデバイスコンテキストが保持していたデフォルトビットマップのハンドルです。

書式HBITMAP m_hDefBitmap;

m_Bitmap　メンバ変数

CDIBオブジェクトが保持するビットマップの情報が格納されたBITMAP構造体です。

書式BITMAP m_Bitmap;

(３)CDIBクラスのソースファイル(DIB.cpp)を以下のように作成しましょう。

－DIB.cpp－

```
/*
================================================================================
                        オブジェクト指向ゲームプログラミング
     Programmed by Hibikino software. Copyright (c) 2004 Hibikino software. All rights reserved.
================================================================================

  【対象ＯＳ】
     Microsoft Windows98/2000以降

  【コンパイラ】
     Microsoft VisualC++ 6.0J ServicePack6

  【プログラム】
     DIB.cpp
                DIB(デバイス独立ビットマップ)クラス

  【履歴】
     * Version     1.00       2004/04/dd  hh:mm:ss

================================================================================
*/

/*****************************************************************************/
/*                                インクルードファイル                       */
/*****************************************************************************/
#include ※ここは各自考えましょう
#include <olectl.h>

/*****************************************************************************/
/*                              デフォルトコンストラクタ                     */
/*****************************************************************************/
CDIB::CDIB() : m_hDC(NULL), m_hBitmap(NULL), m_hDefBitmap(NULL)
{
    ::ZeroMemory(&m_Bitmap, sizeof(m_Bitmap));
}

/*****************************************************************************/
/*                                コンストラクタ                             */
/*****************************************************************************/
CDIB::CDIB(const LONG inWidth, const LONG inHieght, const WORD inBPP)
    : m_hDC(NULL), m_hBitmap(NULL), m_hDefBitmap(NULL)
{
    Create(inWidth, inHieght, inBPP);
}

/*****************************************************************************/
/*                                DIB生成                                    */
/*****************************************************************************/
bool CDIB::Create(const LONG inWidth, const LONG inHeight, const WORD inBPP)
```

```
{
    return true;
}

/*****************************************************************************************************/
/*                                              DIB生成                                              */
/*****************************************************************************************************/
bool CDIB::CreateFromBitmap(LPCTSTR inBMPFileName)
{
    return true;
}

/*****************************************************************************************************/
/*                                              解　放                                               */
/*****************************************************************************************************/
void CDIB::Release()
{
}

/*****************************************************************************************************/
/*                                          イメージファイル読み込み                                 */
/*****************************************************************************************************/
bool CDIB::LoadFromFile(LPCTSTR inFileName)
{
    return true;
}

/*****************************************************************************************************/
/*                                          パレット設定                                             */
/*****************************************************************************************************/
UINT CDIB::SetPalette(RGBQUAD inColors[], const UINT inEntries)
{
    return ::SetDIBColorTable(m_hDC, 0, inEntries, inColors);
}

/*****************************************************************************************************/
/*                                          パレット取得                                             */
/*****************************************************************************************************/
UINT CDIB::GetPalette(RGBQUAD outColors[])
{
    return ::GetDIBColorTable(m_hDC, 0, 1 << GetBPP(), outColors);
}
```

(４)Create関数の足りない部分を補い、プログラムを完成させましょう。

```
bool CDIB::Create(const LONG inWidth, const LONG inHeight, const WORD inBPP)
{
    Release();

    // デバイスコンテキスト生成
    m_hDC = ::CreateCompatibleDC(NULL);
    if(m_hDC == NULL) {
        ::OutputDebugString("*** Error - デバイスコンテキスト生成失敗(CDIB_Create)\n");
        return false;
    }

    // ビットマップ設定
    BITMAPINFO   bmpi;
    ::ZeroMemory(&bmpi, sizeof(bmpi));
    bmpi.bmiHeader.biSize        = sizeof(bmpi.bmiHeader);
    bmpi.bmiHeader.biWidth       = inWidth;
    bmpi.bmiHeader.biHeight      = inHeight;
    bmpi.bmiHeader.biPlanes      = 1;
    bmpi.bmiHeader.biBitCount    = inBPP;
    bmpi.bmiHeader.biCompression = BI_RGB;

    // DIB生成
    LPVOID   pImage;
    m_hBitmap = ::※ここは各自考えましょう;
    if(m_hBitmap == NULL) {
        ::OutputDebugString("*** Error - DIB生成失敗(CDIB_Create)\n");
        return false;
    }
```

```
    // デバイスコンテキストにビットマップを設定
    m_hDefBitmap = (HBITMAP)::SelectObject(m_hDC, m_hBitmap);
    if(m_hDefBitmap == NULL) {
        ::OutputDebugString("*** Error - ビットマップ設定失敗(CDIB_Create)\n");
        Release();
        return false;
    }

    // ビットマップ情報取得
    ::GetObject(m_hBitmap, sizeof(m_Bitmap), &m_Bitmap);

    return true;
}
```

(５)CreateFromBitmap関数の足りない部分を補い、プログラムを完成させましょう。

```
bool CDIB::CreateFromBitmap(LPCTSTR inBMPFileName)
{
    // ビットマップファイル読み込み
    HBITMAP   hBMP = (HBITMAP)::※ここは各自考えましょう;
    if(hBMP == NULL) {
        ::OutputDebugString("*** Error - ファイル読み込み失敗(CDIB_CreateFromBitmap)\n");
        Release();
        return false;
    }

    // ビットマップ情報取得
    BITMAP   bmp;
    ::?????????(hBMP, sizeof(bmp), &bmp);

    // DIB生成
    if(??????(bmp.bmWidth, bmp.bmHeight, bmp.bmBitsPixel) == false) {
        ::DeleteObject(hBMP);
        return false;
    }

    // メモリデバイスコンテキスト生成
    HDC   hMemDC = ::※ここは各自考えましょう;
    if(hMemDC == NULL) {
        ::OutputDebugString("*** Error - メモリDC生成失敗(CDIB_CreateFromBitmap)\n");
        ::DeleteObject(hBMP);
        return false;
    }

    // 読み込んだビットマップをメモリデバイスコンテキストに設定する
    HBITMAP   hOldBMP = (HBITMAP)::※ここは各自考えましょう;
    if(hOldBMP == NULL) {
        ::OutputDebugString("*** Error - ビットマップ設定失敗(CDIB_CreateFromBitmap)\n");
        ::DeleteDC(hMemDC);
        ::DeleteObject(hBMP);
        return false;
    }

    // パレット設定
    if(bmp.bmBitsPixel <= 8) {
        RGBQUAD   Colors[256];
        const UINT   Entries = ::GetDIBColorTable(hMemDC, 0, 1 << bmp.bmBitsPixel, Colors);
        SetPalette(Colors, Entries);
    }

    // 読み込んだビットマップをDIBに描画
    ::BitBlt(m_hDC, ※ここは各自考えましょう);

    // ビットマップ解放
    ::SelectObject(hMemDC, hOldBMP);
    ::DeleteObject(hBMP);
    ::DeleteDC(hMemDC);

    return true;
}
```

(６)Release関数の足りない部分を補い、プログラムを完成させましょう。

```
void CDIB::Release()
{
    ::ZeroMemory(&m_Bitmap, sizeof(m_Bitmap));

    // ビットマップ設定解除
    if(m_hDefBitmap != NULL) {
        ::SelectObject(m_hDC, m_hDefBitmap);
        m_hDefBitmap = NULL;
    }

    // ビットマップ解放
    if(m_hBitmap != NULL) {
        ::????????????(m_hBitmap);
        m_hBitmap = NULL;
    }

    // デバイスコンテキスト解放
    if(m_hDC != NULL) {
        ::????????(m_hDC);
        m_hDC = NULL;
    }
}
```

(７)LoadFromFile関数を以下のように作成しましょう。

```
bool CDIB::LoadFromFile(LPCTSTR inFileName)
{
    if(m_hDC == NULL) {
        ::OutputDebugString("*** Error - DDB未生成(CDIB_LoadFromFile)\n");
        return false;
    }

    // ファイルオープン
    HANDLE   hFile = ::CreateFile(inFileName, GENERIC_READ, 0, NULL,
                                  OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if(hFile == INVALID_HANDLE_VALUE) {
        ::OutputDebugString("*** Error - ファイルオープン失敗(CDIB_LoadFromFile)\n");
        return false;
    }

    // グローバルメモリ確保
    const DWORD   FileSize = ::GetFileSize(hFile, NULL);        // ファイルサイズ取得
    HGLOBAL       hGlobal  = ::GlobalAlloc(GPTR, FileSize);
    if(hGlobal == NULL) {
        ::OutputDebugString("*** Error - グローバルメモリ確保失敗(CDIB_LoadFromFile)\n");
        ::CloseHandle(hFile);
        return false;
    }

    // ファイル読み込み
    DWORD   Actual;
    ::ReadFile(hFile, hGlobal, FileSize, &Actual, NULL);
    ::CloseHandle(hFile);

    // ストリーム生成
    IStream*    pStream;
    if(::CreateStreamOnHGlobal(hGlobal, FALSE, &pStream) != S_OK) {
        ::OutputDebugString("*** Error - ストリーム生成失敗(CDIB_LoadFromFile)\n");
        ::GlobalFree(hGlobal);
        return false;
    }

    // ピクチャー生成
    IPicture*   pPicture;
    if(::OleLoadPicture(pStream, FileSize, TRUE, IID_IPicture, (LPVOID*)&pPicture) != S_OK) {
        ::OutputDebugString("*** Error - ピクチャー生成失敗(CDIB_LoadFromFile)\n");
        pStream->Release();
        ::GlobalFree(hGlobal);
        return false;
    }

    // ピクチャーサイズ取得
```

```
    OLE_XSIZE_HIMETRIC   Width;     pPicture->get_Width (&Width);
    OLE_YSIZE_HIMETRIC   Height;    pPicture->get_Height(&Height);

    // 単位をHiMetric(1/100mm)からピクセルに変換(2540 = HiMetric / Inch)
    const long   DestWidth  = ::MulDiv(Width,  ::GetDeviceCaps(m_hDC, LOGPIXELSX), 2540);
    const long   DestHeight = ::MulDiv(Height, ::GetDeviceCaps(m_hDC, LOGPIXELSY), 2540);

    // ピクチャー転送
    pPicture->Render(m_hDC, 0, 0, DestWidth, DestHeight, 0, Height, Width, -Height, NULL);

    pPicture->Release();        // ピクチャー解放
    pStream ->Release();        // ストリーム解放
    ::GlobalFree(hGlobal);      // グローバルメモリ解放

    return true;
}
```

(８)CDIBクラスが正しく動作するか確認します。以下のようにプログラムを追加、変更しましょう。

・CTestSceneクラスのメンバに追加

```
CDIB   m_Bitmap;
```

・CTestScene::~CTestScene関数に追加

```
m_Bitmap.Release();
```

・CTestScene::CTestScene関数を以下のように変更

```
CTestScene::CTestScene()
{
    // ここに、機能テストの初期化処理を記述します
    m_Bitmap.Create(640, 480, 24);
    m_Bitmap.LoadFromFile("BMP\\Test.bmp");

    // イメージの書き換え
    for(int y = 0; y < m_Bitmap.GetHeight() / 2; y++) {
        // ポインタを行の先頭に設定
        LPBYTE   pImage = (LPBYTE)m_Bitmap.GetImage() + m_Bitmap.GetPitch() * y;
        for(int x = 0; x < m_Bitmap.GetWidth(); x++) {
            *(pImage + 0) = 255;                // 青成分を最大にする
//          *(pImage + 1) = *(pImage + 1) / 2;  // 緑成分を１／２にする
//          *(pImage + 2) = 0;                  // 赤成分を０にする

            pImage += m_Bitmap.GetBPP() / 8;    // 次のピクセルへ
        }
    }

    FPSTimer().Reset();         // タイマリセット
}
```

・CTestScene::ActivateProc関数を以下のように変更

```
int CTestScene::ActiveProc()
{
    // ここに、機能テストのメイン処理を記述します
    // 内部処理

    // 描画処理
    if(FPSTimer().IsSkip() == false) {
        // ここに、機能テストの描画処理を記述します
        HDC   hDestDC = ::GetDC(GameApp().GetHWnd());
        ::BitBlt(hDestDC, 0, 0, 640, 480, m_Bitmap.GetDC(), 0, 0, SRCCOPY);
        FPSTimer().DrawFPS(hDestDC);
        ::ReleaseDC(GameApp().GetHWnd(), hDestDC);
    }

    return 0;
}
```