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２回　ＤＤＢ ◀

ビットマップオブジェクト

Windowsの提供する機能には、ビットマップやアイコンなど画像を使うものが多数あります。これらの画像は、ビットマップオブジェクトとしてWindowsが管理しています。したがって、これらの機能を使うためには、ビットマップオブジェクトを作成しなければなりません。

ビットマップオブジェクトには、「デバイス依存ビットマップ(DDB:Device Dependent Bitmap)」と「デバイス独立ビットマップ(DIB:Device Independent Bitmap)」の２種類あります。

デバイス依存ビットマップ(DDB)は、デバイスの仕組みや性能に依存するビットマップのことです。デバイスの性能や現在の状態によって、扱うことのできる色数、フォーマットなどが制限されます。また、ピクセルのビット列を取得した場合、それがどのような意味を持つのかに関してもデバイスに依存します。そのため、その値からどんな色なのかを一意的に決定することができません。通常、ビットマップオブジェクトと言ったときは、このDDBオブジェクトを指します。
このように、DDBはデバイスに強く依存しますが、イメージはビデオカード上のメモリ(VRAM)に優先的に配置され、転送もハードウェアアクセラレータにより、高速に行うことができます。

一方、デバイス独立ビットマップ(DIB)は、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では、イメージはソフトウェアメモリに配置され、転送もDDBより低速ですが、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。

DDBの生成

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

CreateCompatibleBitmap関数

－説明－
CreateCompatibleBitmap関数は、指定されたデバイスコンテキストと互換性のあるビットマップを生成します。

－書式－
HBITMAP CreateCompatibleBitmap(HDC *hdc*, int *nWidth*, int *nHeight*)

－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。このデバイスコンテキストと互換性のあるビットマップが生成されます。

２つ目の引数(nWidth)は、ビットマップの幅(ピクセル数)です。

３つ目の引数(nHeight)は、ビットマップの高さ(ピクセル数)です。

－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。

```
// DDBオブジェクトの生成
HDC       hDC  = GetDC(hWnd);   // hWndはウィンドウのハンドル
HBITMAP   hDDB = CreateCompatibleBitmap(hDC, 640, 480);
ReleaseDC(hWnd, hDC);
```

CreateCompatibleBitmap関数により、指定されたデバイスコンテキストと同等の色数、フォーマットを持つビットマップが生成され、そのハンドルが返されます。生成されたビットマップの内容は不定であり、真っ黒の場合もあれば、ゴミのような状態の場合もあります。

DDB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、DDB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DDB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DDBを設定する
HBITMAP    hOldBitmap = (HBITMAP)SelectObject(hMemDC, hDDB);
```

課　題

DDBを管理するクラスCDDBを作成しましょう。

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

－DDB.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】
     DDB.hpp
                DDB(デバイス依存ビットマップ)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>

/*************************************************************************************************/
/*                                          DDBクラス定義                                        */
/*************************************************************************************************/
class CDDB {
public:
    CDDB();
    CDDB(const HDC hDC, const int inWidth, const int inHieght);
    CDDB(const HDC hDC, LPCTSTR inBMPFileName) { CreateFromBitmap(hDC, inBMPFileName); }
    ~CDDB() { Release(); }

    bool Create(const HDC hDC, const int inWidth, const int inHeight);
    bool CreateFromBitmap(const HDC hDC, LPCTSTR inBMPFileName);
    void Release();

    bool LoadFromFile(LPCTSTR inFileName);

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

    // アクセス関数
    HDC    GetDC()     const { return ?????;    }
```

```
    BITMAP GetBitmap() const { return m_Bitmap; }
    LONG   GetWidth()  const { return m_Bitmap.???????;     }
    LONG   GetHeight() const { return m_Bitmap.????????;    }
    WORD   GetBPP()    const { return m_Bitmap.???????????; }

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

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

(２)CDDBクラスのメンバは、以下のとおりです。

CDDB コンストラクタ

CDDBオブジェクトを構築します。

書式 CDDB();

CDDB コンストラクタ

CDDBオブジェクトを構築し、指定されたデバイスコンテキストと同等の能力を持つビットマップを生成します。

書式 CDDB(const HDC hDC, const int inWidth, const int inHieght);

hDC	デバイスコンテキストのハンドル
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)

CDDB コンストラクタ

CDDBオブジェクトを構築し、指定されたデバイスコンテキストと同等の能力を持つビットマップを生成し、指定されたビットマップファイルを読み込みます。

書式 CDDB(const HDC hDC, LPCTSTR inBMPFileName);

hDC	デバイスコンテキストのハンドル
inBMPFileName	ビットマップファイルの名前

~CDDB デストラクタ

CDDBオブジェクトを解放します。

Create 生成

指定されたデバイスコンテキストと同等の能力を持つ空のビットマップを生成します。

書式 bool Create(const HDC hDC, const int inWidth, const int inHeight);

Return	成功：true　それ以外：false
hDC	デバイスコンテキストのハンドル
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)

CreateFromBitmap 生成

指定されたデバイスコンテキストと同等の能力を持つビットマップを生成し、指定されたビットマップファイルを読み込みます。

書式 bool CreateFromBitmap(const HDC hDC, LPCTSTR inBMPFileName);

Return	成功：true　それ以外：false
hDC	デバイスコンテキストのハンドル
inBMPFileName	ビットマップファイルの名前

Release 解放

CDDB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。

書式 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。

書式 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。

書式 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。

書式 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。

書式 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。

書式 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。

書式 LONG GetHeight() const;

Return	ビットマップの高さ(ピクセル数)

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。

書式 WORD GetBPP() const;

Return	１ピクセルあたりのビット数

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

CDDB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

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

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

－DDB.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】
     DDB.cpp
                DDB(デバイス依存ビットマップ)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                               */
/*********************************************************************************************/
CDDB::CDDB() : m_hDC(NULL), m_hBitmap(NULL), m_hDefBitmap(NULL)
{
    ::ZeroMemory(&m_Bitmap, sizeof(m_Bitmap));
}

/*********************************************************************************************/
/*                                        コンストラクタ                                       */
/*********************************************************************************************/
CDDB::CDDB(const HDC hDC, const int inWidth, const int inHieght)
    : m_hDC(NULL), m_hBitmap(NULL), m_hDefBitmap(NULL)
{
    Create(hDC, inWidth, inHieght);
}

/*********************************************************************************************/
/*                                          DDB生成                                          */
/*********************************************************************************************/
bool CDDB::Create(const HDC hDC, const int inWidth, const int inHeight)
{
    return true;
}

/*********************************************************************************************/
/*                                          DDB生成                                          */
/*********************************************************************************************/
bool CDDB::CreateFromBitmap(const HDC hDC, LPCTSTR inBMPFileName)
{
    return true;
}

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

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

/*********************************************************************************************/
/*                                        パレット設定                                         */
```

```
/*****************************************************************************************************/
UINT CDDB::SetPalette(RGBQUAD inColors[], const UINT inEntries)
{
    return ::SetDIBColorTable(m_hDC, 0, Entries, inColors);
}

/*****************************************************************************************************/
/*                                          パレット取得                                             */
/*****************************************************************************************************/
UINT CDDB::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 CDDB::Create(const HDC hDC, const int inWidth, const int inHeight)
{
    Release();

    // デバイスコンテキスト、DDB生成
    if(hDC != NULL) {
        m_hDC     = ::CreateCompatibleDC(hDC);
        m_hBitmap = ::?????????????????????(hDC, ???????, ????????);
    } else {
        // hDCにNULLが渡されたときは、ディスプレイ互換のDCを作成する
        HDC    hDisplayDC = ::CreateDC("DISPLAY", NULL, NULL, NULL);

        m_hDC     = ::CreateCompatibleDC(hDisplayDC);
        m_hBitmap = ::?????????????????????(hDisplayDC, ???????, ????????);

        ::DeleteDC(hDisplayDC);
    }
    if(m_hDC == NULL || m_hBitmap == NULL) {
        ::OutputDebugString("*** Error - DDB生成失敗(CDDB_Create)\n");
        Release();
        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(CDDB_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 CDDB::CreateFromBitmap(const HDC hDC, 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(CDDB_CreateFromBitmap)\n");
        Release();
        return false;
    }

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

    // DDB生成
    if(??????(hDC, bmp.bmWidth, bmp.bmHeight) == 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(CDDB_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(CDDB_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DDB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 CDDB::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 CDDB::LoadFromFile(LPCTSTR inFileName)
{
    if(m_hDC == NULL) {
        ::OutputDebugString("*** Error - DDB未生成(CDDB_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(CDDB_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(CDDB_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(CDDB_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(CDDB_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)CDDB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

```
CDDB   m_Bitmap;
```

・CTestScene::CTestScene関数に追加

```
m_Bitmap.CreateFromBitmap(NULL, "BMP\\Test.bmp");
```

・CTestScene::~CTestScene関数に追加

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

・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;
}
```