オブジェクト指向と
ゲームプログラミング

▶ DirectX Graphics 2D編 － 第５回　サーフェス ◀

サーフェス

DirectX Graphicsでは、画像を保持しておく領域をサーフェス(Surface)と呼びます。スプライト(テクスチャ)のイメージも、サーフェスに格納されています。
サーフェスは、目的別に「フロントバッファ」「バックバッファ」「オフスクリーンサーフェス」があります。

フロントバッファ
フロントバッファは、ディスプレイに表示されている画像を保持するサーフェスです。DirectX Graphicsでは、フロントバッファのコピーしか取得できないようになっているので、直接ふれることはできません。

バックバッファ
バックバッファは、フロントバッファに接続(チェイン)されているサーフェスですが、画面に表示されません。いわば、フロントバッファの裏側です。ディスプレイに同期してフロントバッファと瞬時に入れ換えることができるので、ちらつきのない描画を実現するために使用されます。

オフスクリーンサーフェス
オフスクリーンサーフェスは、背景やキャラクタ、テクスチャなどのイメージを保持しておく領域です。このサーフェスも画面に表示されません。バックバッファと違い、フロントバッファから完全に独立しており、サイズやピクセルフォーマットを自由に指定して作成することができます。画像を保持したり、画像を加工するためのサーフェスです。

サーフェスの取得・生成と解放

DirectX Graphicsでは、フロントバッファとバックバッファは、Direct3DDevice9オブジェクトを生成したときに、自動的に生成されます。フロントバッファのコピーの取得はIDirect3DDevice9::GetFrontBufferDataメソッド、バックバッファの取得はIDirect3DDevice9::GetBackBufferメソッドで行います。また、オフスクリーンサーフェスの生成は、IDirect3DDevice9::CreateOffscreenPlainSurfaceメソッドで行います。
サーフェスは、IDirect3DSurface9インタフェースで制御します。このインタフェースを用いれば、デバイスコンテキストの取得やサーフェスメモリへのポインタの取得などが行えます。
取得および生成したサーフェスは、不要になったらIDirect3DSurface9::Releaseメソッドで解放します。

課　題

バックバッファを管理するクラスを作成しましょう。

(１)バックバッファクラスのヘッダファイル(BackBuffer.hpp)を次のように作成しましょう。

－BackBuffer.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) 2005 Hibikino software. All rights reserved.
==========================================================================================

【対象ＯＳ】
    Microsoft Windows2000/XP

【コンパイラ】
    Microsoft Visual C++ 2005

【プログラム】
```

```
    BackBuffer.hpp
                バックバッファクラスヘッダ

  【履歴】
    * Version    1.00       2005/04/dd  hh:mm:ss

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

#pragma once

/*********************************************************************************************/
/*                                        インクルードファイル                                 */
/*********************************************************************************************/
#include <d3dx9.h>

/*********************************************************************************************/
/*                                      バックバッファクラス定義                               */
/*********************************************************************************************/
class CBackBuffer {
public:

private:
};
```

(２)バックバッファクラスのソースファイル(BackBuffer.cpp)を以下のように作成しましょう。

－BackBuffer.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) 2005 Hibikino software. All rights reserved.
================================================================================================

  【対象ＯＳ】
    Microsoft Windows2000/XP

  【コンパイラ】
    Microsoft Visual C++ 2005

  【プログラム】
    BackBuffer.hpp
                バックバッファクラスヘッダ

  【履歴】
    * Version    1.00       2005/04/dd  hh:mm:ss

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

#pragma once

/*********************************************************************************************/
/*                                        インクルードファイル                                 */
/*********************************************************************************************/
#include "BackBuffer.hpp"
#include <cassert>
```

(３)バックバッファクラスに、バックバッファを管理するのに必要なメンバを追加します。以下のプログラムを適切な場所に追加しましょう。

```
    IDirect3DDevice9*    m_pD3DDevice;
    IDirect3DSurface9*   m_pD3DSurface;
```

バックバッファを管理するだけなら、IDirect3DSurface9インタフェースだけでよさそうですが、デバイスをリセットしたり、バックバッファを塗りつぶしたりするときにDirect3DDevice9オブジェクトがあるとプログラムが簡潔になるので、これもメンバ変数に加えておきます。

(４)バックバッファクラスのコンストラクタとデストラクタを作成します。以下のプロトタイプを適切な場所に追加しましょう。

```
CBackBuffer(IDirect3DDevice9* pD3DDevice);
virtual ~CBackBuffer();
```

コンストラクタでは、メンバ変数を初期化しますが、初期化に必要な情報のうち、外部から受け取らなければならない値を引数としています。コンストラクタの仕様は、以下のとおりです。

CBackBuffer　コンストラクタ

CBackBufferオブジェクトを構築します。

試 CBackBuffer(IDirect3DDevice9* pD3DDevice);

pD3DDevice　Direct3DDevice9オブジェクトへのインタフェース

(５)コンストラクタを実装します。以下のプログラムを完成させ、ソースファイルに追加しましょう。

```
/*****************************************************************************************************/
/*                                          コンストラクタ                                           */
/*****************************************************************************************************/
CBackBuffer::CBackBuffer(IDirect3DDevice9* pD3DDevice) : m_pD3DDevice(NULL), m_pD3DSurface(NULL)
{
    assert(pD3DDevice != NULL);

    m_pD3DDevice = pD3DDevice;
    m_pD3DDevice->AddRef();     // 参照カウンタインクリメント

    // バックバッファ取得
    m_pD3DDevice->????????????(0, 0, D3DBACKBUFFER_TYPE_MONO, &m_pD3DSurface);
}
```

コンストラクタでは、受け取ったインタフェースのAddRefメソッドを呼び出し、参照カウンタを増やしています。COMインタフェースを別の変数にコピーした場合、その期間が一時的でない場合は、このようにした方が安全です。

この例では、バックバッファオブジェクトが存在している間は、各インタフェースが参照しているオブジェクトが解放されなくなります(ただし、不正にReleaseメソッドが呼び出された場合はこの限りではありません)。もちろん、このままでは参照カウンタが増えたままになり、解放されなくなるので、バックバッファオブジェクトが破棄されるときに、Releaseメソッドを呼び出して参照カウンタを減らします。

(６)デストラクタを実装します。以下のプログラムをソースファイルに追加しましょう。

```
/*****************************************************************************************************/
/*                                          デストラクタ                                             */
/*****************************************************************************************************/
CBackBuffer::~CBackBuffer()
{
    SafeRelease(m_pD3DSurface);
    SafeRelease(m_pD3DDevice);
}
```

デストラクタでは、コンストラクタで増やされたインタフェースの参照カウンタを減らし、正しく解放されるようにします。

(７)デバイスのリセット処理を作成します。

バックバッファは、ビデオメモリ(VRAMまたはAGPメモリ)に作成され、D3DPOOL_DEFAULTプールというメモリクラスに配置されます。D3DPOOL_DEFAULTプールのリソースは、IDirect3DDevice9::Resetメソッドを呼び出す前に解放し、リセット後に復元しなければ、再度使用することができません。

デバイスのリセット前後に呼び出す以下の関数のプロトタイプを適切な場所に追加しましょう。

```
void OnLostDevice();
void OnResetDevice();
```

(８)デバイスのリセット前に呼び出すOnLostDevice関数を実装します。以下のプログラムをソースファイルに追加しましょう。

```
/*****************************************************************************************************/
/*                                          デバイス消失処理                                         */
/*****************************************************************************************************/
void CBackBuffer::OnLostDevice()
{
    SafeRelease(m_pD3DSurface);
}
```

OnLostDevice関数では、(７)の仕様にしたがい、サーフェスをいったん解放します。

(９)デバイスのリセット後に呼び出すOnRestDevice関数を実装します。以下のプログラムを完成させ、ソースファイルに追加しましょう。

```
/*****************************************************************************************************/
/*                                          デバイスリセット処理                                     */
/*****************************************************************************************************/
void CBackBuffer::OnResetDevice()
{
    m_pD3DDevice->?????????????(0, 0, D3DBACKBUFFER_TYPE_MONO, &m_pD3DSurface);
}
```

OnLostDevice関数では、(８)で解放されたバックバッファを再度取得します。

(10)バックバッファを指定した色で塗りつぶす機能を追加します。

IDirect3DDevice9::ColorFillメソッドは、D3DPOOL_DEFAULTプールに配置されたサーフェスを塗りつぶすことができます。

以下のプログラムを完成させ、適切な場所に追加しましょう。また、プロトタイプも適切な場所に作成しましょう。

```
/*****************************************************************************************************/
/*                                          塗りつぶし                                               */
/*****************************************************************************************************/
void CBackBuffer::ColorFill(const RECT* pFill, const D3DCOLOR inColor)
{
    m_pD3DDevice->?????????(m_pD3DSurface, NULL, inColor);
}
```

ColorFill関数の仕様は、以下のとおりです。

ColorFill　　描画

バックバッファの指定された領域を指定された色で塗りつぶします。

```
試ColorFill(const RECT* pFill, const D3DCOLOR inColor);
```

pFill	塗りつぶす領域を格納したRECT構造体のアドレス。NULLを指定すると全領域
inColor	D3DCOLOR型の塗りつぶす色値

(11)バックバッファは、グラフィック機能の一部と考えられるので、CDXGraphics9クラスに集約します。以下のプログラムを適切な場所に追加しましょう。

```
CBackBuffer*    m_pBackBuffer;        // バックバッファ
```

(12)以下のコンストラクタ初期化子を適切な場所に追加しましょう。

```
m_pBackBuffer(NULL)
```

(13)DirectX Graphicsの初期化時に、バックバッファクラスのオブジェクトを生成します。以下のプログラムを適切な場所に追加しましょう。

```
// バックバッファ生成
m_pBackBuffer = new CBackBuffer(m_pD3DDevice);
```

(14)DirectX Graphicsの解放時に、バックバッファクラスのオブジェクトを解放します。以下のプログラムを適切な場所に追加しましょう。

```
// バックバッファ解放
delete m_pBackBuffer;
m_pBackBuffer = NULL;
```

(15)デバイスのリセット前に、バックバッファクラスのあるメンバ関数を呼び出します。以下のプログラムを完成させ、CDXGraphics9::Reset関数の適切な場所に追加しましょう。

```
// リセット前処理
m_pBackBuffer->???????????();
```

(16)デバイスのリセット後に、バックバッファクラスのあるメンバ関数を呼び出します。以下のプログラムを完成させ、CDXGraphics9::Reset関数の適切な場所に追加しましょう。

```
// リセット後処理
m_pBackBuffer->????????????();
```

(17)バックバッファクラスのオブジェクトを返す関数を作成します。以下のプログラムを適切な場所に追加しましょう。

```
CBackBuffer* GetBackBuffer() const { return m_pBackBuffer; }
```

GetBackBuffer関数により、クラスの外部でもバックバッファの機能を使用することができるようになります。

(18)以下のプログラムを適切な場所に追加しましょう。

```
inline CBackBuffer* DXGBackBuffer() { return DXGraphics().GetBackBuffer(); }
```

バックバッファの機能を使うには、「DXGraphics().GetBackBuffer()->メンバ関数名」のような長い名前が必要になります。上のようなインライン関数を定義しておくと、「DXGBackBuffer()->メンバ関数名」というわかりやすく、かつ短い名前で使用することができるようになります。