オブジェクト指向と ゲームプログラミング

▶ DirectX Graphics 2D編 － 第６回　テクスチャとサーフェス ◀

テクスチャサーフェスの取得

テクスチャのイメージは、サーフェスが保持しています。このサーフェスは、IDirect3DTexture9::GetSurfaceLevelメソッドで取得することができます。
テクスチャのサーフェスもIDirect3DSurface9インタフェースをとおして操作を行うので、バックバッファやオフスクリーンサーフェスと同様の機能を持ちます。また、テクスチャはD3DPOOL_MANAGEDプールで作成することが多いので、デバイスリセット時のわずらわしい解放と再生成処理は不要です。

課　　題

スプライトクラスにサーフェスの機能を追加しましょう。

(１)CSpriteクラスに、サーフェス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。

```
IDirect3DSurface9*    m_pD3DSurface;
```

(２)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_pD3DSurface(NULL)
```

(３)CSpriteクラスのオブジェクト生成時に、サーフェスの取得を行います。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。

```
// サーフェス取得
m_pD3DTexture->??????????????(0, &m_pD3DSurface);
```

(４)CSpriteクラスのオブジェクト解放時に、サーフェスの解放を行います。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。

```
m_pD3DSurface->Release();
```

(５)スプライトクラスに、サーフェスを利用した機能を追加します。以下のプロトタイプをISpriteクラス定義の適切な場所に追加しましょう。

```
virtual void ColorFill(const RECT* pFill, const D3DCOLOR inColor) = 0;

virtual void LoadFromFile  (LPCTSTR inFileName,  const RECT* pSrcRect, const RECT* pDestRect,
                            const DWORD inFilter = D3DX_DEFAULT, const D3DCOLOR inColorKey = 0) = 0;
virtual void LoadFromSprite(ISprite* pSrcSprite, const RECT* pSrcRect, const RECT* pDestRect,
                            const DWORD inFilter = D3DX_DEFAULT, const D3DCOLOR inColorKey = 0) = 0;
```

各関数の仕様は、以下のとおりです。

ColorFill 描画

スプライトの指定された領域を指定された色で塗りつぶします。

書式 `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

LoadFromFile 読込

ファイルからイメージを読み込みます。

書式
```
void LoadFromFile(LPCTSTR inFileName, const RECT* pSrcRect, const RECT* pDestRect,
                  const DWORD inFilter = D3DX_DEFAULT, const D3DCOLOR inColorKey = 0);
```

inFileName　読み込むファイルの名前
pSrcRect　転送元の領域を格納したRECT構造体のアドレス。NULLを指定すると全領域
pDestRect　転送先の領域を格納したRECT構造体のアドレス。NULLを指定すると全領域
inFilter　イメージを拡大縮小するときに使用するフィルタ。透明色を省略するときは省略可
inColorKey　D3DCOLOR型の透明色の色値を指定。省略すると透明色なし

LoadFromSprite　読込

スプライトからイメージを読み込みます。

書式 `void LoadFromSprite(ISprite* pSrcSprite, const RECT* pSrcRect, const RECT* pDestRect, const DWORD inFilter = D3DX_DEFAULT, const D3DCOLOR inColorKey = 0);`

pSrcSprite　転送元となるスプライトインタフェースのポインタ
pSrcRect　転送元の領域を格納したRECT構造体のアドレス。NULLを指定すると全領域
pDestRect　転送先の領域を格納したRECT構造体のアドレス。NULLを指定すると全領域
inFilter　イメージを拡大縮小するときに使用するフィルタ。透明色を省略するときは省略可
inColorKey　D3DCOLOR型の透明色の色値を指定。省略すると透明色なし

(６)CSpriteクラスにColorFill関数を追加します。この関数のプロトタイプを(５)をもとに作成し、適切な場所に追加しましょう。

(７)ColorFill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。

```
/*****************************************************************************************************/
/*                                          塗りつぶし                                               */
/*****************************************************************************************************/
void CSprite::ColorFill(const RECT* pFill, const D3DCOLOR inColor)
{
    IDirect3DSurface9*   pFillSurface = NULL;
    IDirect3DDevice9*    pD3DDevice   = NULL;

    try {
        // 塗りつぶし領域設定
        RECT   fill;
        if(pFill == NULL) {
            D3DSURFACE_DESC   desc;
            m_pD3DSurface->GetDesc(&desc);
            ::SetRect(&fill, 0, 0, desc.Width, desc.Height);
        } else {
            fill = *pFill;
        }

        // Direct3Dデバイスオブジェクト取得
        if(m_pD3DTexture->GetDevice(&pD3DDevice) != D3D_OK)
            throw "デバイスオブジェクト取得失敗";

        // 塗りつぶし用のサーフェスを生成
        if(pD3DDevice->CreateOffscreenPlainSurface(fill.right - fill.left, fill.bottom - fill.top,
                                                   D3DFMT_A8R8G8B8, D3DPOOL_DEFAULT, &pFillSurface, NULL))
            throw "塗りつぶしサーフェス生成失敗";

        // 指定された色で塗りつぶす
        pD3DDevice->?????????(pFillSurface, NULL, inColor);

        // 塗りつぶし用サーフェスをテクスチャのサーフェスへロード
        ::D3DXLoadSurfaceFromSurface(m_pD3DSurface, NULL, &fill,
                                     pFillSurface, NULL, NULL, D3DX_FILTER_POINT, 0);
    } catch(LPCTSTR ErrorString) {
        // エラーメッセージ生成
        TCHAR   msg[128];
        ::wsprintf(msg, "*** Error - %s(CSprite_ColorFill)\n", ErrorString);
        ::OutputDebugString(msg);
    }

    SafeRelease(pFillSurface);
    SafeRelease(pD3DDevice);
}
```

IDirect3DDevice9::ColorFillメソッドは、D3DPOOL_DEFAULTプールのサーフェスしか塗りつぶせません。テクスチャは、D3DPOOL_MANAGEDプールに配置されることが多いので、D3DPOOL_DEFAULTプールのサーフェスを生成、それを塗りつぶし、テクスチャのサーフェスにロードしています。

(８)CNullSpriteクラスに「何もしない」ColorFill関数を実装します。以下のプログラムを適切な場所に追加しましょう。

```
virtual void ColorFill(const RECT* pFill, const D3DCOLOR inColor) {}
```

(９)CSpriteクラスにLoadFromFile関数を追加します。この関数のプロトタイプを(５)をもとに作成し、適切な場所に追加しましょう。

(10)LoadFromFile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。

```
/*****************************************************************************************************/
/*                                          ファイル読み込み                                         */
/*****************************************************************************************************/
void CSprite::LoadFromFile(LPCTSTR inFileName, const RECT* pSrcRect, const RECT* pDestRect,
                           const DWORD inFilter, const D3DCOLOR inColorKey)
{
    ::D3DX??????????????????(m_pD3DSurface, NULL, pDestRect, inFileName, pSrcRect,
                             inFilter, inColorKey, NULL);
}
```

Direct3DXには、ファイルからサーフェスにイメージを読み込む関数が用意されています。この関数を用いれば、サーフェスに画像を読み込むプログラムが簡単に作成できます。対応しているファイルフォーマットは、.bmp、.dds、.dib、.hdr、.jpg、.pfm、.png、.ppm、.tgaです。

(11)CNullSpriteクラスに「何もしない」LoadFromFile関数を実装しましょう。

(12)CSpriteクラスにLoadFromSprite関数を追加します。この関数のプロトタイプを(５)をもとに作成し、適切な場所に追加しましょう。

(13)スプライトクラスにLoadFromSprite関数を実装するために必要な作業用の関数を作成します。

サーフェスからサーフェスへのイメージの転送は、IDirect3DDevice9::StretchRectメソッドとIDirect3DDevice9::UpdateSurfaceメソッドがありますが、これらのメソッドは、メモリクラスに制限があり、テクスチャに多く使われるD3DPOOL_MANAGEDプールには使用できません。また、ピクセルフォーマットも同一でないと転送できません。そこで、Direct3DXのD3DXLoadSurfaceFromSurface関数を用います。この関数は、転送速度はかなり低速ですが、拡大縮小、フォーマット変換機能、カラーキー処理、クリッピング、メモリクラス制限なしといった多くの利点があります。
D3DXLoadSurfaceFromSurface関数を使用するには、転送元と転送先のIDirect3DSurface9インタフェースが必要になります。スプライトクラスには、IDirect3DSurface9インタフェースを取得する機能がないので、追加しなければなりません。

IDirect3DSurface9インタフェースを返すGetSurface関数のプロトタイプをISpriteクラス定義の適切な場所に追加しましょう。

```
virtual IDirect3DSurface9* GetSurface() = 0;
```

(14)CSpriteクラスに、GetSurface関数を実装します。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。

```
virtual IDirect3DSurface9* GetSurface() { return ????????????; }
```

(15)CNullSpriteクラスに、GetSurface関数を実装します。以下のプログラムを適切な場所に追加しましょう。

```
virtual IDirect3DSurface9* GetSurface() { return NULL; }
```

CNullSpriteクラスは、サーフェスを保持していないので、NULLを返します。

(16)CSpriteクラスに、LoadFromSprite関数を実装します。次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。

```
/*****************************************************************************************************/
/*                                          スプライトロード                                           */
/*****************************************************************************************************/
void CSprite::LoadFromSprite(ISprite* pSrcSprite, const RECT* pSrcRect, const RECT* pDestRect,
                             const DWORD inFilter, const D3DCOLOR inColorKey)
{
    // 転送元サーフェス取得
    IDirect3DSurface9*   pSrcSurface = pSrcSprite->?????????();
    if(pSrcSurface == NULL)
        return;

    // ロード
    ::D3DX?????????????????????(m_pD3DSurface, NULL, pDestRect, pSrcSurface, NULL, pSrcRect,
                                 inFilter, inColorKey);
}
```