# オブジェクト指向と ゲームプログラミング

## ▶ DirectX Graphics 2D編 － 第８回　ピクセルフォーマット ◀

### ピクセルフォーマット

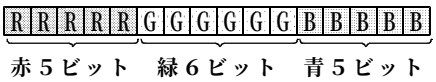
画像処理においてもっとも理解しにくいと言われているのがピクセルフォーマットです。しかし、これが理解できれば、テクスチャやサーフェスのピクセルに直接アクセスして書き換えることが可能になり、高度な画像効果を作成することができます。

DirectX Graphicsのピクセルは、RGBと呼ばれる光の三原色、赤(Red)、緑(Green)、青(Blue)とアルファチャンネル(Alpha Channel)を調合して作られます。ピクセルフォーマットとは、サーフェスの１ピクセルを構成するビット数と、そのビットの並びかたのことです。同じ色でも、ピクセルフォーマットにより表現の方法が変わります。

#### ○１６ビットのピクセルフォーマット

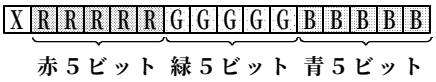
**Ｒ５：Ｇ６：Ｂ５(D3DFMT_R5G6B5)**

１６ビットを赤５ビット、緑６ビット、青５ビットと割り当てたピクセルフォーマットです。人間の目は、緑がもっとも敏感に識別できると言われているので、緑に１ビット多く割り当てています。65,535色扱うことができます。



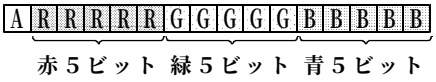

**Ｘ１：Ｒ５：Ｇ５：Ｂ５(D3DFMT_X1R5G5B5)**

１ピクセルあたり１６ビット割り当てられますが、３で割った余りの１ビットは使用せず、RGB各成分に５ビットずつ割り当てたピクセルフォーマットです。32,678色扱うことができます。




**Ａ１：Ｒ５：Ｇ５：Ｂ５(D3DFMT_A1R5G5B5)**

RGB各成分に５ビット、アルファチャンネルに１ビット割り当てたピクセルフォーマットです。



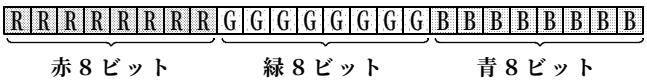

**Ａ４：Ｒ４：Ｇ４：Ｂ４(D3DFMT_X4R4G4B4)**

RGBおよびアルファチャンネルの各成分に４ビット割り当てたピクセルフォーマットです。4,096色扱うことができます。


#### ○２４ビットのピクセルフォーマット

**Ｒ８：Ｇ８：Ｂ８(D3DFMT_R8G8B8)**

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。16,777,216色扱うことができます。




#### ○３２ビットのピクセルフォーマット

**Ｘ８：Ｒ８：Ｇ８：Ｂ８(D3DFMT_X8R8G8B8)**

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。本質はD3DFMT_R8G8B8と同じなので、色数は増えていません。にもかかわらずの多くメモリを消費しているのは、３２ビットというデータがシステムにとって相性がよいため、演算や転送効率に高いパフォーマンスが望めるためです。RGBの並びが逆順のD3DFMT_X8B8G8R8というフォーマットもあります。


**Ａ８：Ｒ８：Ｇ８：Ｂ８**(D3DFMT_A8R8G8B8)
RGBおよびアルファチャンネルの各成分に８ビット割り当てたピクセルフォーマットです。RGBの並びが逆順のD3DFMT_A8B8G8R8というフォーマットもあります。


**Ａ２：Ｒ１０：Ｇ１０：Ｂ１０**(D3DFMT_A10R10G10B10)
RGB各成分に１０ビット、アルファチャンネルに２ビット割り当てたピクセルフォーマットです。1,073,741,824(約１０億)色表現することができます。RGBの並びが逆順のD3DFMT_A2B10G10R10というフォーマットもあります。


DirectX Graphicsでは、これらのフォーマットのほか、６４ビットのピクセルフォーマットやピクセルの色値を浮動小数点で扱う浮動小数点数フォーマットもサポートしています。

## D3DCOLORマクロ

RGBとアルファチャンネルを調合して色を作成するために、DirectX GraphicsではD3DCOLORで始まるD3DCOLOR_RGBA、D3DCOLOR_XRGB、D3DCOLOR_ARGBマクロを提供しています。これらのマクロは、赤、緑、青の輝度と透明度をそれぞれ0～255の256段階で指定すると、その色を表すD3DCOLOR型の32ビット値(フォーマットはD3DFMT_A8R8G8B8)を返します。
たとえば、ピンク色(赤：255、緑：128、青：192、アルファ：96)を作成するには、

```
D3DCOLOR    color = D3DCOLOR_RGBA(255, 128, 192, 96);
```

とします。返された値は、DirectX Graphicsのメソッドや関数を用いて描画するときに使用します。デバイスコンテキストでは使用できません。テクスチャやサーフェスにアクセスしてピクセルを直接書き換える場合は、フォーマットが同一の場合を除き、この値をそのまま使用することはできません。

## テクスチャとサーフェスへのアクセス

サーフェスメモリへアクセスしてピクセルデータを書き換えることができます。サーフェスメモリへのアクセスするには、IDirect3DSurface9::LockRectメソッドを呼び出し、サーフェスをロックしてポインタを取得します。
サーフェスメモリへのアクセスが終了した場合は、IDirect3DSurface9::::UnlockRectメソッドを呼び出してロックを解除します。ロックを解除しないと、描画できません。

```
// サーフェスメモリへのポインタ取得(pD3DSurfaceは初期化済みのサーフェスインタフェース)
D3DLOCKED_RECT    lock;
pD3DSurface->Lock(&lock, NULL, 0);

// lock.pBitsにポインタが返されるので、これを介してサーフェスメモリにアクセスする

pD3DSurface->UnlockRect();  // ロック解除
```

テクスチャでは、Direct3DTexture9::LockRectメソッドとIDirect3DTexture9::::UnlockRectメソッドを提供しているので、サーフェスを取得しなくてもイメージへアクセスできます。

```
// テクスチャイメージへのポインタ取得(pD3DTextureは初期化済みのテクスチャインタフェース)
pD3DTexture->Lock(0, &lock, NULL, 0);

// lock.pBitsにポインタが返されるので、これを介してテクスチャイメージにアクセスする

pD3DTexture->UnlockRect();  // ロック解除
```

# 課　題

バックバッファクラスとスプライトクラスに、サーフェス(テクスチャ)の幅や高さといった情報を取得する機能と、イメージを格納しているメモリへのポインタを取得する機能を追加しましょう。

(１)バックバッファクラスに、サーフェスの幅や高さといった情報を取得するGetDesc関数を作成します。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。また、プロトタイプも忘れずに作成しましょう。

```
/*****************************************************************************************************/
/*                                          サーフェス記述取得                                         */
/*****************************************************************************************************/
D3DSURFACE_DESC CBackBuffer::GetDesc()
{
    // テクスチャ記述取得
    D3DSURFACE_DESC    desc;
    m_pD3DSurface->GetDesc(&desc);

    return desc;
}
```

(２)スプライトクラスに、スプライトの幅や高さといった情報を取得するGetDesc関数を作成します。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 D3DSURFACE_DESC GetDesc() = 0;
```

(３)CSpriteクラスとCNullSpriteクラスにGetDesc関数を追加します。この関数のプロトタイプを(２)をもとに作成し、適切な場所に追加しましょう。

(４)CSpriteクラスにGetDesc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。

```
/*****************************************************************************************************/
/*                                          スプライト記述取得                                         */
/*****************************************************************************************************/
D3DSURFACE_DESC CSprite::GetDesc()
{
    // テクスチャ記述取得
    D3DSURFACE_DESC    desc;
    m_pD3DTexture->GetLevelDesc(0, &desc);

    return desc;
}
```

(５)CNullSpriteクラスにGetDesc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。

```
/*****************************************************************************************************/
/*                                          スプライト記述取得                                         */
/*****************************************************************************************************/
D3DSURFACE_DESC CNullSprite::GetDesc()
{
    D3DSURFACE_DESC    desc;
    ::ZeroMemory(&desc, sizeof(desc));

    return desc;
}
```

(６)各ヘッダファイルで必要となる情報をまとめたヘッダファイルを作成します。ヘッダファイル(DXGTypes.h)を以下のように作成しましょう。

－DXGTypes.h－

```
/*
==========================================================================================
                                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】
     DXGTypes.h
                DirectX Graphics型定義

  【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>

/*****************************************************************************************/
/*                                          型定義                                       */
/*****************************************************************************************/
typedef D3DXVECTOR3  DXVEC3;

/*****************************************************************************************/
/*                                        構造体定義                                     */
/*****************************************************************************************/
// ロック詳細記述
struct DXGLOCKED_DESC {
    D3DFORMAT   Format;     // フォーマット
    UINT        Width;      // 幅
    UINT        Height;     // 高さ
    int         Pitch;      // ピッチ
    void*       pBits;      // イメージへのポインタ
};
```

DXGLOCKED_DESC構造体は、BackBuffer.hppとSprite.hppで必要となります。このような場合、どちらかのヘッダファイルで定義してしまうと、ヘッダ間の依存関係が複雑になり、エラーが多発する場合があります。両方のヘッダで定義すると、多重定義エラーになります。
これらの問題を解決するために、共通部分を抽出し、上記のようなヘッダファイルを作成します。

(７)バックバッファクラスに、サーフェスをロックしてイメージへのポインタを取得する機能とロックを解除する機能を追加します。以下のプロトタイプを適切な場所に追加しましょう。

```
DXGLOCKED_DESC Lock();
void Unlock();
```

DXGLOCKED_DESC構造体はDXGTypes.hで定義されていることに注意しましょう。

(８)バックバッファクラスに、サーフェスをロックしてイメージへのポインタを取得するLock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。

```
/*****************************************************************************************************/
/*                                              ロック                                               */
/*****************************************************************************************************/
DXGLOCKED_DESC CBackBuffer::Lock()
{
    // サーフェス記述取得
    D3DSURFACE_DESC   surface;
    m_pD3DSurface->???????(&surface);

    // ロック
    D3DLOCKED_RECT   lock;
    m_pD3DSurface->????????(&lock, NULL, 0);

    // ロック記述設定
    DXGLOCKED_DESC   desc;
    desc.Format = surface.??????;
    desc.Width  = surface.?????;
    desc.Height = surface.??????;
    desc.Pitch  = lock   .?????;
    desc.pBits  = lock   .?????;

    return desc;
}
```

サーフェスをロックしたときに得られる情報は、サーフェスのピッチ(次の行までのバイト数)とサーフェスメモリへのポインタだけで、幅や高さといった情報は取得できません。そこで、ロック前にサーフェス記述で取得し、ロック情報と合わせてDXGLOCKED_DESC構造体にまとめて返すようにしています。

(９)バックバッファクラスに、ロックを解除するUnlock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。

```
/*****************************************************************************************************/
/*                                           ロック解除                                              */
/*****************************************************************************************************/
void CBackBuffer::Unlock()
{
    m_pD3DSurface->??????????();
}
```

(10)バックバッファクラスに、テクスチャをロックしてイメージへのポインタを取得するLock関数とロックを解除するUnlock関数を追加します。
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 DXGLOCKED_DESC Lock() = 0;
virtual void Unlock() = 0;
```

(11)CSpriteクラスに、Lock関数とUnlock関数を追加します。この関数のプロトタイプを(10)をもとに作成し、適切な場所に追加しましょう。

(12)CSpriteクラスに、Lock関数とUnlock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。

```
/*****************************************************************************************************/
/*                                              ロック                                               */
/*****************************************************************************************************/
DXGLOCKED_DESC CSprite::Lock()
{
    // テクスチャ記述取得
    D3DSURFACE_DESC   surface;
    m_pD3DTexture->???????????(0, &surface);

    // ロック
    D3DLOCKED_RECT   lock;
    m_pD3DTexture->????????(0, &lock, NULL, 0);
```

```
    // スプライト情報設定
    DXGLOCKED_DESC   desc;
    desc.Format = surface.??????;
    desc.Width  = surface.?????;
    desc.Height = surface.??????;
    desc.Pitch  = lock   .?????;
    desc.pBits  = lock   .?????;

    return desc;
}

/*****************************************************************************************************/
/*                                          ロック解除                                               */
/*****************************************************************************************************/
void CSprite::Unlock()
{
    m_pD3DTexture->??????????(0);
}
```

(11)CNullSpriteクラスに、Lock関数とUnlock関数を追加します。この関数のプロトタイプを(10)をもとに作成し、適切な場所に追加しましょう。

(12)CNullSpriteクラスに、Lock関数とUnlock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。

```
/*****************************************************************************************************/
/*                                          ロック                                                   */
/*****************************************************************************************************/
DXGLOCKED_DESC CNullSprite::Lock()
{
    DXGLOCKED_DESC   desc;
    ::ZeroMemory(&desc, sizeof(desc));

    return desc;
}

/*****************************************************************************************************/
/*                                          ロック解除                                               */
/*****************************************************************************************************/
void CNullSprite::Unlock()
{
}
```