ESライブラリ＆＆
ゲームプログラミング

○　ピクセルシェーダー編 - 第６回　ぼかし　○

ぼかし

・ぼかしは、ピントがずれたようなぼやけた画像にするエフェクト
・転送先の領域より小さい領域にある画像を拡大して転送したり、ピクセルシェーダーで実現
・ぼかしは、単体で使うよりも被写界深度やブルームなど、他のエフェクトと組み合わせて使われる

概要

ぼかしは、画像をぼやけさせて不鮮明にする効果です。ピントをずらして撮影した写真のようになります。ぼかし単体で使うことは少なく、被写界深度やブルームのようなエフェクトと組み合わせたり、レンダリング画像に合成するなど、ほかのエフェクトや画像と組み合わせて使います。

ぼかしは、スケーリングよるものとピクセルシェーダーによるものがあります。

スケーリングによるぼかし

低解像度の画像を高解像度の領域へ拡大して転送するとき、バイリニアなどの補間法を用いると、ぼやけた画像になります。これらのエフェクトは、周辺のピクセルを参照して補間するためです。解像度が低ければ、それだけ補間されるピクセルに影響が強くなるため、ぼやけ具合が大きくなります。ただし、低解像度の画像はもとの画像に比べ情報量が減ってしまいます。

ピクセルシェーダーによるぼかし

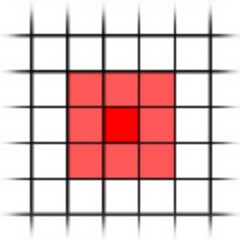
ピクセルシェーダーでぼかすには、描画しようとしているピクセルと、その周辺ピクセルを任意の割合で混合したものを出力するようにします。そうすれば、周辺のピクセルが混じった色になり、ぼやけて見えるという仕組みです。中心ピクセルの割合を低くしたり、周辺ピクセルの範囲を広げれば、ぼやけ具合が大きくなります。

問題となるのが、隣のピクセルを参照する方法です。ピクセルシェーダーではほかのピクセルの参照はできません(そもそも描画自体されていない可能性があります)。しかし、ＵＶ座標の操作により、ほかの(描画されるであろう)テクスチャのピクセル＝テクセルの参照はできます。ＵＶ座標は、どんな解像度でも0.0～1.0の範囲のため、単に±１や±0.1をしたからといって、隣のピクセルを参照できるわけではありません。テクスチャの解像度により、隣のピクセルまでの増減値が変わります。

さらに、ピクセルシェーダーでは、テクスチャの解像度を取得することはできません。よって、隣のピクセルまでの増減値も計算できません。これは、プログラム側で計算し、グローバル変数経由で受け渡しを行います。この増減値をもとに、ＵＶ座標を増減させて周辺のテクセルを読み取り、あらかじめ設定した割合で混合して出力すれば、ぼかし処理を行うことができます。

例：ＵＶ座標の増減値

256	×	256の場合	→	1	÷ 256	＝ 0.00390625　Ｕ，Ｖともに±0.00390625　すると隣
1024	×	1024の場合	→	1	÷ 1024	＝ 0.0009765625　Ｕ，Ｖともに±0.0009765625すると隣
2048	×	512の場合	→	1	÷ 2048	＝ 0.00048828125　Ｕは±0.00048828125,
				1	÷ 512	＝ 0.001953125　Ｖは±0.001953125すると隣のピクセル



0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

中心のピクセル0.2
周辺のピクセル0.1で混合
合計が1.0になるようにする
0.2 ＋ 0.1 × 8 ＝ 1.0

課　　題

ピクセルシェーダーを使って画面をぼかしてみましょう。

(１)ぼかしエフェクトを作成します。次のプログラムを"Blur.fx"として作成し、プロジェクトの"Shader"フォルダに保存しましょう。

－Blur.fx－

```
//--------------------------------------------------------------------------------
// File: Blur.fx
//
// The effect file for the Basic Blur HLSL sample.
//--------------------------------------------------------------------------------

//--------------------------------------------------------------------------------
// Global variables
//--------------------------------------------------------------------------------
sampler tex0 : register(s0);

float   AddU;
float   AddV;

//--------------------------------------------------------------------------------
// pass0 PixelShader Main Function
//--------------------------------------------------------------------------------
float4 PS_P0_Main(float2 UV : TEXCOORD0) : COLOR0
{
    float4   txl0 = tex2D( tex0, UV + float2( 0.0f,  0.0f) ) * 0.2f;

    float4   txl1 = tex2D( tex0, UV + float2(-AddU, -AddV) ) * 0.1f;
    float4   txl2 = tex2D( tex0, UV + float2( 0.0f, -AddV) ) * 0.1f;
    float4   txl3 = tex2D( tex0, UV + float2(+AddU, -AddV) ) * 0.1f;

    float4   txl4 = tex2D( tex0, UV + float2(-AddU,  0.0f) ) * 0.1f;
    float4   txl5 = tex2D( tex0, UV + float2(+AddU,  0.0f) ) * 0.1f;

    float4   txl6 = tex2D( tex0, UV + float2(-AddU, +AddV) ) * 0.1f;
    float4   txl7 = tex2D( tex0, UV + float2( 0.0f, +AddV) ) * 0.1f;
    float4   txl8 = tex2D( tex0, UV + float2(+AddU, +AddV) ) * 0.1f;

    float4   color = txl0 + txl1 + txl2 + txl3 + txl4 + txl5 + txl6 + txl7 + txl8;

    return color;
}

//--------------------------------------------------------------------------------
// Techniques
//--------------------------------------------------------------------------------
technique Blur
{
    pass P0
    {
        VertexShader = NULL;
        PixelShader  = compile ps_2_0 PS_P0_Main();
    }
}
```

ピクセルシェーダーに渡されてくるＵＶ座標とグローバル変数AddU, AddVをもとに周辺のピクセル値を読み取り、中心ピクセル0.2, 周辺ピクセル0,1の割合で混合して出力しています。

(２)シェーダーのグローバル変数に適切な値をプログラム側で設定し、ぼかしをかけてみましょう。
ヒント１　RenderTargetクラスのGetTextureDesc関数でテクスチャのサイズを取得
ヒント２　上記の戻り値はTextureDescription型(D3DSURFACE_DESC型)
ヒント３　上記構造体のメンバにWidthとHeightがある

(３)(２)で設定したグローバル変数AddU, AddVに定数を掛け、値を大きくするとよりぼやけるようになります。AddU, AddVの値を増減し、ぼやけ具合がどのように変わるか確認しましょう。