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７回　フィルター　○

フィルター

・フィルターは、不要な情報を取り除き、目的とする情報のみ取り出す処理
・２×２～１６×１６ピクセルの領域を読み取り処理を行う
・ぼかし、シャープネス、エンボス、エッジ抽出などがある

概要

フィルターは、不要な情報を取り除き、目的とする情報を取り出す処理です。Photoshopなどのフォトレタッチソフトに搭載されています。ぼかし処理、シャープネス、エンボス、エッジ抽出といったものが基本的なフィルターです。

内部的には、基点と周辺のピクセルを取得し、重み付けを施したピクセルを出力することで行っています。基本部分を作成すれば、重み付けの部分を変更するだけでいくつものフィルターを生み出すことができます。

平滑化

平滑化(Smoothing)とは、ランダムなノイズを除去したり、濃度の細かい変化を少なくし、滑らかな画像にする処理のことです。ピクセル値の変動を一様に平滑化するため、輪郭がぼやけてしまうという欠点があります。中心からの距離により重みを変えたり(ガウシアン)、輪郭や線などの大きな濃度変化を保存しながら微少変動のみを平滑化するという方法もあります。重み付けは以下のようになります。

0	0	0
0	1	0
0	0	0

▲元画像

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

▲平滑(平均)化フィルター

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

▲ガウシアンフィルター

鮮鋭化(シャープネス，アンシャープマスキング)

鮮鋭化とは、輪郭を際だたせる処理です。画像のピクセル値が本来は急変しているべき輪郭の部分などで値の変化が緩やかになっている場合、ぼやけた画像に見えます。このような画像は、値の変化を強調(その部分を微分)することで鮮明にすることができます。
基点ピクセルに大きな重み付けを行い、周辺ピクセルを引くことで、基点が周辺と比較して異常な画素であった場合により強く残るようにしたものです。各ピクセルの重みは、上のフィルタを使って求まります。「元画像＋(元画像－平滑化)」とすることにより、元画像からぼやけ成分を除去しています。

0	-1	0
-1	5	-1
0	-1	0

▲４近傍鮮鋭化

-1	-1	-1
-1	9	-1
-1	-1	-1

▲８近傍鮮鋭化

エンボス

エンボスとは凹凸加工のことで、画像の陰影部を強調し白黒画像化させます。斜め方向のエッジだけを抽出したものにピクセルの取りうる値の中央値(0.0～1.0なら0.5､0～255なら128)を足したものです。重み付けを行うピクセルの距離を変更することで、エンボスの深度を変えることができます。

-1	0	0
0	1	0
0	0	0

ピクセルシェーダーでは、上記で求めたピクセル値のRGB成分に+0.5するとエンボスになります。この方法では完全に白黒化されないため、さらに以下のようにRGB成分の平均値を求めると、モノクロ化され、見栄えがよくなります。

```
color.rgb = (color.r + color.g + color.b) / 3.0f;
```

エッジ抽出

エッジ(edge)とは、周辺のピクセルに比べ急激に値が変化する部分のことです。エッジは輪郭の場合が多く、重要なピクセルといえます。エッジの検出は、特定オブジェクトの抜き出し、２つの画像間の対応点算出、さらに複雑な画像処理のための前処理などに用いられる基本的な画像処理のひとつです。

基点ピクセルに大きな重み付けを行い、周辺ピクセルを引くことで、基点が周辺と比較して異常な画素であった場合だけ強く値が残るようにしています。さらにRGB成分の平均値をとってモノクロ化すると、よりエッジがわかりやすくなります。

0	-1	0
-1	4	-1
0	-1	0

４近傍

-1	-1	-1
-1	8	-1
-1	-1	-1

８近傍

▲Laplacian(ラプラシアン)フィルター

エッジ抽出には、ｘ軸方向とｙ軸方向で異なる重みを用い、それぞれの合計を２乗して平方根を求める方法もあります。

$$C=\sqrt{C_x^2+C_y^2}$$

-1	0	1
-2	0	2
-1	0	1

ｘ軸方向

-1	-2	-1
0	0	0
1	2	1

ｙ軸方向

▲Sobel(ソーベル)フィルター

さらに、重みの付け方により、Gradient(微分)、Roberts(ロバーツ)、Prewitt(プレヴィット)フィルターといったものがあります。

0	0	0
0	1	-1
0	0	0

ｘ軸方向

0	0	0
0	0	1
0	-1	0

ｙ軸方向

▲Gradientフィルター

0	0	0
0	1	0
0	0	-1

ｘ軸方向

0	0	0
0	0	1
0	-1	0

ｙ軸方向

▲Robertsフィルター

-1	0	1
-1	0	1
-1	0	1

ｘ軸方向

-1	-1	-1
0	0	0
1	1	1

ｙ軸方向

▲Prewittフィルター

1	-2	1
2	-4	2
1	-2	1

ｘ軸方向

1	2	1
-2	-4	-2
1	2	1

ｙ軸方向

▲Qperフィルター

課　　題

(１)ピクセルシェーダーで平滑化フィルターとガウシアンフィルターを作成しましょう。

(２)ピクセルシェーダーで鮮鋭化フィルター(４近傍および８近傍)を作成しましょう。

(３)ピクセルシェーダーでエンボスフィルターを作成しましょう。

(４)ピクセルシェーダーでLaplacianフィルター(４近傍および８近傍)を作成しましょう。

(５)以下は、Sobelフィルターでエッジ抽出を行うシェーダープログラムです。

－Sobel.fx－

```
//-----------------------------------------------------------------------------
// File: Sobel.fx
//
// The effect file for the Sobel Filter HLSL sample.
//-----------------------------------------------------------------------------

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

float   AddU;
float   AddV;

//-----------------------------------------------------------------------------
// pass0 PixelShader Main Function
//-----------------------------------------------------------------------------
float4 PS_P0_Main(float2 UV : TEXCOORD0) : COLOR0
{
    // pixel
    float4   p0 = tex2D( tex0, UV );

    float4   p1 = tex2D( tex0, UV + float2(-AddU, -AddV) );
    float4   p2 = tex2D( tex0, UV + float2( 0.0f, -AddV) );
    float4   p3 = tex2D( tex0, UV + float2(+AddU, -AddV) );

    float4   p4 = tex2D( tex0, UV + float2(-AddU,  0.0f) );
//  float4   p5 = tex2D( tex0, UV + float2( 0.0f,  0.0f) );
    float4   p6 = tex2D( tex0, UV + float2(+AddU,  0.0f) );

    float4   p7 = tex2D( tex0, UV + float2(-AddU, +AddV) );
    float4   p8 = tex2D( tex0, UV + float2( 0.0f, +AddV) );
    float4   p9 = tex2D( tex0, UV + float2(+AddU, +AddV) );

    // x-axsis
    float4   x0 = p0 *  0.0f;

    float4   x1 = p1 * -1.0f;
    float4   x2 = p2 *  0.0f;
    float4   x3 = p3 *  1.0f;

    float4   x4 = p4 * -2.0f;
    float4   x6 = p6 *  2.0f;

    float4   x7 = p7 * -1.0f;
    float4   x8 = p8 *  0.0f;
    float4   x9 = p9 *  1.0f;

    // y-axsis
    float4   y0 = p0 *  0.0f;

    float4   y1 = p1 * -1.0f;
    float4   y2 = p2 * -2.0f;
    float4   y3 = p3 * -1.0f;
```

```
    float4   y4 = p4 *  0.0f;
    float4   y6 = p6 *  0.0f;

    float4   y7 = p7 *  1.0f;
    float4   y8 = p8 *  2.0f;
    float4   y9 = p9 *  1.0f;

    // variance
    float4   cx = x0 + x1 + x2 + x3 + x4 + x6 + x7 + x8 + x9;
    float4   cy = y0 + y1 + y2 + y3 + y4 + y6 + y7 + y8 + y9;

    // pixel-color
    float4   color;
    color     = sqrt(cx * cx + cy * cy);
//  color.rgb = (color.r + color.g + color.b) / 3.0f;
    color.a   = 1.0f;

    return color;
}

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

プログラムを実行し、結果を確認しましょう。

(６)(５)を参考に、Gradientフィルターを作成しましょう。

(７)Robertsフィルターを作成しましょう。

(８)Prewittフィルターを作成しましょう。

(９)Qperフィルターを作成しましょう。