ESライブラリ＆＆
ゲームプログラミング

○　シェーダー共通編 - 第２回　ＨＬＳＬの書き方　○

全体図

基本的なHLSLファイルは、以下のようになっています。

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

// VertexShader Output Structure
struct VS_OUTPUT
{
   float4 Pos   : POSITION;
   float2 Tex   : TEXCOORD0;
};

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

//-----------------------------------------------------------------------------
// pass0 VertexShader Main Function
//-----------------------------------------------------------------------------
VS_OUTPUT VS_P0_Main(float4 Pos : POSITION,
                     float4 Col : COLOR0,
                     float2 Tex : TEXCOORD0)
{
   VS_OUTPUT Out;

   Out.Pos = Pos;
   Out.Tex = Tex;

   return Out;
}

//-----------------------------------------------------------------------------
// pass0 PixelShader Main Function
//-----------------------------------------------------------------------------
float4 PS_P0_Main(VS_OUTPUT In) : COLOR0
{
   float4 Color = tex2D(tex0, In.Tex);

   Color.rgb = 1.0f - Color.rgb;

   return Color;
}

//-----------------------------------------------------------------------------
// Techniques
//-----------------------------------------------------------------------------
technique Inverse
{
   pass P0
   {
      VertexShader = compile vs_1_1 VS_P0_Main();
      PixelShader  = compile ps_1_1 PS_P0_Main();
   }
}
```

大まかにはC／C++とほとんど同じで、以下のような構成をしています。

構造体の定義
グローバル変数の宣言
関数の定義

テクニックの定義
{
　パスの定義
　パスの定義
　　…
}
 	
テクニックの定義
　　…

コメントには日本語も使用できますが、DirectX SDKに付属していた"EffectEdit.exe"は、日本語(２バイト文字)でエラーが出るため、サンプルでは使用しないようにしています。EffectEditを使わないのであれば、日本語コメントも問題なく使用できます。

テクニックの定義

テクニックとは、レンダリング方法を記述したパスを複数まとめたものです。テクニックは、以下のように定義します。

```
technique テクニック名
{
    パスの定義
    パスの定義
        …
}
```

１つのファイルに複数のテクニックを定義することができます。また、１つのテクニック内に、複数のパスを定義することができます。プログラム側からは、エフェクトクラス(インタフェース)のBegin～End関数によって、テクニック単位で呼び出されます(読み込みはファイル単位、呼び出しはテクニック単位、実行はパス単位)。Begin関数の呼び出し前に、テクニックの文字列もしくはテクニックへのハンドルで呼び出すテクニックの指定ができるので、状況に合わせたものを呼び出すことができます。

パスの定義

パスとは、バーテックスシェーダーとピクセルシェーダーを１つずつペアでまとめたものです。パスは、以下のように定義します。

```
pass パス名
{
    VertexShader = compile vs_バージョン 関数名();
    PixelShader  = compile ps_バージョン 関数名();
}
```

バーテックスシェーダーとピクセルシェーダーが最初に実行する関数とバージョンを指定します。使用しないシェーダーにはNULLを指定できます。パスの定義とは、シェーダーのメイン関数名とバージョンを設定するもの、ともいえます。

キーワード"complie"の後に、バージョンの指定が必要です。バージョンによって実行できる関数などが異なります。バージョンを低くすると多くの環境で動作するようになり、バージョンを高くすると高性能になる代わりに、最新の環境でなければ動作しなくなります。

バージョンには以下の文字列を指定します(カッコ内はDirectX10以降)。

```
バーテックスシェーダー vs_1_1                  vs_2_0 vs_2_x vs_3_0 (vs_4_0 vs_4_1 vs_5_0)
ピクセルシェーダー     ps_1_1～ps_1_3 ps_1_4 ps_2_0 ps_2_x ps_3_0 (ps_4_0 ps_4_1 ps_5_0)
```

パスは、１つのテクニック内に、いくつでも定義できます。プログラム側からBeginPass関数呼び出し時に、パスの指定ができるので、同じテクニック内であれば任意のパスを実行することができます。パスを１つだけ実行したり、同じパスを連続で実行したり、複数のパスを組み合わせてレンダリングができます。
パスから戻り値をプログラム側に返すことはできませんが、HLSL内で宣言されたグローバル変数はプログラム側からも読み書きできるため、これを参照することにってシェーダーの実行結果をもとに次に実行すべきパスを切り換える、ということもできます。

関数の定義

HLSLでは、C／C++と同様に、関数を定義し呼び出すことができます。定義方法もまったく同じです。

```
戻り値の型　関数名(引数の型　変数名 [ : セマンティクス],
                    引数の型　変数名 [ : セマンティクス], … ) [: 戻り値のセマンティクス]
{
    (関数の処理)
        ⋮
    return 戻り値;
}
```

関数内で別の関数を呼び出すことや、別ファイルで定義されている関数をインクルードして使うこともできます。

型

HLSLでは、C／C++と同様に、関数内のローカル変数やシェーダー全体で参照できるグローバル変数、構造体、配列を宣言できます。宣言方法も同じで、以下のように行います。

```
型名　変数名;
```

HLSLでは、変数の型として以下のものを使用できます。

```
bool   - true または false
int    - 32 ビット符号付き整数
uint   - 32 ビット符号なし整数
float  - 32 ビット浮動小数点値
double - 64 ビット浮動小数点値
```

ベクトルは、vectorもしくは型名のあとに数字をつけることによって表現します。４次元ベクトルまでサポートしているので、数字は１～４の範囲で指定します。

```
int2     pos2d;              // int    型の２次元ベクトル
float3   pos3d;              // float 型の３次元ベクトル

vector <double, 4>    vec4; // double型の４次元ベクトル
```

各メンバへのアクセスは「x, y, z, w」もしくは「r, g, b, a」を組み合わせることにより行います。

```
pos2d.xy  pos3d.xyz  pos3d.xz  pos3d.z  pos3d.rgb  pos3d.grb
```

ｘ～ｗもしくはｒ～ａは自由に組み合わせられるので、上記のような表現ができます。

行列は、１×１から４×４まで定義できます。行数と列数が異なる行列も扱えます。matrixもしくは型名のあとに「行数x列数」で宣言します。

```
int3x2     mat1;             // int    型３行２列の行列
float4x4   mat2;             // float 型４行４列の行列

matrix <double, 3, 3> mat3; // double型３行３列の行列
```

行列の各メンバへのアクセスは、以下のメンバ名を指定することにより行えます。

```
_m00, _m01, _m02, _m03
_m10, _m11, _m12, _m13
_m20, _m21, _m22, _m23
_m30, _m31, _m32, _m33
```

または

_11，_12，_13，_14
_21，_22，_23，_24
_31，_32，_33，_34
_41，_42，_43，_44

制御文

HLSLでは、プログラムの流れを制御する「フロー制御文」も使用できます。DirectX9.0では、if-else，for，while，do-while文を使うことができます。記述方法もC／C++とまったく同じです。ただし、brake，countinue，(switch-case文)は、DirectX10以降でなければ使用できません。

セマンティクス

シェーダーが最初に実行する関数の引数と構造体のメンバに、変数の使用目的を示した「セマンティクス」という文字列を指定できます。

関数が呼び出される際、頂点またはピクセルの情報は、該当するセマンティクスが付いた変数に渡されます。これにより、頂点またはピクセル情報の中から必要なものだけを受け取ることができます。なお、頂点にもピクセル情報にも含まれていないセマンティクスが付いている変数は、0が格納されます。また、シェーダーが最初に実行する関数の引数すべてに、セマンティクスを指定する必要があります。

セマンティクスは、バーテックスシェーダーとピクセルシェーダーで指定できる文字列が異なり、それぞれ以下のものが指定できます。

・バーテックスシェーダー

入力	説明	型
BINORMAL[n]	従法線	float4
BLENDINDICES[n]	ブレンド インデックス	uint
BLENDWEIGHT[n]	ブレンドの重み	float
COLOR[n]	ディフューズ カラーとスペキュラ カラー	float4
NORMAL[n]	法線ベクトル	float4
POSITION[n]	オブジェクト空間内の頂点位置。	float4
POSITIONT	トランスフォームされた頂点位置。	float4
PSIZE[n]	ポイントサイズ	float
TANGENT[n]	接線	float4
TEXCOORD[n]	テクスチャー座標	float4

出力	説明	型
COLOR[n]	ディフューズ カラーとスペキュラ カラー	float4
FOG	頂点フォグ	float
POSITION[n]	頂点の位置	float4
PSIZE	ポイントサイズ	float
TESSFACTOR[n]	テッセレーション係数	float
TEXCOORD[n]	テクスチャー座標	float4

・ピクセルシェーダー

入力	説明	型
COLOR[n]	ディフューズ カラーとスペキュラ カラー	float4
TEXCOORD[n]	テクスチャー座標	float4
VFACE	背面のプリミティブを示す浮動小数点スカラー	float
VPOS	現在のピクセル (x,y) 位置を含みます。	float2

出力	説明	型
COLOR[n]	出力カラー	float4
DEPTH[n]	出力深度	float

※[n]には0以上の数字が入ります。最大数は環境によって異なります。
例：COLOR0，TEXCOORD2

組み込み関数一覧(抜粋)

関数	入力の型	出力の型	vs	ps	説明
abs(x)	float, int, vector, matrix	入力と同じ	1.1	1.4	絶対値 (成分ごと)。
acos(x)	float, vector, matrix	入力と同じ	1.1	2.0	x の各成分のアークコサインを返します。
all(x)	float, int, bool, vector, matrix	bool	1.1	1.4	x のすべての成分が 0 以外の値であるかどうかをテストします。
any(x)	float, int, bool, vector, matrix	bool	1.1	1.4	x のいずれかの成分が 0 以外の値であるかどうかをテストします。
asin(x)	float, vector, matrix	入力と同じ	1.1	2.0	x の各成分のアークサインを返します。
atan(x)	float, vector, matrix	入力と同じ	1.1	2.0	x のアークタンジェントを返します。
atan2(y, x)	float, vector, matrix	入力と同じ	1.1	2.0	2 つの値 (x,y) のアークタンジェントを返します。
ceil(x)	float, vector, matrix	入力と同じ	1.1	2.0	x 以上の最小の整数を返します。
clamp(x, min, max)	float, int, vector, matrix	入力と同じ	1.1	1.4	x を [min, max] の範囲にクランプします。
clip(x)	float, vector, matrix	なし	×	1.1	x のいずれかの成分が 0 未満の場合、現在のピクセルを破棄します。
cos(x)	float, vector, matrix	入力と同じ	1.1	2.0	x のコサインを返します。
cosh(x)	float, vector, matrix	入力と同じ	1.1	2.0	x のハイパーボリック コサインを返します。
cross(x, y)	floatを成分とするvector	入力と同じ	1.1	1.4	2 つの 3D ベクトルの外積を返します。
ddx(x)	float, vector, matrix	入力と同じ	×	2.x	スクリーン空間の x 座標に対する、x の偏微分を返します。
ddy(x)	float, vector, matrix	入力と同じ	×	2.x	スクリーン空間の y 座標に対する、x の偏微分を返します。
degrees(x)	float, vector, matrix	入力と同じ	1.1	2.0	x をラジアン単位から度単位に変換します。
determinant(m)	floatを成分とするvector	float	1.1	1.4	正方行列 m の行列式を返します。
distance(x, y)	floatを成分とするvector	入力と同じ	1.1	2.0	2 点間の距離を返します。
dot(x, y)	float, intを成分とするvector	float, int	1.1	1.4	2 つのベクトルの内積を返します。
exp(x)	float, vector, matrix	入力と同じ	1.1	2.0	e を底とする指数を返します。
exp2(x)	float, vector, matrix	入力と同じ	1.1	2.0	2 を底とする指数 (成分ごと)。
faceforward(n, i, ng)	floatを成分とするvector	入力と同じ	1.1	1.4	-n * sign(dot(i, ng)) を返します。
floor(x)	float, vector, matrix	入力と同じ	1.1	2.0	x 以下の最大の整数を返します。
fmod(x, y)	float, vector, matrix	入力と同じ	1.1	2.0	x/y の浮動小数点の剰余を返します。
frac(x)	float, vector, matrix	入力と同じ	1.1	2.0	x の小数部を返します。
frexp(x, exp)	float, vector, matrix	入力と同じ	3.0	2.x	x の仮数と指数を返します。
fwidth(x)	float, vector, matrix	入力と同じ	3.0	2.x	abs(ddx(x)) + abs(ddy(x)) を返します。
isfinite(x)	float, vector, matrix	bool	1.1	2.0	x が有限なら true 、それ以外は false を返します。
isinf(x)	float, vector, matrix	bool	1.1	2.0	x が ±INF なら true 、それ以外は false を返します。
isnan(x)	float, vector, matrix	bool	1.1	2.0	x が NAN または QNAN なら true 、それ以外は false を返します。
ldexp(x, exp)	float, vector, matrix	入力と同じ	1.1	2.0	x * 2exp を返します。
length(v)	float, vector, matrix	float	1.1	2.0	ベクトル v の長さを返します。
lerp(x, y, s)	float, vector, matrix	入力と同じ	1.1	1.1	x + s(y - x) を返します。
lit(n dot l, n dot h, m)	float	float4	1.1	2.0	ライティング係数ベクトルを返します。
log(x)	float, vector, matrix	入力と同じ	1.1	2.0	x の e を底とする対数を返します。
log10(x)	float, vector, matrix	入力と同じ	1.1	2.0	x の 10 を底とする対数を返します。
log2(x)	float, vector, matrix	入力と同じ	1.1	2.0	x の 2 を底とする対数を返します。
max(x, y)	float, int, vector, matrix	入力と同じ	1.1	1.4	x と y のうちの大きい方の値を選択します。
min(x, y)	float, int, vector, matrix	入力と同じ	1.1	1.4	x と y のうちの小さい方の値を選択します。
modf(x, out ip)	float, int, vector, matrix	入力と同じ	1.1	2.0	値 x を小数部と整数部に分割します。
mul(x, y)	float, int, vector, matrix	x, yによる	1.1	1.1	x と y の行列乗算を実行します。
noise(x)	floatを成分とするvector	float	2.0	2.0	パーリン ノイズ アルゴリズムを使用してランダムな値を生成します。
normalize(x)	floatを成分とするvector	入力と同じ	1.1	2.0	正規化されたベクトルを返します。
pow(x, y)	float, vector, matrix	入力と同じ	1.1	2.0	x の y 乗を返します。
radians(x)	float, vector, matrix	入力と同じ	1.1	1.1	x を度単位からラジアン単位に変換します。
reflect(i, n)	floatを成分とするvector	入力と同じ	1.1	1.1	反射ベクトルを返します。
refract(i, n, R)	floatを成分とするvector	入力と同じ	1.1	2.0	屈折ベクトルを返します。
round(x)	float, vector, matrix	float	1.1	2.0	x を最も近い整数に丸めます。
rsqrt(x)	float, vector, matrix	入力と同じ	1.1	2.0	1 / sqrt(x) を返します。
saturate(x)	float, vector, matrix	入力と同じ	1.1	1.1	x を [0, 1] の範囲にクランプします。
sign(x)	float, int, vector, matrix	int	1.1	1.4	x の符号を計算します。
sin(x)	float, vector, matrix	入力と同じ	1.1	2.0	x のサインを返します。
sincos(x, out s, out c)	float, vector, matrix	入力と同じ	1.1	2.0	x のサインおよびコサインを返します。
sinh(x)	float, vector, matrix	入力と同じ	1.1	2.0	x のハイパーボリック サインを返します。
smoothstep(min, max, x)	float, vector, matrix	入力と同じ	1.1	2.0	0 と 1 の間のスムーズなエルミート補間を返します。
sqrt(x)	float, vector, matrix	入力と同じ	1.1	2.0	平方根 (成分ごと)。
step(a, x)	float, vector, matrix	入力と同じ	1.1	2.0	(x >= a) ? 1 : 0 を返します。
tan(x)	float, vector, matrix	入力と同じ	1.1	2.0	x のタンジェントを返します。
tanh(x)	float, vector, matrix	入力と同じ	1.1	2.0	x のハイパーボリック タンジェントを返します。
tex2D(s, t)	s…サンプラ, t…テクスチャ座標	float4	×	1.1	2D テクスチャー ルックアップ。
tex2Dproj(s, t)	s…サンプラ, t…float4	float4	2.0	2.0	2D 射影除算テクスチャー ルックアップ。
texCUBE(s, t)	s…サンプラ, t…float3	float4	×	1.1	キューブ テクスチャー ルックアップ。
transpose(m)	matrix	matrix	1.1	1.1	行列 m の転置行列を返します。
trunc(x)	float, vector, matrix	入力と同じ	1.1	1.1	浮動小数点値の端数を切り捨てて整数値に変換します。

出典：http://msdn.microsoft.com/ja-jp/library/bb509611(v=vs.85).aspx