DxLib環境における頂点シェーダ

頂点シェーダの役割は、一言で座標変換です。

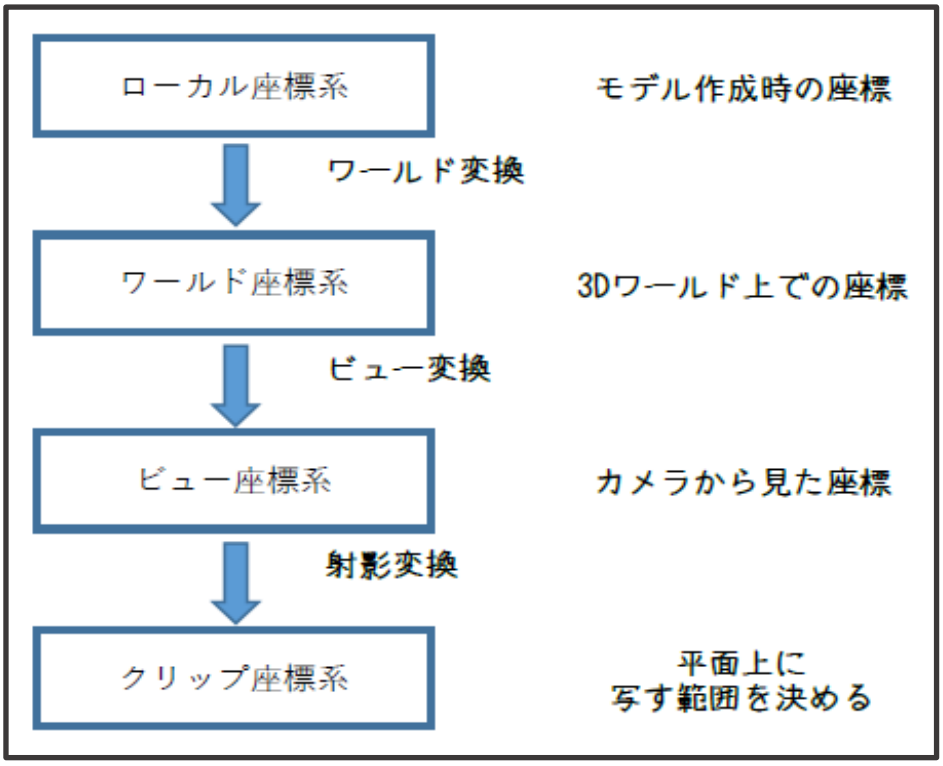
3DCannonの教材「01_３Ｄゲーム制作のはじめに」、
もしくは、下記サイトで勉強してください。

バーテックスシェーダによる座標系変換
https://tkengo.github.io/blog/2015/01/10/opengl-es-2-2d-knowledge-3/

3次元CGと座標変換
https://lab.sdm.keio.ac.jp/ogi/vr/step3.html

MVP行列による座標変換について
https://matcha-choco010.net/2018/08/30/mvp-matrix/

コンピューターグラフィックスＳ
http://www.cg.ces.kyutech.ac.jp/lecture/cg/cg08_transformation1_s.pdf



必要最低限の処理しか書いてない頂点シェーダ

```
// VS/PS共通
#include "../Common/VertexToPixelHeader.hlsli"

// IN
#include "../Common/Vertex/VertexInputType.hlsli"
#define VERTEX_INPUT DX_MVI_VERTEX_TYPE_NMAP_1FRAME

// OUT
#define VS_OUTPUT VertexToPixelLit
#include "../Common/Vertex/VertexShader3DHeader.hlsli"

VS_OUTPUT main(VS_INPUT VSInput)
{

    VS_OUTPUT ret;

    // 頂点座標変換 ++++++++++++++++++++++++++++++++++++++( 開始 )
    float4 lLocalPosition;
    float4 lWorldPosition;
    float4 lViewPosition;

    // float3 → float4
    lLocalPosition.xyz = VSInput.pos;
    lLocalPosition.w = 1.0f;

    // ローカル座標をワールド座標に変換(剛体)
    lWorldPosition.w = 1.0f;
    lWorldPosition.xyz = mul(lLocalPosition, g_base.localWorldMatrix);

    // ワールド座標をビュー座標に変換
    lViewPosition.w = 1.0f;
    lViewPosition.xyz = mul(lWorldPosition, g_base.viewMatrix);
    ret.vwPos.xyz = lViewPosition.xyz;

    // ビュー座標を射影座標に変換
    ret.svPos = mul(lViewPosition, g_base.projectionMatrix);
```

```
    // 頂点座標変換 ++++++++++++++++++++++++++++++++++++( 終了 )


    // 出力パラメータを返す
    return ret;

}
```

最終的に変換された情報が、ピクセルシェーダにも渡っていき、
特にスクリーン(ウィンドウ)座標を使用して、画面の色を最終的に
決めていきます。

共通ファイルの説明

■ VertexInputType.hlsli

頂点シェーダの入力を定義するファイル。main関数の引数で渡されてくる。
この定義はDxLibの仕様で決められており、モデルの種類によって、
入力定義が変わってくる。

① 1フレームの影響を受ける頂点
② 1～4フレームの影響を受ける頂点
③ 5～8フレームの影響を受ける頂点
④ 法線マップの情報が含まれる1フレームの影響を受ける頂点
⑤ 法線マップの情報が含まれる1～4フレームの影響を受ける頂点
⑥ 法線マップの情報が含まれる5～8フレームの影響を受ける頂点
⑦ DrawPolygon3DToShader系

他にもあると思いますので、該当する定義が見つかったら、
このファイルに追加していきましょう。(例：９フレーム以上)

モデルがどのタイプに属するかは、
　MV1GetTriangleListVertexType
上記関数で識別することができます。

使用する際には、以下のようにdefineで使用する定義を決めます
#define VERTEX_INPUT DX_MV1_VERTEX_TYPE_NMAP_1FRAME

■ VertexToPixelHeader.hlsli

頂点シェーダの出力を定義するファイル。
頂点シェーダの出力 = ピクセルシェーダの入力となるので、
定義内容や、セマンティクスは同じものを使う必要がある。
ifdef で、定義を切っていないので、VertexInput.hlsli のように
定数を使用して、必要な定義のみ絞った方が処理効率は良い。

■ VertexShader3DHeader.hlsli
CommonShader3DHeader.hlsli

頂点シェーダを使用するあたり、DxLib側から渡されている情報定義。
座標変換に使用する行列や、ライトなどの盛りだくさんの情報がある。

オリジナル頂点シェーダの使用方法

要領は、ピクセルシェーダと同じです。

■初期処理

```
// 頂点シェーダのロード
shaderVS_ = LoadVertexShader(
    (Application::PATH_SHADER + shaderFileNameVS).c_str());

// 頂点定数バッファの確保サイズ(FLOAT4をいくつ作るか)
constBufFloat4SizeVS_ = constBufFloat4SizeVS;

// 頂点シェーダー用の定数バッファを作成
constBufVS_ = CreateShaderConstantBuffer(
                                sizeof(FLOAT4) * constBufFloat4SizeVS);
```

■描画処理

```
// オリジナルシェーダ設定(ON)
MVISetUseOrigShader(true);

// 定数バッファの設定
UpdateShaderConstantBuffer(constBuf);

// 頂点シェーダー用の定数バッファを定数バッファレジスタにセット
SetShaderConstantBuffer(
    constBuf, DX_SHADERTYPE_VERTEX, CONSTANT_BUF_SLOT_BEGIN_VS);

// 頂点シェーダー設定
SetUseVertexShader(modelMaterial_.GetShaderVS());

// 描画
MVIDrawModel(modelId_);

// オリジナルシェーダ設定(OFF)
MVISetUseOrigShader(false);
```