UNIT21: SKINNED MESH - MULTIMESH

【学習要項】
☐Multi mesh
☐Global transform
☐Coordinate system transformation

【演習手順】
1．複数のメッシュを持つ FBX ファイルをロードし描画する
2．framework クラスの initialize メンバ関数で skinned_mesh コンストラクタ引数を.¥¥resources¥¥cube.003.0.fbx に変更する
※cube.003.fbx は正四面体の上に Suzanne を配置したモデルである
3．実行し正四面体の上に Suzanne が配置されずに重なっていることを確認する
※正しく表示するためにはシーン内の位置・姿勢・スケール情報（global_transform）を取得して描画に反映させることが必要である
①mesh 構造体にメンバ変数を追加する

```
DirectX::XMFLOAT4X4 default_global_transform{ 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1 };
```

②fetch_meshes メンバ関数の適当な場所に下記のコードを挿入する

```
mesh.default_global_transform = to_xmfloat4x4(fbx_mesh->GetNode()->EvaluateGlobalTransform());
```

③skinned_mesh.cpp に to_xmfloat4x4 関数、to_ xmfloat3 関数および to_ xmfloat4 関数を定義する

```
inline XMFLOAT4X4 to_xmfloat4x4(const FbxAMatrix& fbxamatrix)
{
    XMFLOAT4X4 xmfloat4x4;
    for (int row = 0; row < 4; ++row)
    {
        for (int column = 0; column < 4; ++column)
        {
            xmfloat4x4.m[row][column] = static_cast<float>(fbxamatrix[row][column]);
        }
    }
    return xmfloat4x4;
}
inline XMFLOAT3 to_xmfloat3(const FbxDouble3& fbxdouble3)
{
    XMFLOAT3 xmfloat3;
    xmfloat3.x = static_cast<float>(fbxdouble3[0]);
    xmfloat3.y = static_cast<float>(fbxdouble3[1]);
    xmfloat3.z = static_cast<float>(fbxdouble3[2]);
    return xmfloat3;
}
inline XMFLOAT4 to_xmfloat4(const FbxDouble4& fbxdouble4)
{
    XMFLOAT4 xmfloat4;
    xmfloat4.x = static_cast<float>(fbxdouble4[0]);
    xmfloat4.y = static_cast<float>(fbxdouble4[1]);
    xmfloat4.z = static_cast<float>(fbxdouble4[2]);
    xmfloat4.w = static_cast<float>(fbxdouble4[3]);
    return xmfloat4;
}
```

④skinned_mesh クラスの render メンバ関数で、定数バッファ構造体の world メンバ変数の計算を変更する

```
XMStoreFloat4x4(&data.world, XMLoadFloat4x4(&mesh.default_global_transform) * XMLoadFloat4x4(&world));
```

⑤実行し正四面体の上に Suzanne が配置されていることを確認する

4．ロードするファイルを cube.003.1.fbx に変更する
①赤黄青の３つの正四面体が縦に積まれたシーンが描画されるはずが正しく描画されない
※このファイルのメッシュデータは３角形化されていないので、ロード時に３角形化をおこなう
②framework クラスの render メンバ関数で座標系の変換を行う
※このファイルのシーンは右手系・Z 軸アップで記録されているので、左手系・Y 軸アップに変換する
※このファイルのシーンの単位はメートル記録されているので、スケールファクタは 1 にセットする

```
* 1:  const DirectX::XMFLOAT4X4 coordinate_system_transforms[]{
* 2:      { -1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1 },      // 0:RHS Y-UP
```

```
* 3:     { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1 },      // 1:LHS Y-UP
* 4:     { -1, 0, 0, 0, 0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 1 },    // 2:RHS Z-UP
* 5:     { 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1 },      // 3:LHS Z-UP
* 6: };
* 7: // To change the units from centimeters to meters, set 'scale_factor' to 0.01.
* 8: const float scale_factor = 1.0f;
* 9: DirectX::XMMATRIX C{ DirectX::XMLoadFloat4x4(&coordinate_system_transforms[2])
*10:      * DirectX::XMMatrixScaling(scale_factor, scale_factor, scale_factor) };
 11:
 12: DirectX::XMMATRIX S{ DirectX::XMMatrixScaling(1, 1, 1) };
 13: DirectX::XMMATRIX R{ DirectX::XMMatrixRotationRollPitchYaw(0, 0, 0) };
 14: DirectX::XMMATRIX T{ DirectX::XMMatrixTranslation(0, 0, 0) };
 15: DirectX::XMFLOAT4X4 world;
*16: DirectX::XMStoreFloat4x4(&world, C * S * R * T);
 17: skinned_meshes[0]->render(immediate_context.Get(), world, material_color);
```

③上記の方法だけでは面カリングが逆になってしまうのでラスタライザステートで制御する

```
 1: D3D11_RASTERIZER_DESC rasterizer_desc{};
 2: rasterizer_desc.FillMode = D3D11_FILL_SOLID;
 3: rasterizer_desc.CullMode = D3D11_CULL_BACK;
*4: rasterizer_desc.FrontCounterClockwise = TRUE;
 5:     :
 6:     :
 7:     :
```

５．これまで使用したすべてのモデルデータが正しく描画できていることも確認する
※座標系変換行列・ラスタライザステートはその都度変更する必要がある

【評価項目】
□複数メッシュを持つ FBX ファイルの描画
□ポリゴンの三角形化
□座標系変換