3Dモデル表示

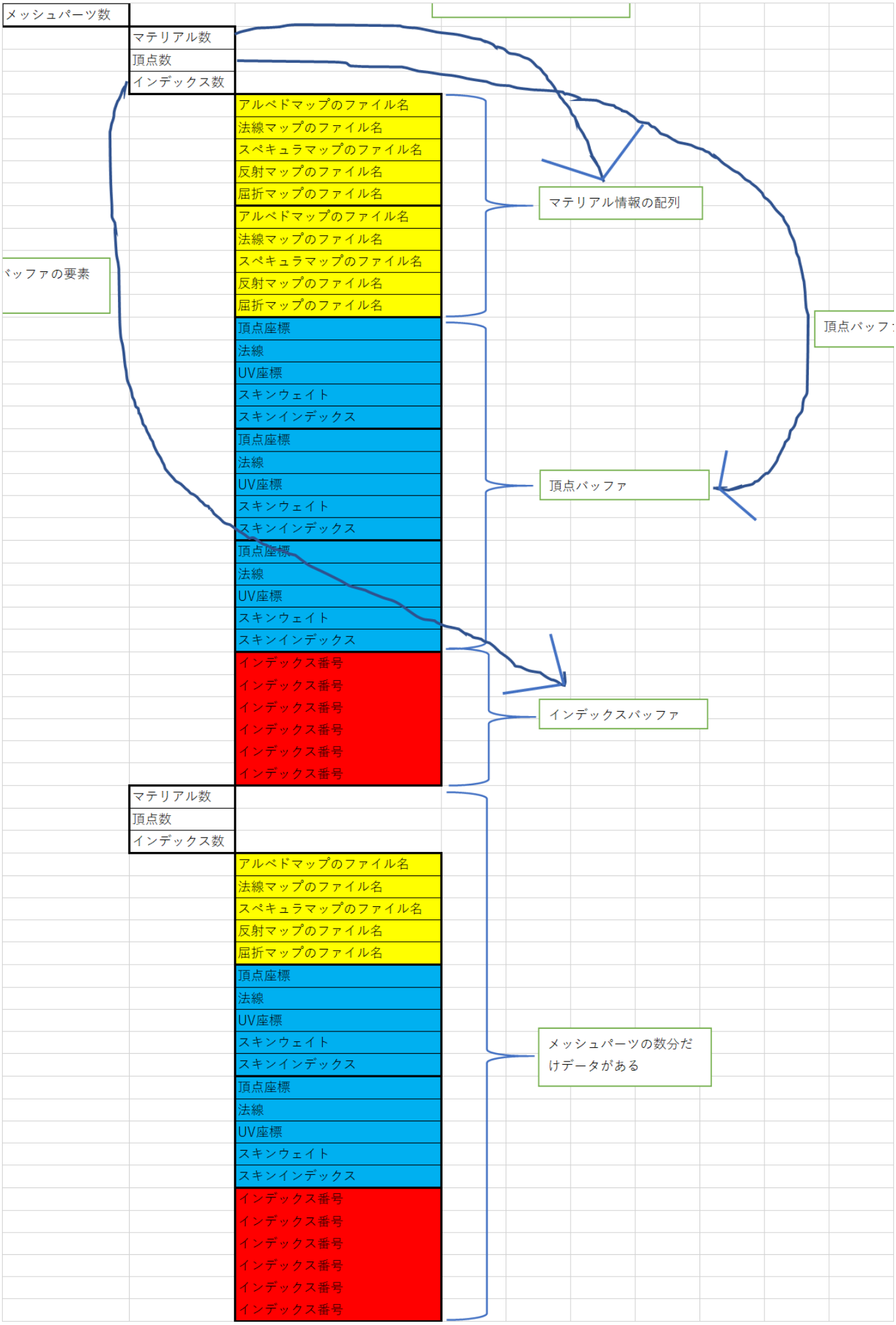
　Chapter2,3で単純な三角形ポリゴン表示について見てきました。単純な三角形ポリゴンを表示するためには、次の３つの情報が必要でした。

1. 頂点バッファ
2. インデックスバッファ
3. テクスチャ(マテリアル情報)

　実は複雑な3Dモデルの表示というのも、この三角形ポリゴン表示の知識で行うことができます。ようは複雑な3Dモデルというのは、三角形ポリゴンがたくさんあるだけです。本書では、これ以降独自のモデルフォーマットのtkmファイルというものを利用してモデル表示が行えるModelクラスを利用して話を進めていきます。

　tkmファイルというのは、3dsMaxやBlenderなどで、アーティストが作成したモデルデータから、頂点バッファ、インデックスバッファ、マテリアル情報を抽出したファイルです。次ページにtkmファイルのファイルフォーマットが記載されています。

tkmファイルのファイルフォーマット



これをプログラムで記述すると下記のような感じになります。

```
struct SMesh {
    SMaterial materials[3];          //マテリアルの配列。
    SVertex vertexBuffer[3];         //頂点バッファ。
    SIndexBuffer indexBuffer[3];     //インデックスバッファの配列。マテリアルの数分だけインデックスバッファはあるよ。
};
```

頂点バッファ、インデックスバッファ、マテリアル情報が入っているデータです。3Dモデルデータはパーツによってメッシュが分割されていることがあります（人型モデルであれば、髪の毛と体といった感じで分かれている場合があります）。なので、各メッシュごとに頂点バッファ、インデックスバッファ、マテリアル情報のデータを保持しています。

本書のサンプルプログラムでtkmファイルをロードする処理はMiniEngine/tkFile/TkmFile.h、TkmFile.cppに記載されています。tkmファイルをロードすることができたら、このデータを元にDirectX12のAPIを利用して、頂点バッファ、インデックスバッファ、テクスチャをグラフィックメモリ上に作成していきます。この処理を行っているのがMeshPartsクラスとなります。この処理はChapter2,3で勉強してきた内容となります。簡単にだけこれらを行っているソースの箇所を紹介します。

[MiniEngine/MeshParts.cpp(83行目)]

```
void MeshParts::CreateMeshFromTkmMesh(
    const TkmFile::SMesh& tkmMesh,
    int meshNo,
    const wchar_t* fxFilePath,
    const char* vsEntryPointFunc,
    const char* psEntryPointFunc)
{
     .
     .
     .
    省略
     .
     .
     .
}
```

コードの詳細は気にしなくて構いませんので、この関数内の次の３つのコメントに注目してください。

1. 頂点バッファを作成(90行目)
2. インデックスバッファを作成(108行目)
3. マテリアルを作成(137行目)

このように、難しいことをやっているように見えますが、本質的には三角形を描画するときと同等のことを行っています。
各種データを作成することができたら、後は毎フレーム、ドローコールを実行するだけです。これも

Chapter2,3で見てきた内容となります。これも簡単にだけソースの箇所を紹介しておきます。
[MiniEngine/MeshParts.cpp(159行目)]

```
void MeshParts::Draw(
    RenderContext& rc,
    const Matrix& mWorld,
    const Matrix& mView,
    const Matrix& mProj
)
{
     .
     .
     .
    省略
     .
     .
     .
}
```

こちらもコードの詳細は気にしなくて構いません。次の４つのコメントに注目してください。

1. 頂点バッファを設定。(187行目)
2. ディスクリプタヒープを登録。(193行目)
3. インデックスバッファを設定。(195行目)
4. ドローコールを実行。(199行目)

このように、毎フレームの描画処理も本質的にはポリゴンを表示する処理と全く同じことをしているだけとなります。
本書ではモデル表示処理の詳細は説明はしませんが、処理を追いかけてみると基礎的な知識の理解が深まると思います。最後に、本書のモデル表示関連のクラス図を記載しておきます。

【ハンズオン】3Dモデルを表示する処理を記述する。

step-1 3Dモデルをロードする。

[main.cpp]

```
//ロードするための初期化情報を作成。
ModelInitData initData;
//tkmファイルのファイルパス。
initData.m_tkmFilePath = "Assets/modelData/unityChan.tkm";
//使用するシェーダーのファイルパス。
initData.m_fxFilePath = "Assets/shader/NoAnimModel_Texture.fx";
//作成した初期化情報を使って、初期化する。
Model charaModel;
charaModel.Init(initData);
```

step-2 3Dモデルをドローする。

[main.cpp]

```
//step-2 3Dモデルをドローする。
charaModel.Draw(renderContext);
```

【ハンズオン】tkmファイルのロード処理を作成する。

【ハンズオン】一つのメッシュパーツをドローする処理を作成する。

【ハンズオン】全てのメッシュパートをドローする処理を作成する。

【ハンズオン】平行移動行列を作成して3Dモデルを動かす。

【ハンズオン】回転行列を作成して3Dモデルを回す。

【ハンズオン】拡大行列を作成して3Dモデルを拡大する。

【ハンズオン】拡大行列 × 回転行列 × 平行移動行列を計算して3Dモデルを動かす。