後期課題2

５日で理解する3Dプログラミング 「概論編」

■3D ゲームの仕組み

1） 3D ゲームを作る為には、3DCG の知識が必要不可欠となります。

パソコンなどで 3DCG を表現する為には「OpenGL」「DirectX」といった API を使用しますが、どちらも計算方式（レンダリング方式）は「ラスタライズ方式」と言います。これは、リアルタイムで高速に描画する為に、Zバッファという考え方で表現を行う方式となります。この方式は、高速性を優先する為に反面反射や屈折などの計算を簡易的に行う様にしています。

逆に、速度性を犠牲にして光の反射や屈折などに注力した方式を「レイトレーシング方式」と言います。ゲーム数学の授業でも取り組んでいると思いますが、いわゆるリアルな CG 表現を追求するものです。速度面で辛い部分があったのですが、最近ではハードウェアの性能向上により「リアルタイムレイトレーシング」が可能となったりしてきています。

とは言え、今回はDirectX(DxLib)を基準とするため「ラスタライズ方式」の 3DCG を扱っていきます。

2） 2D も 3D の仲間

3D のゲームイメージを考えると、「2D はどうなっているの？」と疑問に思う事もあるかもしれません。基本的には、2D も 3D 空間の一つと考えていきます。「3D 空間の中にある平面の板が配置しある」「描画面のスクリーンに張り付いているもの」という 2 種類の性質があるので、状況に合わせて合成していきます。

3）3D ゲームはどうなっているのか？

3D 表現では、モデルデータを 3D 空間に配置しただけでは思い通りに画面に描画されません。3D 空間の「どこから」「どの方向」で見えるかなどの設定する事で画面に映る事になります。

その為、いくつかの座標系を持っており、それぞれの工程に沿って変換する事で必要な結果を得る事ができます。

主な座標系。

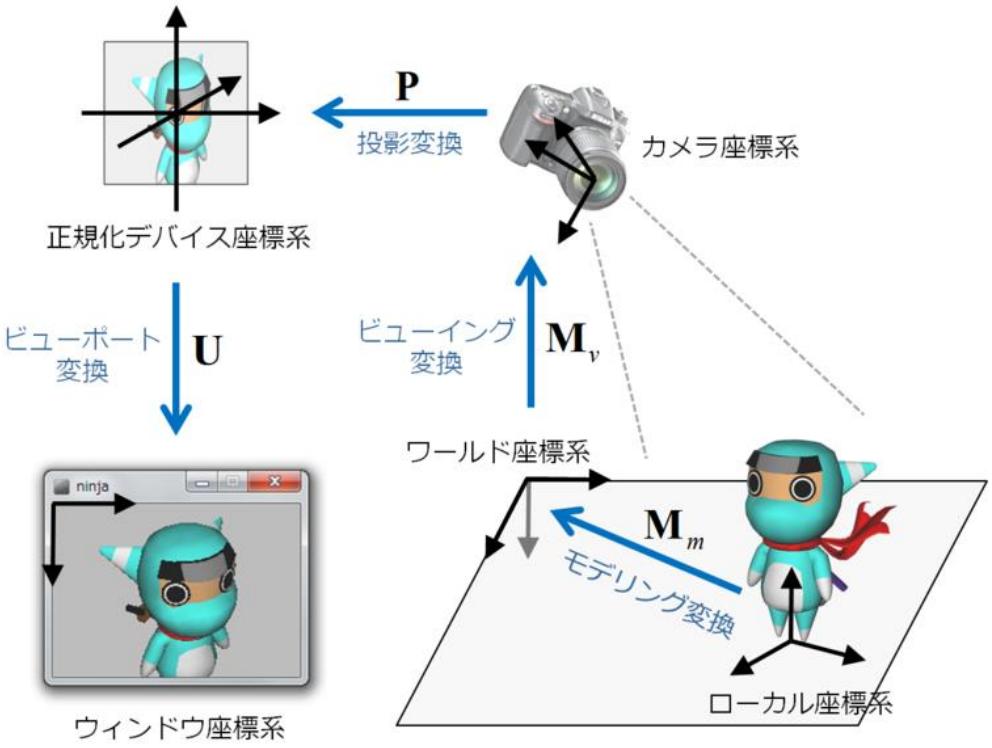
・モデルデータ単体の頂点の座標系・・・・・・・・・・・ ローカル座標系

・モデルやカメラなどの空間の中での座標系・・・・ ワールド座標系（モデリング変換）

・カメラの位置が原点座標になる座標系・・・・・・・ ビュー座標系（ビューイング変換）

・視野、パースや投影面の範囲を設定する・・・・・ 射影変換行列（プロジェクション変換） ※クリップ空間 −1〜+1

・スクリーン上での座標系・・・・・・・・・・・・・・・・・・ スクリーン座標系（ビューポート変換 ※自動）



開発イメージ

開発時の状態

ゲーム実行画面

■3D ゲームプログラミングの手順

1）2D に比べ 3D での制御はやるべき事が多くなります。しかしながら、手順を踏む事で確実に実装できますので根気よく取り組んでいきましょう。下記に大まかな考え方とやるべき事を記載していますので、この順番で制御を行って行きたいと思います。