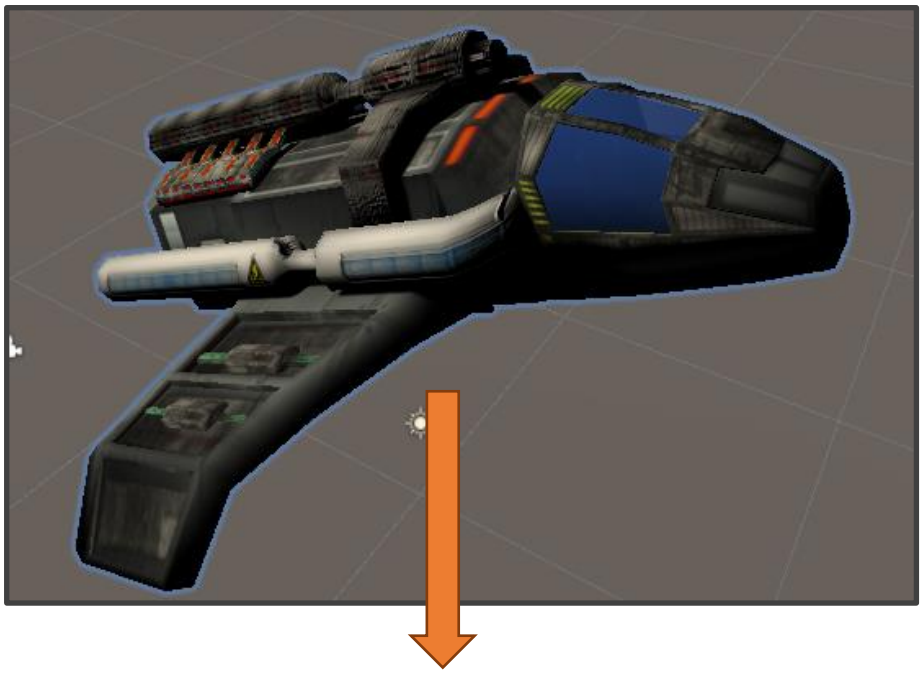
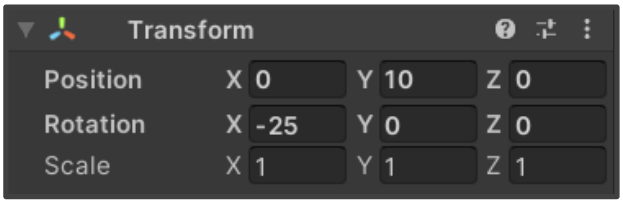
Transformクラスの説明

３Ｄの制御には、行列だったりクォータニオンだったり、
色々な方法があるため、クラス(３Ｄオブジェクト)ごとに処理が異なると、
制御の統一ができませんし、処理を修正したい場合、至る所に修正が入ります。

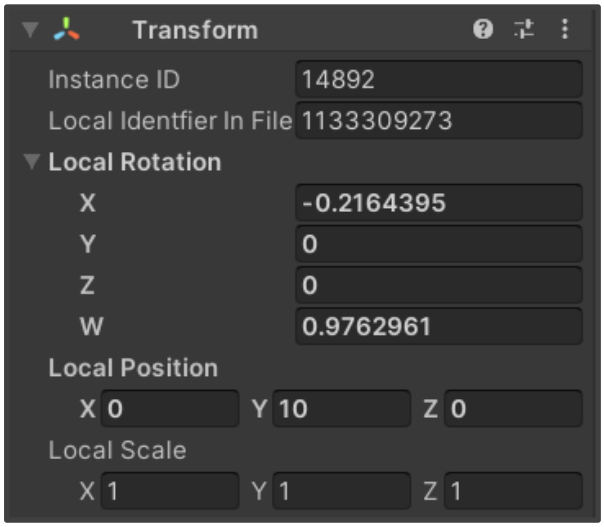
そこで、３Ｄの制御専用のクラスを作っておくと、
管理や運用がとても楽になりますので、
今回は、Unity風にTransformクラスを作ってみました。





Positionが位置(座標)
Rotationが角度(向き)
Scaleが大きさ

ちなみに角度がオイラー角(デグリー)で表示されていますが、
内部的には、クォータニオンが使用されています。
デバッグモードで表示してみると、下図のように四元数（しげんすう）で
管理されていることがわかります。



勉強のために、Transformクラスを使わずに、
自力でコードを書いたり、自分のアーキテクチャ設計に合わせて、
独自のクラスを作って貰っても大丈夫です。
ご参考になれば。

==========

```
class Transform
{

public:

    Transform(void);
    Transform(int model);

    // モデルのハンドルID
    int modelId;

    // 大きさ
    VECTOR scl;                    制御基準
    // 回転
    VECTOR rot;
    // 位置
    VECTOR pos;                    制御基準
```

```
MATRIX matScl;
MATRIX matRot;
MATRIX matPos;

// 回転
Quaternion quaRot;              制御基準

// ローカル回転
Quaternion quaRotLocal;

/// <summary>
/// モデル制御の基本情報更新
/// </summary>
/// <param name=""></param>
void Update(void);

void SetModel(int model);

// 前方方向を取得
VECTOR GetForward(void);

// 後方方向を取得
VECTOR GetBack(void);

// 右方向を取得
VECTOR GetRight(void);

// 左方向を取得
VECTOR GetLeft(void);

// 上方向を取得
VECTOR GetUp(void);

// 下方向を取得
VECTOR GetDown(void);

// 対象方向を取得
VECTOR GetDir(VECTOR vec);
```

};
==========

本プロジェクトで使用するTransformは、

    大きさ：VECTOR基準
    回転　：Quaternion基準
    位置　：VECTOR基準

で、計算し、それぞれ行列やDxLibの３Ｄ制御に使用します。
そのため、回転行列(matRot)の値を変えても、
３Ｄ制御に反映されません。

また、上記の制御基準となる値を変更したら、
Updateを必ず実行してください。
３Ｄ制御に反映されない場合があります。

    MATRIX matScl;
    MATRIX matRot;
    MATRIX matPos;

このあたりの行列系は、Updateの計算過程の中で、
メンバー変数に代入します。
何かの処理の計算に行列を使用したい場合は、
念のため、Updateを実行した後の方が無難です。
(やりすぎに注意！)

取り出しが面倒だと思いましたので、全てPublic属性にしました。
modelIdは、セットする前に何かしらのクッションがあった方が良いと
思いましたので、セッターを作りましたが、結局、やりたい処理が無かったので、
publicのままにしています。本当は隠蔽(private)にする予定でした。

わかりづらいのが、ローカル回転ですが、

```
// ローカル回転
Quaternion quaRotLocal;
```

モデルの向きがデフォルトで、Zの正方向を向いていない場合が多々あると思います。
UnityからFBX出力した際も、Zの負方向が正面になっていたりしましたので、
あくまでモデルの回転(向き)は、quaRotで、
オフセットのような位置づけで、モデルの向きを変えたい場合は、
quaRotLocalに値を入れておきます。

そのため、GetForwardなどの向きを取得する関数内では、
quaRotLocalを計算に入れていません。