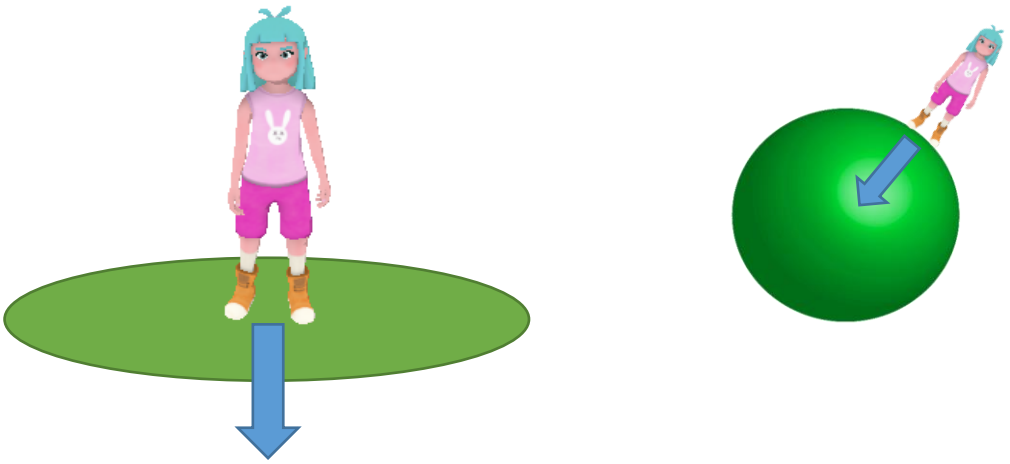
重力制御

いよいよ来ました。
このゲームのメインどころ、重力制御を実装していきます。
難しそうに感じるかもしれませんが、ここまできたら、簡単です。

なぜなら、これまで実装してきた処理が、
Yのプラスマイナスや、Zのプラスマイナスではなく、
(固定値を使っていない)
GravityManagerで管理させている回転／方向に沿って、
きちんとベクトル計算を行ってきたからです。

AsoGalaxyは、惑星によって、重力方向が変わってきます。



逆に言えば、惑星によって、重力方向は【確定】できるということになります。
確定できるところから、１つずつ計算していきましょう。

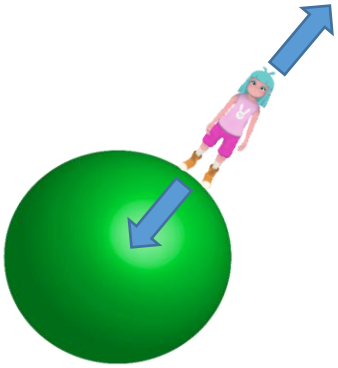
まずは、GravityManagerのCalcDirGravity関数にて、
惑星タイプがSPHERE(球体)である時の重力方向を作りましょう。
上図でいうと右側の惑星タイプになります。

これはすぐできますね・・・？

次に、GravityManagerのCalculate関数。
ここで、全ての回転・方向を計算していきます。

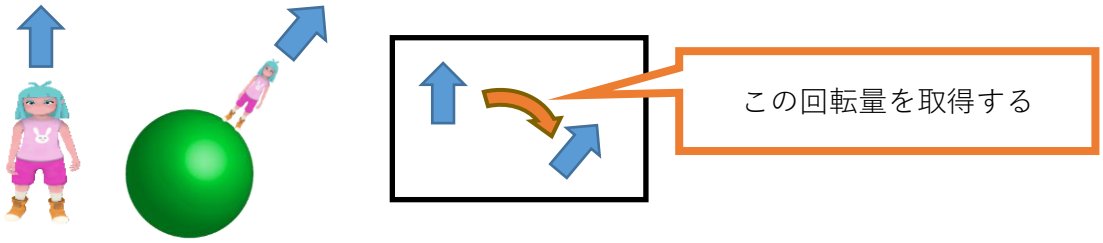
```
// まずは一番最初に確定できる【重力方向】
mDirGravity = CalcDirGravity();

// 重力方向が確定できたら、次に重力の反対【ジャンプ方向】が確定できる
mDirUpGravity = VScale(mDirGravity, -1.0f);
```



★ここがポイントです

この重力方向の変化によって、どれくらいの回転量が発生したかを確認します。
(回転向きがあべこべにならないように、上方向を基準に考えていきます)




```
// ２つのベクトル間の回転量を求める
VECTOR up = mTransform.GetUp();
Quaternion toRot = Quaternion::FromToRotation(up, mDirUpGravity);
```

mTransform.GetUp(); → これは、元々の上方向です

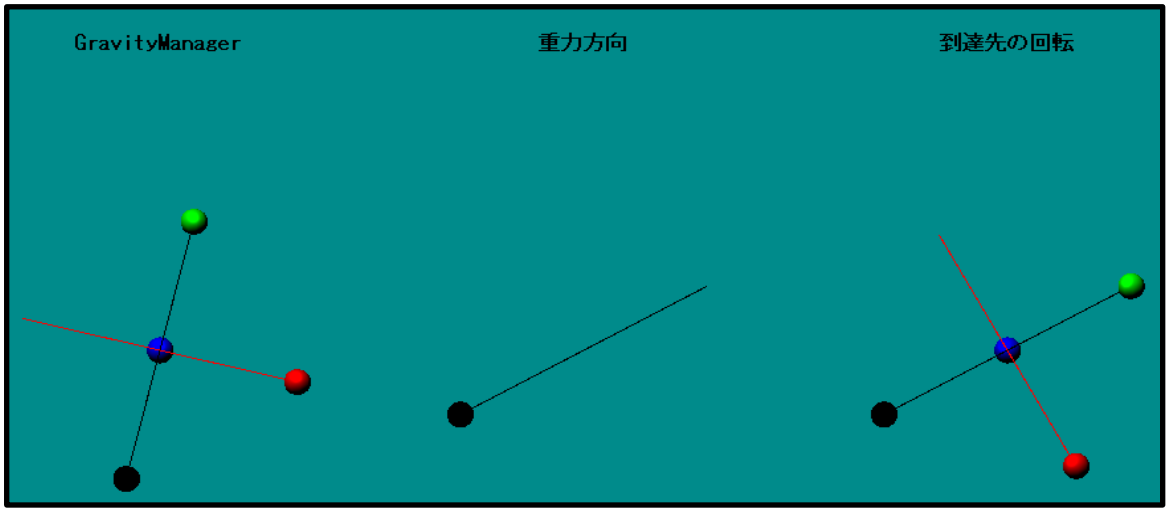
元の回転とこの回転量を合成すると、
今回の重力方向の変化分の回転が元の回転に加わります。

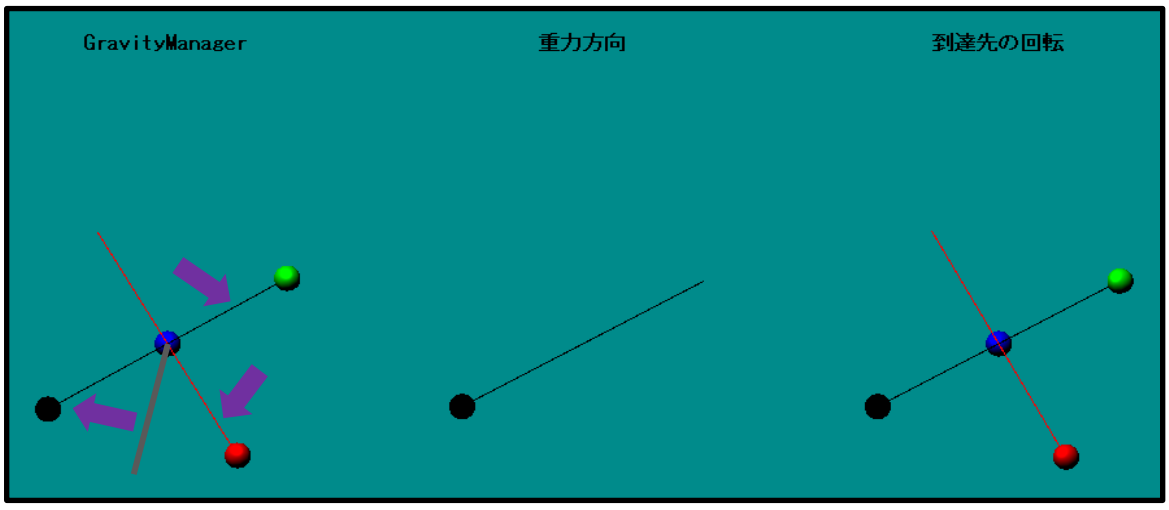
```
// 到達したい回転
toRot = toRot.Mult(mTransform.quaRot);
```

このtoRotをmTransform.quaRotに代入すると、
それで、回転の計算自体は完結するのですが、
一気に回転させると、カメラが急に回転して、酔ったりしますので、
徐々に回転させたい場合は、

```
// 球面補間
mTransform.quaRot = Quaternion::Slerp(mTransform.quaRot, toRot, mSlerpPow);
```

mSlerpPowには、デフォルトで0.08fが設定されていますので、
少しずつ到達したい回転に近づいていきます。
第3引数には、0～1の値を入れていきますので、一気に回転させたい場合は、
mSlerpPowの値を1に近づければ良いです。





デモ画面で見て貰ったとおり、
重力方向の回転分、きちんと追従するように回転してくれていますので、
この回転情報から、前方、後方、右方向、左方向を正しく取得する
ことができます。

これまで、カメラや、プレイヤーの挙動などは、
GravityManagerの回転を元に計算してきました。

カメラ

Camera::SyncTransform

```
// 重力の方向制御に従う
Quaternion gRot = mGravityManager->GetTransform()->quaRot;

// 正面から設定されたY軸分、回転させる
mQuaRotOutX = gRot.Mult(
    Quaternion::AngleAxis(mAngles.y, AsoUtility::AXIS_Y));

// 正面から設定されたX軸分、回転させる
mQuaRot = mQuaRotOutX.Mult(
    Quaternion::AngleAxis(mAngles.x, AsoUtility::AXIS_X));
```

重力制御の正面(前方)と、カメラの正面(前方)を合わせることで、
プレイヤーの進行方向を正しく取得することができる。
また、カメラ固有の首振り回転(X軸、Y軸)に関しては、
正面(前方)に対して、回転を合成していく。

プレイヤー

Player::UpdatePlay

```
// 重力方向に沿って回転させる
mTransform.quaRot = mGravityManager->GetTransform()->quaRot;
mTransform.quaRot = mTransform.quaRot.Mult(mPlayerRotY);
```

⇒ カメラと同じく、重力制御の正面と合わせる。
そして、プレイヤー固有の回転(Y軸のみ)を加える。

Player::ProcessMove

```
// X軸回転を除いた、重力方向に垂直なカメラ角度(XZ平面)を取得
Quaternion cameraRot = mSceneManager->GetCamera()->GetQuaRotOutX();

// 回転したい角度
double rotRad = 0;

VECTOR dir = AsoUtility::VECTOR_ZERO;

// カメラ方向に前進したい
if (CheckHitKey(KEY_INPUT_W))
{
    rotRad = AsoUtility::Deg2RadD(0.0);
    dir = cameraRot.GetForward();
}
```

移動方向に関しては、カメラの首振りも考慮しないといけないので、
カメラから取得してくる。
但し、X軸回転を考慮してしまうと、
地中や空中に移動してしまうので、X軸回転を抜いた、
Y軸回転のみの回転情報を取得して、移動させる。

たった数行のコードで実装できてしまいました。