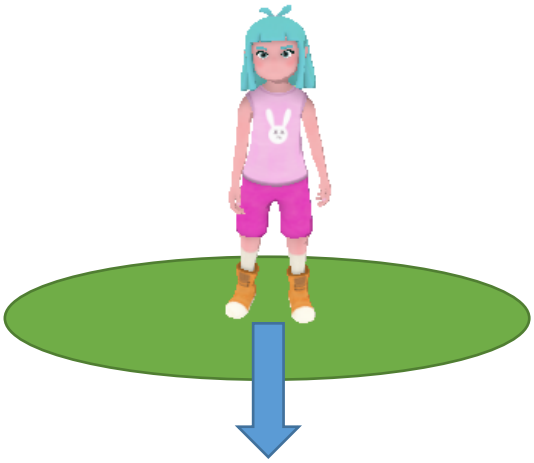
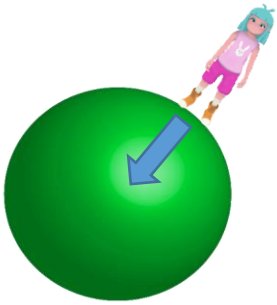
重力制御

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、ここまできたらあと一歩です。

仕様のおさらいです。
AsoGalaxyは、惑星によって、重力方向が変わってきます。



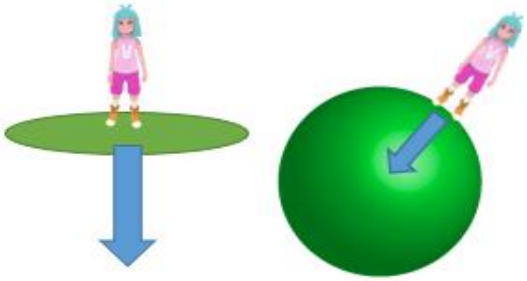


重力方向やジャンプ方向を処理に使用する際に、
Yのプラスマイナスや、Zのプラスマイナスなどの固定値は使えません。

そのため、これまでの実装では、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ました。

GravityManagerで管理させている、"回転／方向" をキチンと実装できれば、
全体が上手く処理されるはずです。

次には、どのようにして、回転させていくかの解説に入ります。



GravityManagerの“回転／方向”は、transform_.quaRot で管理しています。
こちらを、新しい惑星の重力方向と合うように、
回転させていくわけなのですが、まず、上図で確定ですぐに求められそうな
情報としては、『 重力方向 』です。

そして、重力方向を求めることができれば、
重力の反対方向を求めることができますので、実装していきます。

GravityManager.cpp

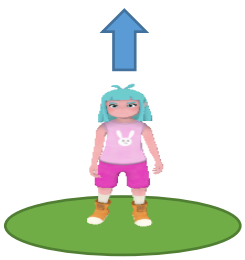
```
void GravityManager::Calculate(void)
{

    // 重力方向
    dirGravity_ = CalcDirGravity();

    // 重力の反対方向(ジャンプ方向)
    dirUpGravity_ = VScale(dirGravity_, -1.0f);

}
```

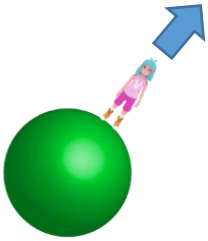
※ CalcDirGravityは、あとで演習として実装して貰います

重力の反対方向は、その回転方向と同じ意味になりますので、

```
    // 現在の上方向(つまり、重力の反対方向)
    VECTOR up = transform_.GetUp();
```

上記のコードを追記すると、
下図の２種類のベクトルを取得できている状態になります。

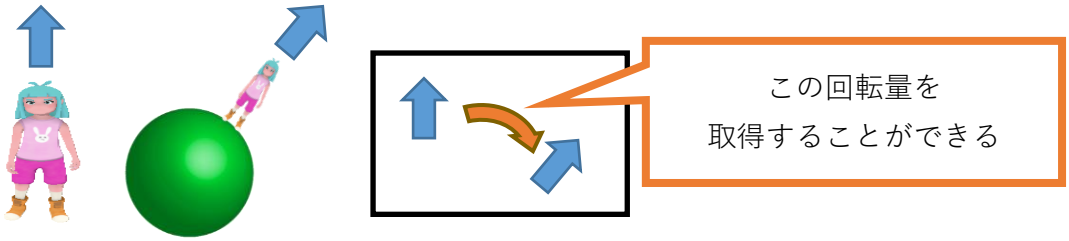




クォータニオンは、とても便利で、２つのベクトル間の回転量(回転差)を
求めることができます。

```
// ２つのベクトル間の回転量(差)を求める
Quaternion rot = Quaternion::FromToRotation(up, dirUpGravity_);
```

この関数を使用すれば、



現在の回転に対して、求めた回転差を加えることで、
計算した重力方向が下方向になるような回転を作ることができます。

※　下方向(重力方向)でも、同じことができるかもしれませんが、
　　回転向きがあべこべにならないように、上方向を基準する方が無難

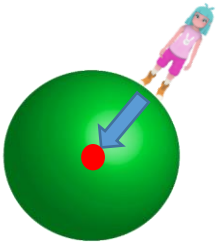
最後に以下のコードを追加します。

```
// 求めた回転量で、現在の重力制御を回転させる(差が埋まる)
transform_.quaRot = rot.Mult(transform_.quaRot);
```

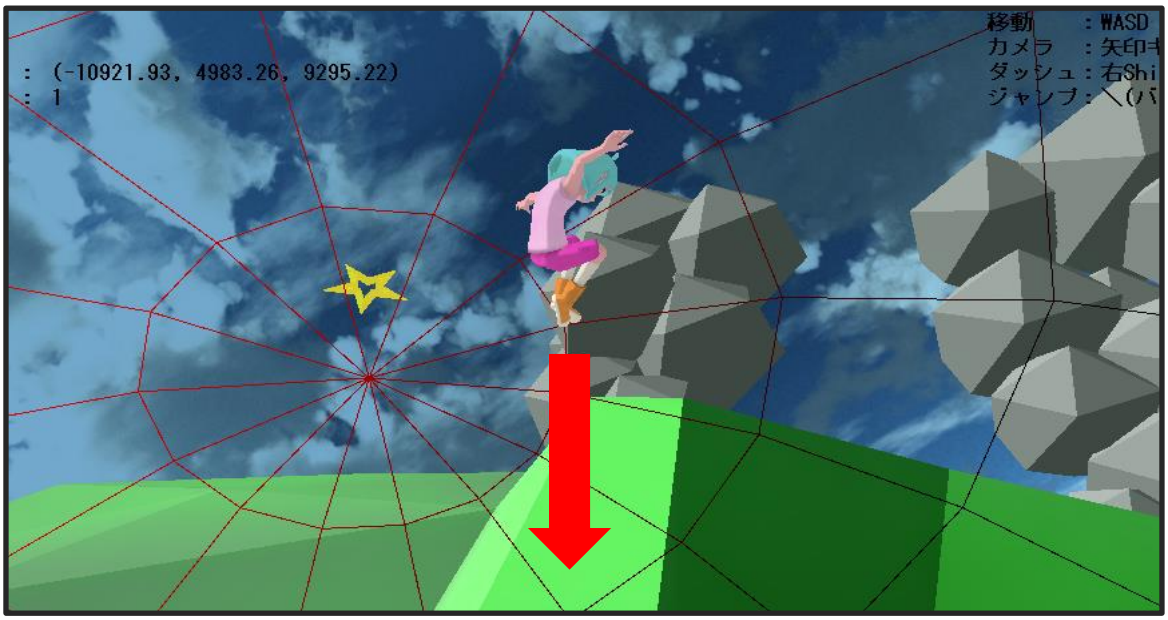
重力方向さえ、しっかり計算することができれば、
世界を回転させる制御ができるようになりました。

それでは演習です。

演習①　　CalcDirGravity関数で、TYPE::SPHEREにおける
重力方向を計算する処理を実装してください



惑星の座標と、
プレイヤーの座標がわかれば、
求めることができます



惑星に向かって重力が働き、惑星に着地できたらＯＫです。

但し、一気に世界が回転してしまいますので、
画面が急に切り替わるような挙動になってしまい、見栄えが良くありません。
そこで、徐々に回転していくように実装を変更します。

演習②　　球面補間を使用して、徐々に回転するように実装してください

Slerpの簡易的な使い方

```
from = Quaternion::Slerp(from, to, 0.1f);
```

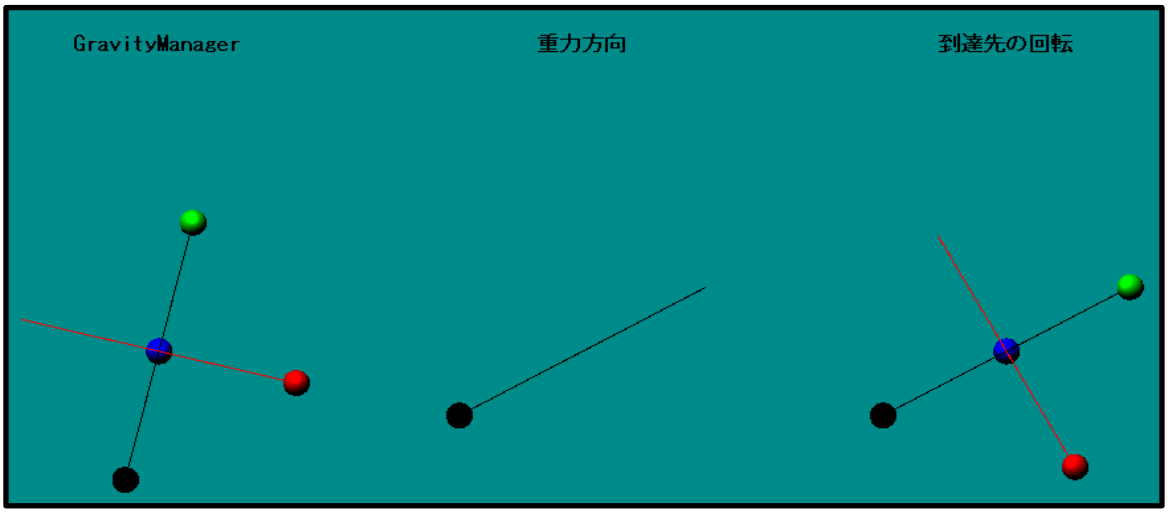
上記コードを記述すると、1割ずつ徐々に到達したい回転に
近づくようになります。

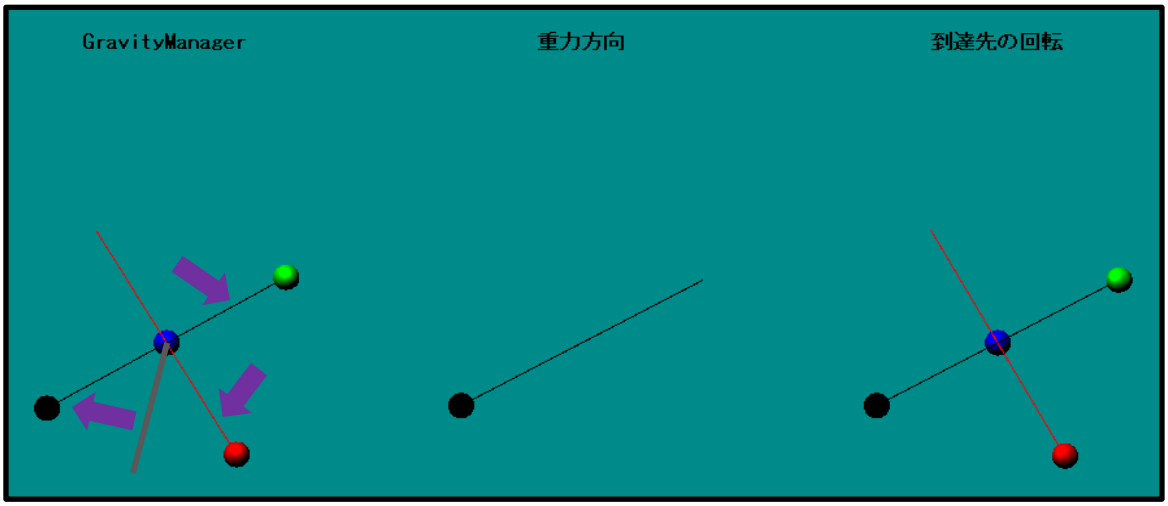
この指定した数字の大きさで、到達する速度が変わってくるように
なります。

ここでは、メンバ変数で宣言されている、slerpPow_ を
使用するようにしてください。

回転の補間

これまで何度か取り上げてきましたが、
どうしても、イメージがしづらい方は、自分で検証用のプロジェクトを作成して、
動きを目視していくと良いと思います。





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。