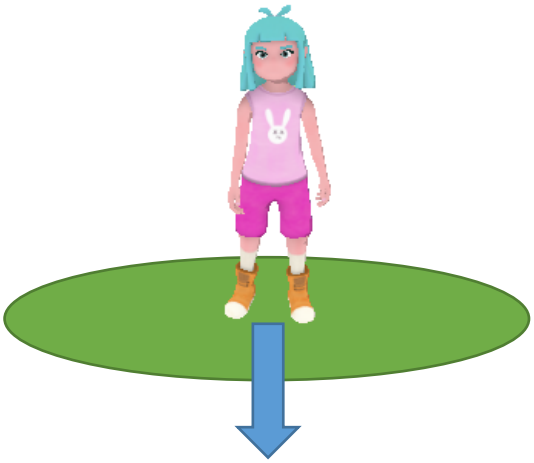
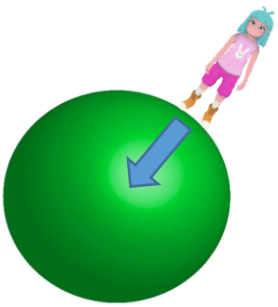
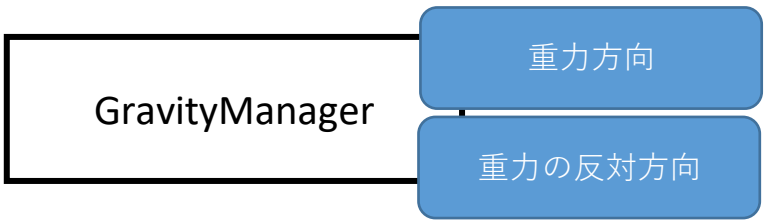
重力と衝突判定とジャンプ

キャラクターの移動と回転を行ってきましたが、
重力を実装していませんので、ずっと宙を浮いたままになっています。
そろそろ地面に足をつけて、フィールドを歩き回れるようにしていきます。





いつもなら、Yの値で高さを制御し、
重力方向は、Yがマイナス、ジャンプ方向はYのプラスでしたが、



AsoGalaxyは、そのような固定値は使わず、
重力制御で管理しているベクトルを使用していく必要があります。

```
// 重力方向を取得
VECTOR GetDirGravity(void);

// 重力方向の反対方向を取得
VECTOR GetDirUpGravity(void);

// 重力の強さを取得
float GetPower(void);
```

プレイヤーがいる惑星のタイプによって、重力方向や重力の強さが決まる。

その処理を、
GravityManagerで行う。

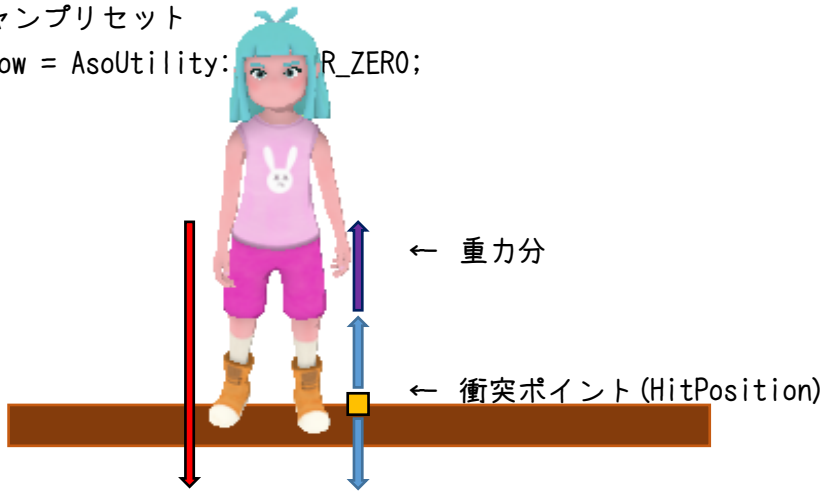
重力と合わせて、実装しなくてはならないのが、
地面との衝突判定ですが、

プレイヤーの座標(足元)を中心として、
ちょっと上から、ちょっと下あたりに、
線分を作り、衝突(コリジョン)チェックを
行うと良いでしょう。

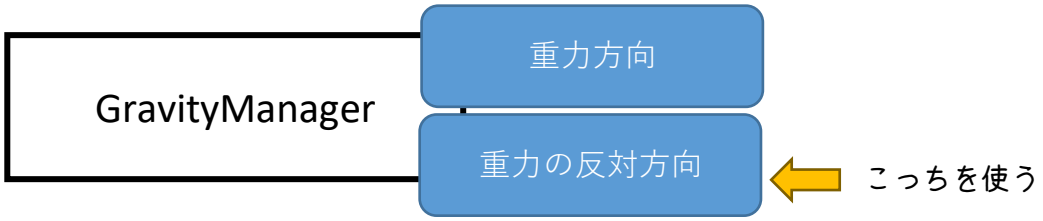
```
float checkPow = 10.0f;
mGravHitUp = VAdd(mMovedPos, VScale(dirUpGravity, gravityPow));
mGravHitUp = VAdd(mGravHitUp, VScale(dirUpGravity, checkPow * 2.0f));
mGravHitDown = VAdd(mMovedPos, VScale(dirGravity, checkPow));

// 地面との衝突
auto hit = MV1CollCheck_Line(c->mModelId, -1, mGravHitUp, mGravHitDown);
if (hit.HitFlag > 0)
{
    // 衝突地点から、少し上に移動
    mMovedPos = VAdd(hit.HitPosition, VScale(dirUpGravity, 2.0f));

    // ジャンプリセット
    mJumpPow = AsoUtility::VECTOR_ZERO;
}
```



次にジャンプ処理です。
Yにプラス、という処理は使いません。




```
mJumpPow = VScale(mGravityManager->GetDirUpGravity(), POW_JUMP);
```

ジャンプボタンの押下時間によって、
少しジャンプ量を上げるために、ジャンプキー受付時間を作ったり、
２段ジャンプをできないようにしたり(敢えて作っても良いです)、
アニメーションを簡易的にでも付けてあげたら、一応は完成です。

動作確認がしやすくなるように丸影をつけて上げても
良いかもしれません。

Dxライブラリのサイトに３Ｄのサンプルプログラムが、いくつかあります。

https://dxlib.xsrv.jp/program/dxprogram_3D.html

その中に、丸影のコードがあります。

https://dxlib.xsrv.jp/program/dxprogram_3DAction_CollObj.html

Player_ShadowRender関数

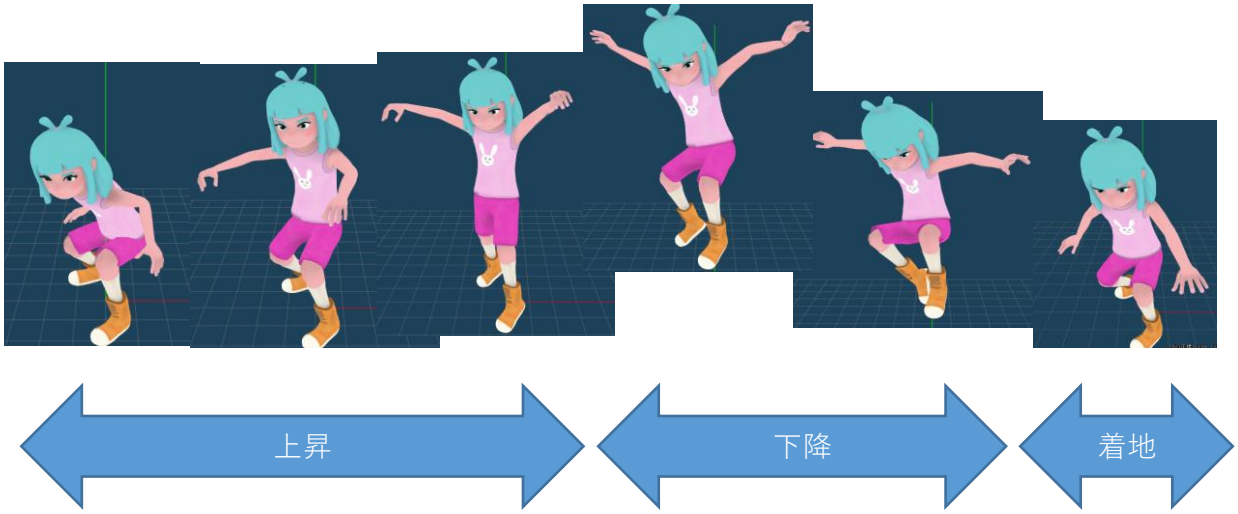
授業では解説しませんが、
画像は、"Data/Image/Shadow.png"に用意しておりますので、
そのまま使用して貰うと良いと思います。

色んなジャンプのやり方がありますので、各々作って貰っても
大丈夫ですが、ここでは２つ、習得して貰いたいことがありますので、
そこは押さえておいて貰いたいです。

覚えて貰いたいこと①

> ジャンプアニメーションは、
> 上昇、下降と最低２種類に分けて！

ジャンプの量が変わるということは、時間が変わるということです。
時間が変われば、アニメーションの時間も変わります。
速度を調整するという手もありますが、上昇・下降の時間比率が
一緒だとは限りません。
上昇中は、上昇アニメーション。
下降中は、下降アニメーションを再生しましょう。
（できれば、着地も分けて再生したい）



Mixamoサイトでは、上昇・下降を分けてくれている
アニメーションもありますが、それが無い場合もあります。
また、私達はプログラマーですので、３Ｄソフトを使って、
モーションデザインに手を出すことができません。

そういった場合は、繋がったアニメーションのうち、
自分の都合の良い部分を再生できるようなプログラムを作りましょう。

```
// 13～25フレームを再生して、上昇アニメーション風に
mAnimationController->Play((int)ANIM_TYPE::JUMP, true, 13.0f, 25.0f);

// 上記の再生が終わったら、23～25フレームをスピード5で再生し、
// 下降モーション風にする
mAnimationController->SetEndLoop(23.0f, 25.0f, 5.0f);
```

```
// 29～45フレームを再生して、着地アニメーション風に
mAnimationController->Play(
            (int)ANIM_TYPE::JUMP, false, 29.0f, 45.0f, false, true);
```

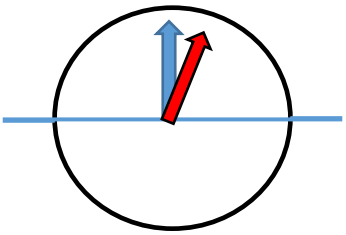
ループしない、強制アニメーションON

覚えて貰いたいこと②

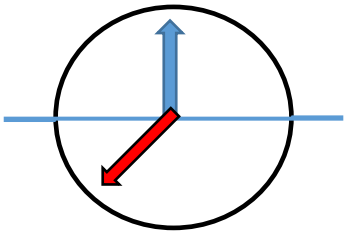
> AsoGalaxyでの上昇方向、下降方向の判断は、
> Yのプラスマイナスではなくて、内積で判断する！

内積の性質として、２つのベクトルが同じ方向を向いていたらプラス、
片方が反対の方向を向いていたら、マイナスになります。

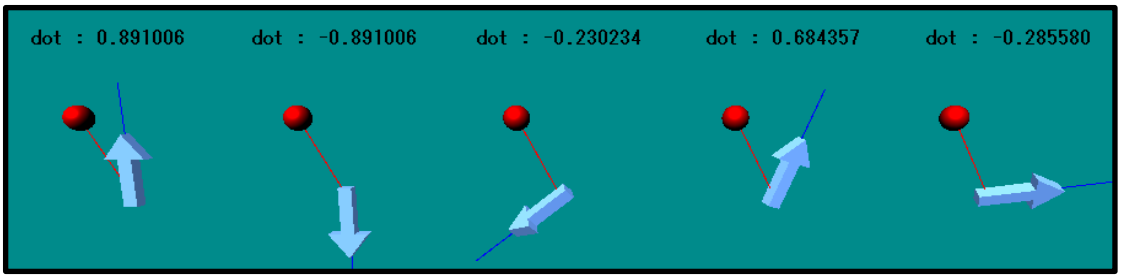
VDot(青ベクトル、赤ベクトル)

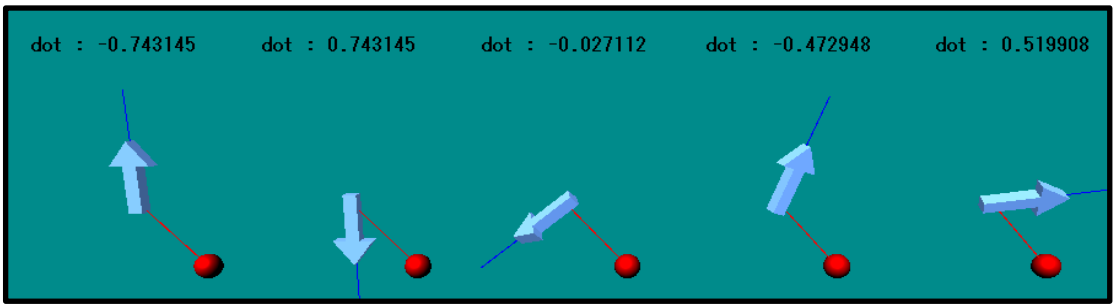


プラスになる。



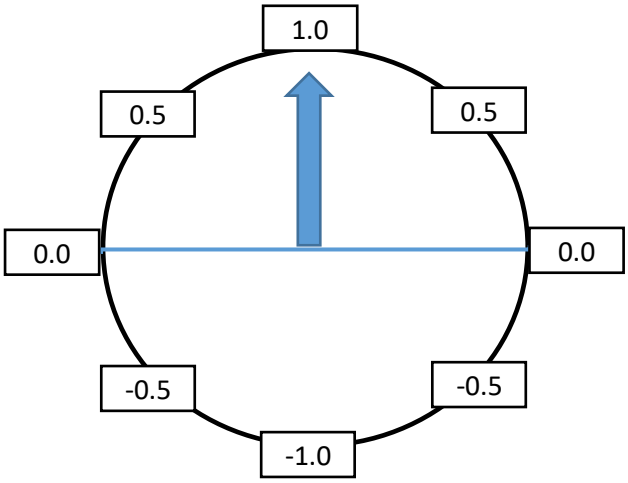
マイナスになる。





更に、渡すベクトルが単位ベクトルの場合、
-1.0～+1.0の値になる。

単位ベクトル同士の内積



ということは、

VDot(重力ベクトル、ジャンプの移動ベクトル)
VDot(dirGravity、mJumpPow)

この内積結果が、0.0以上であれば、下降中、という判断になります。
逆にマイナスであれば、上昇中です。