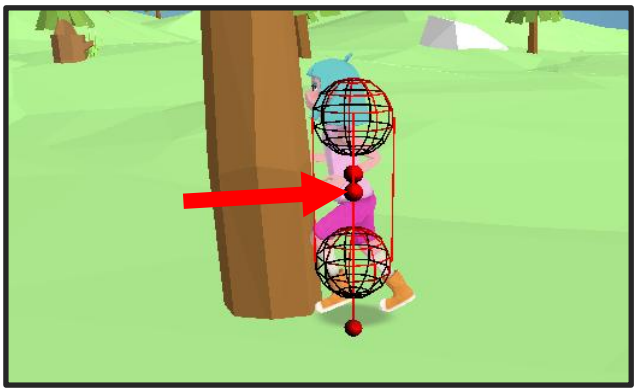
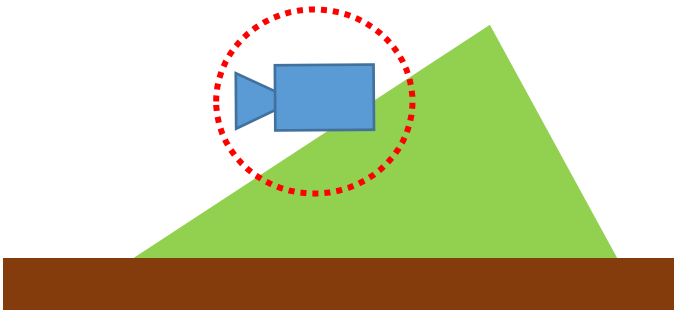
衝突処理の関数化

キャラクターやカメラに、各衝突形状ごとの押し戻し処理を実装していますが、
このやり方だと、他のActorで同様の衝突判定を実装しようとした場合、
押し戻し処理(同じ処理内容)を他ののActor内でも記述しないといけません。



CharactorBaseの
CollisionCapsule関数

モデル内の衝突ポリゴンの
法線方向にカプセルを
押し戻す

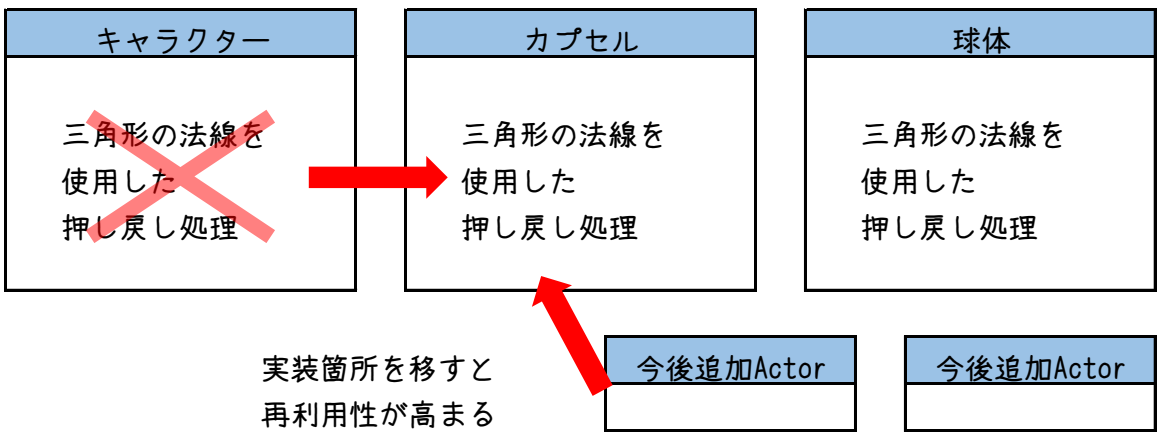


CameraのCollision関数

モデル内の衝突ポリゴンの
法線方向に球体を
押し戻す

同じ処理が複数箇所にあるということは、再利用性が確保されていない証で、
再利用性が低いと、機能追加や不具合修正を行う際に、時間がかかってしまいます。

三角形の法線を使用した押し戻し処理は、各形状ごとに処理内容は
異なりますが、やりたいことは一緒になりますので、
その特性を理解した上で、設計・実装していきます。



まずは、カプセルから。

ColliderCapsule.h

```
public:

    ～ 省略 ～

    // 指定された回数と距離で三角形の法線方向に押し戻した座標を取得
    VECTOR GetPosPushBackAlongNormal(
        const MV1_COLL_RESULT_POLY& hitColPoly,
        int maxTryCnt,
        float pushDistance) const;
```

ColliderCapsule.cpp

```
VECTOR ColliderCapsule::GetPosPushBackAlongNormal(
    const MV1_COLL_RESULT_POLY& hitColPoly,
    int maxTryCnt, float pushDistance) const
{

    // コピー生成
    Transform tmpTransform = *follow_;
    ColliderCapsule tmpCapsule = *this;
    tmpCapsule.SetFollow(&tmpTransform);

    // 衝突補正処理
    int tryCnt = 0;
    while (tryCnt < maxTryCnt)
    {

        // カプセルと三角形の当たり判定
        if (!HitCheck_Capsule_Triangle(
            tmpCapsule.GetPosTop(), tmpCapsule.GetPosDown(),
            tmpCapsule.GetRadius(),
            hitColPoly.Position[0], hitColPoly.Position[1],
            hitColPoly.Position[2]))
        {
            break;
        }
```

```
        // 衝突していたら法線方向に押し戻し
        tmpTransform.pos =
            VAdd(tmpTransform.pos, VScale(hitColPoly.Normal, pushDistance));

        tryCnt++;

    }

    return tmpTransform.pos;

}
```

CollisionCapsule.cpp

```
void CharactorBase::CollisionCapsule(void)
{


        // 衝突した複数のポリゴンと衝突回避するまで、
        // プレイヤーの位置を移動させる
        for (int i = 0; i < hits.HitNum; i++)
        {

            auto hit = hits.Dim[i];

            // 除外フレームは無視する
            if (colliderModel->IsExcludeFrame(hit.FrameIndex))
            {
                continue;
            }

            // 指定された回数と距離で三角形の法線方向に押し戻す
            transform_.pos =
                colliderCapsule->GetPosPushBackAlongNormal(
                    hit, CNT_TRY_COLLISION, COLLISION_BACK_DIS);

        }

        // 検出した地面ポリゴン情報の後始末
        MV1CollResultPolyDimTerminate(hits);
```

```
    }

}
```

【要件①】

ColliderSphereクラスにも同様の三角形の法線を使用した
押し戻し処理を実装し、Cameraクラスに組み込むこと。

ColliderSphere.h

```
public:

    ～ 省略 ～

    // 指定された回数と距離で三角形の法線方向に押し戻した座標を取得
    VECTOR GetPosPushBackAlongNormal(
        const MV1_COLL_RESULT_POLY& hitColPoly,
        int maxTryCnt,
        float pushDistance) const;
```

【目標①】

これまでと同様にカメラと地面の押し戻し処理が実行され、
地面にカメラがめり込まないこと。

【要件②】

Cameraクラスに実装されている、
線分とモデルの最近接(startに近い)衝突ポリゴンを取得する機能を
ColliderModelクラスに写し、再利用性を高めること。

○ ColliderModelクラスに実装する関数の定義は以下のようにすること

ColliderSphere.h

```
public:

    ～ 省略 ～

    // 線分とモデルの最近接(startに近い)衝突ポリゴンを取得
    MV1_COLL_RESULT_POLY GetNearestHitPolyLine(
        const VECTOR& start,
        const VECTOR& end,
         bool isExclude = false, bool isTarget = false) const;
```

○ Cameraクラス側で、GetNearestHitPolyLine関数を呼び出し、
最近接(startに近い)衝突ポリゴン情報を取得して、
処理を置き換えること

【要件②】

これまでと同様にカメラと地面の押し戻し処理が実行され、
地面にカメラがめり込まないこと。