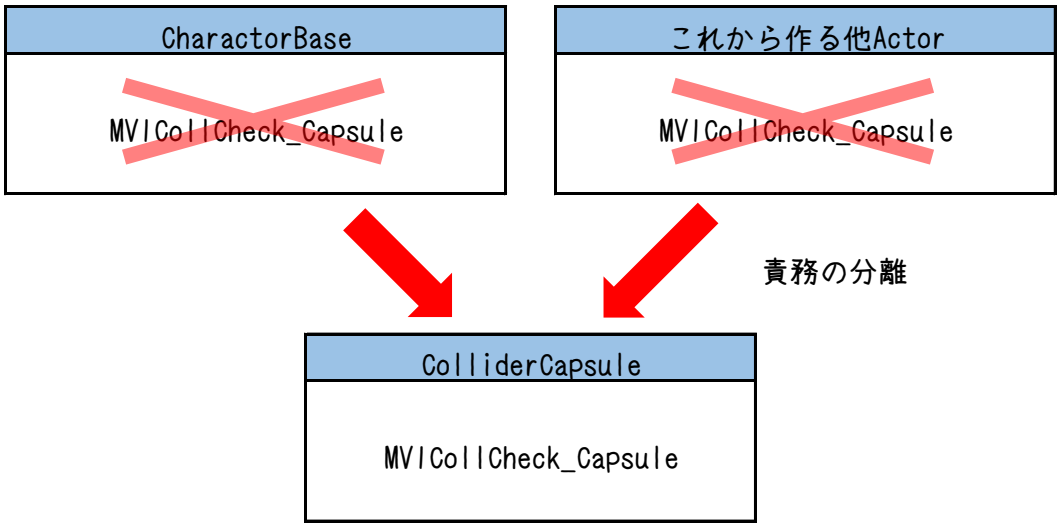
衝突処理の関数化②

衝突に関する処理が関数化され、コード整理された状態になりましたが、
まだ続きがあります。

CharactorBaseのCollisionCapsule関数において、
モデルとカプセルの衝突判定、MVICollCheck_Capsuleが記述されています。
このままだと、他のActor派生が増えた時に、
同じようにMVICollCheck_Capsule関数を記述する必要があります。

この衝突判定こそが、本来は、責務の分離(単一責務原則)として、
Colliderクラスへ移したい処理になります。



ColliderCapsule.h

```
class ColliderCapsule : public ColliderBase
{

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す
    void PushBackAlongNormal(
        const ColliderModel* colliderModel, Transform& transform,
        int maxTryCnt, float pushDistance,
        bool isExclude = false, bool isTarget = false) const;
```

ColliderCapsule.cpp

```
void ColliderCapsule::PushBackAlongNormal(
    const ColliderModel* colliderModel, Transform& transform,
    int maxTryCnt, float pushDistance, bool isExclude, bool isTarget) const
{

    // モデルとカプセルの衝突判定
    auto hits = MV1CollCheck_Capsule(
        colliderModel->GetFollow()->modelId, -1,
        GetPosTop(), GetPosDown(), GetRadius());

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

        auto hitPoly = hits.Dim[i];

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

```
        // 対象フレーム以外は無視する
        if (isTarget && !colliderModel->IsTargetFrame(hitPoly.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 =
            GetPosPushBackAlongNormal(hitPoly, maxTryCnt, pushDistance);

    }

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

}
```

【要件①】

CharactorBaseのCollisionCapsule関数から、
MV1CollCheck_Capsule関数を無くし、ColliderCapsuleクラスに処理を委任すること。

【目標①】

これまでと同様にキャラクターとステージの木と岩が衝突を行い、
めり込まないこと。

【要件②】

CharactorBaseのCollisionGravity関数から、
MV1CollCheck_LineDim関数を無くし、ColliderLineクラスに処理を委任すること。

・ ColliderLineクラスに委任先の関数を新たに作成すること

【目標②】

これまでと同様にキャラクターと地面が衝突を行い、地面に接地すること。