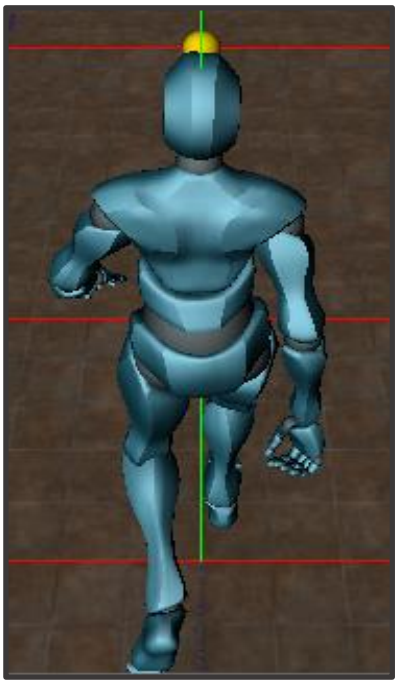
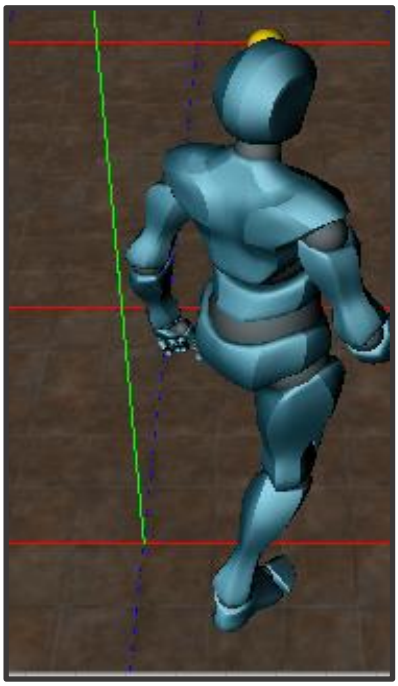
キャラクターの回転動作

３ＤＷｏｒｌｄの時のキャラクターの回転は移動方向に向かって、
回転動作が少ない回転の向きで(時計回りか、反時計回り)で、
『 キー入力している間 』回転するようにプログラムを組んでいました。

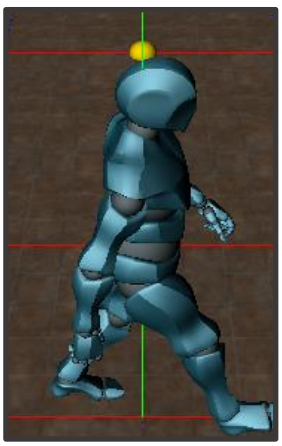


ちょっと
Ｄキーを押した時は
ちょっとだけ
回転する



アクションゲームになると、
進みたい方向や、攻撃方向などを細かく操作したい場合、
この回転方法だと、軸がなかなか定まらず、ユーザにストレスを
与えてしまう場合があります。

そこで、例えばＤキーを一瞬でも押したならば、



完全に右側に振り向くようにしていきたいと思います。

もちろん、瞬間的に振り向くのではなくて、
ゆっくりと、振り向かせます。
(キーを押していない時も、回転動作が続くように)

Player.h

```
public:

    ～ 省略 ～

    // 回転完了までの時間
    static constexpr float TIME_ROT = 1.0f;

private:

    ～ 省略 ～

    // 回転
    Quaternion playerRotY_;
    Quaternion goalQuaRot_;
    float stepRotTime_;

    ～ 省略 ～

    // 回転
    void SetGoalRotate(double rotRad);
    void Rotate(void);
```

Player.cpp

```
void Player::ProcessMove(void)
{

    ～ 省略 ～        → rotRad に回転角度を代入する
                        Dキーだとしたら、右方向を向きたいので、90度(deg)

    if (!AsoUtility::EqualsVZero(dir))
    {

        // 移動処理
        speed_ = SPEED_MOVE;
        if (ins.IsNew(KEY_INPUT_RSHIFT))
        {
```

```
            speed_ = SPEED_RUN;
        }
        moveDir_ = dir;
        movePow_ = VScale(dir, speed_);

        // 回転処理
        SetGoalRotate(rotRad);

        ～ 省略 ～

}

void Player::SetGoalRotate(double rotRad)
{

    VECTOR cameraRot = mainCamera->GetAngles();
    Quaternion axis =
        Quaternion::AngleAxis(
            (double)cameraRot.y + rotRad, AsoUtility::AXIS_Y);

    // 現在設定されている回転との角度差を取る
    double angleDiff = Quaternion::Angle(axis, goalQuaRot_);

    // しきい値
    if (angleDiff > 0.1)
    {
        stepRotTime_ = TIME_ROT;
    }

    goalQuaRot_ = axis;

}
```

↑プレイヤーに向かせたい、ゴールとなる回転を設定する

```
void Player::UpdatePlay(void)
{

    // 移動処理
    ProcessMove();

    // 移動方向に応じた回転
    Rotate();

    // 現在座標を起点に移動後座標を決める
    movedPos_ = VAdd(transform_.pos, movePow_);

    // 移動
    transform_.pos = movedPos_;

    // 重力方向に沿って回転させる
    transform_.quaRot = grvMng_.GetTransform().quaRot;
    transform_.quaRot = transform_.quaRot.Mult(playerRotY_);

}
```

↑重力制御による正面の回転からY軸回転させる。このゲーム特有の処理

```
void Player::Rotate(void)
{

    stepRotTime_ -= scnMng_.GetDeltaTime();

    // 回転の球面補間
    playerRotY_ = Quaternion::Slerp(
        playerRotY_, goalQuaRot_, (TIME_ROT - stepRotTime_) / TIME_ROT);

}
```

↑回転の球面補間を行う。
TIME_ROT定数で指定された時間をかけて、ゆっくりゴールとなる
回転に向かって近づくような回転を行う