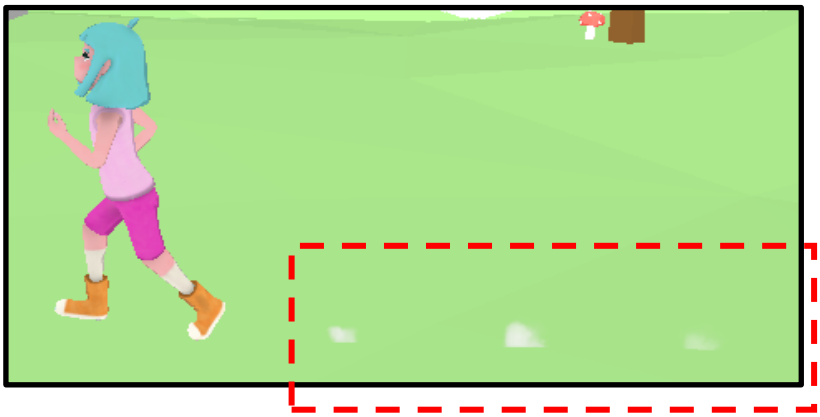
エフェクトの実装

これから、２種類のエフェクトを実装して貰います。
使用するエフェクトの種類は、いつも通りEffekseerです。

■ プレイヤーの移動エフェクト(足煙)



プレイヤーが移動した際に、一定間隔で足煙を表示されることで、
移動している感じが、伝わります。

Player.h

```
public:

    // 煙エフェクト発生間隔
    static constexpr float TERM_FOOT_SMOKE = 0.3f;

private:

    // 足煙エフェクト
    int effectSmokeResId_;
    float stepFootSmoke_;

    // フレームごとの移動値
    VECTOR moveDiff_;

    ～ 省略 ～

    // 足煙エフェクト
    void EffectFootSmoke(void);
```

Player.cpp

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

    ～ 省略 ～

    // 歩きエフェクト
    EffectFootSmoke();

}
```

エフェクトはリソース管理内にあります。

```
    ResourceManager::SRC::FOOT_SMOKE
```

今回の機能要件としては、以下の３つです。

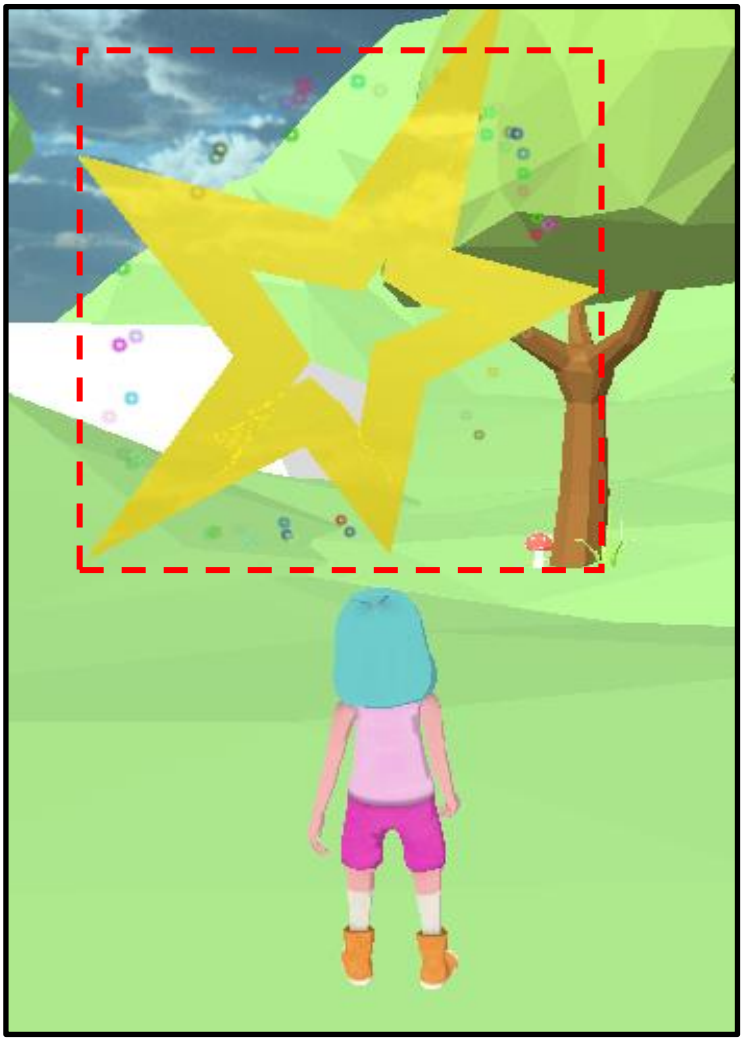
- 移動したら、エフェクトを発生させる
- エフェクトは、一定間隔で何度も発生させる
- 但し、ジャンプ中は発生させない

「 移動したら～ 」という記載がありますので、
移動キーが押されたらとか、イベントをキャッチしてエフェクト再生を
行うイメージをされる方が多いかと思いますが、
移動前座標と移動後座標を比較すると、移動値が取れますので、
そちらを使用して、エフェクト再生の判断を行うと良いでしょう。

Player.cpp

```
void Player::Collision(void)
{

    ～ 省略 ～

    // 移動
    moveDiff_ = VSub(movedPos_, transform_.pos);
    transform_.pos = movedPos_;

}
```

■ スターエフェクト



星の周りをキラキラさせたエフェクトです。
まずは、星のモデルをZ回転させ続けてみましょう。

WarpStar.h

```
private:

    ～ 省略 ～

    // エフェクト
    int effectRotParticleResId_;
    float stepEffect_;

    ～ 省略 ～
```

```
    // 回転
    void RotateZ(float speed);

    // エフェクト再生
    void PlayEffectRotParticle(void);
```

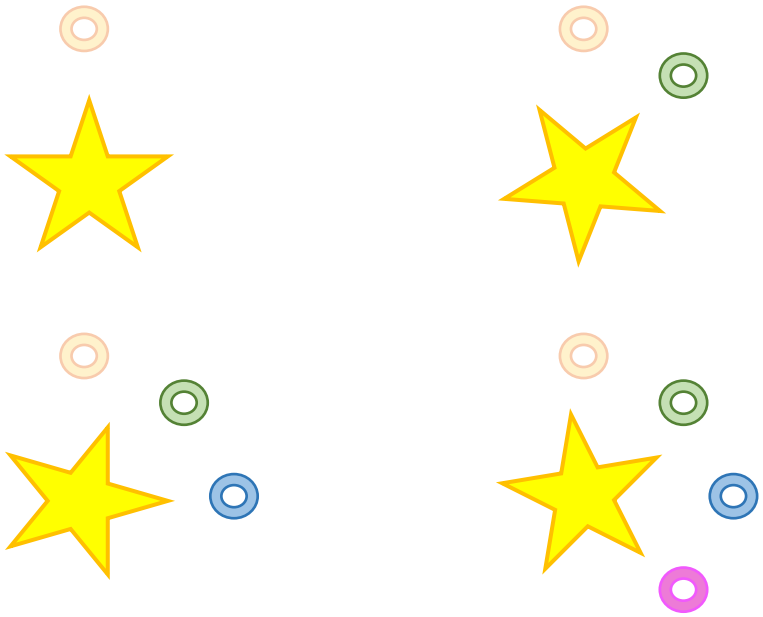
WarpStar.cpp

```
void WarpStar::Init(void)
{

    ～ 省略 ～

    // 回転エフェクト
    effectRotParticleResId_ = resMng_.Load(
        ResourceManager::SRC::WARP_STAR_ROT_EFF).handleId_;

    stepEffect_ = TERM_EFFECT;

    ChangeState(STATE::IDLE);

}

void WarpStar::UpdateIdle(void)
{

    // 回転
    RotateZ(SPEED_ROT_IDLE);

    // エフェクト
    PlayEffectRotParticle();

}
```

↑ RotateZ関数は、引数で指定されたスピードで、
星のモデルを回転させる

星の回転ができたら、PlayEffectRotParticle関数で
エフェクトを再生する

理屈としては、
星自体が回転していますので、星から少し離れた相対座標に
エフェクトを発生させると、



星の周りを彩るような演出になります。