コンポーネント構成

```
PostProcessEffectManager (ActorComponent)
├── UPostProcessComponent (内部で自動生成)
└── 複数のエフェクト (タグ管理)
    ├── Recording
    ├── Rewinding
    ├── SlowMotion
    ├── Damage
    ├── LowHealth
    ├── Boost
    ├── WallRun
    └── Custom1~3
```

主要な役割

PostProcessEffectManager

- 役割: 画面エフェクトの一元管理システム
- 主な機能:
    - タグベースでエフェクトを識別・制御
    - フェードイン/アウトのスムーズな切り替え
    - 優先度システム（複数エフェクト同時表示時の制御）
    - マテリアルパラメータの動的変更

UPostProcessComponent

- 役割: 実際のポストプロセス描画
- 関係: PostProcessEffectManagerが内部で生成・管理

FActivePostProcessEffect（構造体）

- 役割: アクティブなエフェクトの実行時情報
- 保持データ:
    - タグ（識別子）
    - MaterialInstanceDynamic（実際のエフェクト）
    - 現在のウェイト/目標ウェイト
    - 優先度

データの流れ

1. エフェクト設定 (エディタ)
    - `EffectConfigs` マップにマテリアルと設定を登録
2. エフェクト有効化

```
ActivateEffect(Tag)
 → MaterialInstanceDynamic作成
 → ActiveEffectsに追加
 → フェードインコルーチン開始
 → 毎フレームRefreshPostProcessSettings()
```

3. エフェクト無効化

```
DeactivateEffect(Tag)
 → フェードアウトコルーチン開始
 → ウェイト0になったらActiveEffectsから削除
```

技術的特徴

UE5Coroによるコルーチン実装

- 利点: Tick不要、非同期的なフェード処理
- 使用箇所:
    - `FadeInEffect()` - ウェイト 0→1 への補間
    - `FadeOutEffect()` - ウェイト 1→0 への補間
    - `ChangeEffectWeight()` - 任意のウェイト変更

優先度システム

- 複数エフェクト同時表示時、`Priority`値が高い順に適用
- `SortActiveEffects()` で自動ソート

使用例

cpp
```
// ダメージを受けた時
EffectManager->ActivateEffect(EPostProcessEffectTag::Damage);

// 0.5秒後に自動でフェードアウトさせる場合
// (別途タイマーやコルーチンで)
EffectManager->DeactivateEffect(EPostProcessEffectTag::Damage);

// ブースト中はFOV変更と組み合わせ
EffectManager->ActivateEffect(EPostProcessEffectTag::Boost);
CameraControl->SetFOV(110.0f);
```

カメラシステムとの連携

- 独立動作: 各システムは独立しているが、視覚演出で協調

- 組み合わせ例:
    - ブースト時: カメラFOV拡大 + Boostエフェクト
    - 着地時: カメラシェイク + 一瞬のダメージエフェクト
    - 壁走り時: カメラロール + WallRunエフェクト