

コンポーネント構成

UBlackoutComponent (ActorComponent)

 └─ UCharacterMovementComponent への参照

 └─ UPlayerCameraControlComponent への参照

主要な役割

UBlackoutComponent

- 役割: キャラクターに一時的な高速移動能力を付与
- 主な機能:
 - ブースト発動条件の判定
 - 速度・摩擦パラメータの動的変更
 - カメラFOV連動による演出
 - クールダウン管理
 - 空中時のクールダウン遅延

ブーストの処理フロー(BoostSequence)

フェーズ1: ブースト開始

1. 前方向に強い力を加える (BOOST_FORCE = 2500.0f)
2. 摩擦を下げる (0.2f) → 滑らかに加速
3. ブレーキ性能を下げる (64.0f) → 急停止しない
4. カメラFOVを拡大 (通常90° → 110°) → スピード感演出
5. ブースト効果音再生

フェーズ2: ブースト持続(0.5秒)

co_await Seconds(BoostDuration)

→ この間、プレイヤーは高速で移動し続ける

フェーズ3: ブースト終了後の処理

1. 摩擦・ブレーキを元に戻す
2. 入力があるか確認
 - └─ 入力あり → 高速維持モード (3000.0f)
 - └─ 入力がなくなるまで待機 (co_await Until)
 - └─ 入力なし → すぐに通常速度へ
3. MaxWalkSpeedを元の値に戻す
4. カメラFOVをリセット

フェーズ4: クールダウン

```

while (CooldownRemaining > 0.0f)
|— 地上: 通常速度でカウントダウン (1.0秒)
|— 空中: 半分の速度 (2.0秒) ← 空中ブーストを防止

```

主要な定数

```

cpp
namespace BoostConstants
{
    MIN_BOOST_SPEED = 300.0f      // 最低速度条件
    BOOST_FORCE = 2500.0f         // 加速力
    AIR_COOLDOWN_MULTIPLIER = 0.5f // 空中時のクールダウン速度
    BOOST_FRICTION = 0.2f         // ブースト時の摩擦
    BOOST_DECELERATION = 64.0f    // ブースト時の減速
    DEFAULT_FRICTION = 8.0f       // 通常の摩擦
    DEFAULT_DECELERATION = 2048.0f // 通常の減速
    POST_BOOST_SPEED = 3000.0f    // ブースト後の高速維持速度
}

```

発動条件 (CanBoost)

1. **Owner**とコンポーネントが有効
2. 現在ブースト中でない
3. 現在の移動速度が**300.0f**以上 ← 重要: 助走が必要

技術的特徴

1. UE5Coroによる状態管理

```

cpp
TCoroutine<> BoostSequence()
|— co_await Seconds(0.5f)      // ブースト持続
|— co_await Until([...])       // 入力待機
|— while + co_await NextTick() // クールダウン

```

- 利点: Timer不要、状態遷移が明確、キャンセル処理が簡単

2. CharacterMovementの物理パラメータ操作

```

cpp
// 滑りやすくして加速を維持
CachedMovement->GroundFriction = 0.2f;
CachedMovement->BrakingDecelerationWalking = 64.0f;

// 元に戻す

```

```
CachedMovement->GroundFriction = 8.0f;  
CachedMovement->BrakingDecelerationWalking = 2048.0f;
```

3. 入力ベースの速度維持

cpp

```
const bool bHasInput = !CachedMovement->GetLastInputVector().IsNearlyZero(0.01f);
```

- プレイヤーが入力し続ける限り高速状態を維持
- 入力を離すと自動的に減速

4. 空中時のクールダウン調整

cpp

```
float Multiplier = CachedMovement->IsFalling() ? 0.5f : 1.0f;
```

```
CooldownRemaining -= DeltaTime * Multiplier;
```

- 地上: 1秒でクールダウン完了
- 空中: 2秒かかる(連続ブーストを防ぐ)

他システムとの連携

PlayerCameraControlComponent

cpp

```
// ブースト開始  
CachedCameraControl->SetFOV(110.0f, false); // フェードイン  
  
// ブースト終了  
CachedCameraControl->ResetFOV(false); // フェードアウト
```

SoundManager

cpp

```
USoundHandle::PlaySE(this, "Boost");
```

デリゲート

cpp

```
OnBoostStarted.Broadcast(); // UI更新、エフェクト発動など  
OnBoostEnded.Broadcast(); // クールダウンUI表示など
```

使用例

```
cpp
// プレイヤーコントローラー側
void APlayerCharacter::InputBoost()
{
    if (BoostComponent)
    {
        BoostComponent->Boost();
    }
}

// UI側でクールダウン表示
void UBoostWidget::NativeConstruct()
{
    BoostComponent->OnBoostStarted.AddDynamic(this, &UBoostWidget::ShowCooldown);
    BoostComponent->OnBoostEnded.AddDynamic(this,
&UBoostWidget::StartCooldownTimer);
}

// エフェクト連携
void APlayerCharacter::SetupBoostEffects()
{
    BoostComponent->OnBoostStarted.AddLambda([this]()
    {
        EffectManager->ActivateEffect(EPostProcessEffectTag::Boost);
    });

    BoostComponent->OnBoostEnded.AddLambda([this]()
    {
        EffectManager->DeactivateEffect(EPostProcessEffectTag::Boost);
    });
}
```

設計のポイント

なぜ最低速度が必要？

- 静止状態からブーストすると不自然
- 助走が必要という戦略性を追加
- パルクールアクションとの相性が良い

なぜ空中でクールダウンが遅い？

- 空中連続ブーストによる無限移動を防止
- 地上での戦略的な使用を促す

なぜ入力で速度維持？

- プレイヤーに制御感を与える
- ブースト後も勢いを活かせる
- 自然な減速演出